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

Logic-Based Ontology Engineering
Part 4: Ontology Maintenance
The Ontology Life Cycle

- User Requirements
- Knowledge Acquisition
- Formalization
- Integration
- Evaluation
- Documentation
- Usage
- Maintenance
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

Part 1: Introduction
Part 2: Ontology Creation
Part 3: Ontology Integration
Part 4: Ontology Maintenance
   4.1 Debugging
   4.2 Modularization
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 $A \in \mathcal{C}$.

Other errors are less obvious:

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

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 $\alpha$ with $\mathcal{O} \models \alpha$, a justification for $\alpha$ in $\mathcal{O}$ is a subset $\mathcal{J} \subseteq \mathcal{O}$ such that

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

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

{\{A \equiv B \cap \exists r.C, \ B \subseteq C, \ \exists r.\top \subseteq D, \ D \subseteq \neg C, \ A \subseteq \neg D, \ C \cap \exists r.\neg B \subseteq \bot\}\}

has two justifications for $A \subseteq \bot$:

{\{A \equiv B \cap \exists r.C, \ \exists r.\top \subseteq D, \ A \subseteq \neg D\}\}

{\{A \equiv B \cap \exists r.C, \ C \cap \exists r.\neg B \subseteq \bot\}\}

Each justification provides an explanation for the error $\alpha$. 
Given an incoherent ontology $\mathcal{O}$, a justification for incoherence of $\mathcal{O}$ is a minimal subset $\mathcal{J} \subseteq \mathcal{O}$ that is incoherent. We denote by $\text{Just}_\mathcal{O}(\bot)$ the set of all justifications for incoherence of $\mathcal{O}$.

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

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

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

$$\mathcal{O} = \{A \sqsubseteq B, \; B \sqsubseteq \bot\}$$

$\text{Just}_\mathcal{O}(A \sqsubseteq \bot) = \{\{A \sqsubseteq B, \; B \sqsubseteq \bot\}\}$

$\text{Just}_\mathcal{O}(\bot) = \{\{B \sqsubseteq \bot\}\}$

Algorithms to compute $\text{Just}_\mathcal{O}(A \sqsubseteq \bot)$ can often be easily adapted to compute $\text{Just}_\mathcal{O}(\bot)$. 
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 $\alpha$ with $\mathcal{O} \models \alpha$, a diagnosis for $\alpha$ in $\mathcal{O}$ is a subset $\mathcal{D} \subseteq \mathcal{O}$ such that

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

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

\[
\{A \equiv B \cap \exists r.C, \ B \subseteq C, \ \exists r.\top \subseteq D, \ D \subseteq \neg C, \ A \subseteq \neg D, \ C \cap \exists r.\neg B \subseteq \bot\}\n\]

has three diagnoses for $A \subseteq \bot$:

\[
\begin{align*}
\{A \equiv B & \cap \exists r.C \\
\{ & \exists r.\top \subseteq D, \ C \cap \exists r.\neg B \subseteq \bot \}
\end{align*}
\]

\[
\{ & A \subseteq \neg D, \ C \cap \exists r.\neg B \subseteq \bot \}
\]
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.

\[
\{A \equiv B \land \exists r. C, \quad B \subseteq C, \quad \exists r. T \subseteq D, \quad D \subseteq \neg C, \quad A \subseteq \neg D, \quad C \land \exists r. \neg B \subseteq \bot\}
\]

**Justifications:**
\[
\begin{align*}
\{A \equiv B \land \exists r. C, \quad \exists r. T \subseteq D, \quad A \subseteq \neg D\} \\
\{A \equiv B \land \exists r. C, \quad C \land \exists r. \neg B \subseteq \bot\}
\end{align*}
\]

**Diagnoses:**
\[
\begin{align*}
\{A \equiv B \land \exists r. C\} \\
\{\exists r. T \subseteq D, \quad C \land \exists r. \neg B \subseteq \bot\} \\
\{A \subseteq \neg D, \quad C \land \exists r. \neg B \subseteq \bot\}
\end{align*}
\]
Minimal Hitting Sets

Given a finite universe \( U \) and a collection of subsets \( S = \{S_1, \ldots, S_n\} \) of \( U \), a hitting set for \( S \) 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 \( \text{MHS}_U(S) \) the set of all minimal hitting sets for \( S \) in \( U \).

For us, the universe is \( \emptyset \) and \( S \) is the set of all justifications.

Lemma (Minimal Hitting Sets of Justifications)

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

\[
\text{Diag}_\mathcal{O}(\alpha) = \text{MHS}_\mathcal{O}(\text{Just}_\mathcal{O}(\alpha)).
\]

Proof: Blackboard.

Exercise: Prove that \( \text{Just}_\mathcal{O}(\alpha) = \text{MHS}_\mathcal{O}(\text{Diag}_\mathcal{O}(\alpha)) \).
Hitting Set Trees

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

$$\text{Just}_\varnothing(\alpha) = \{\{A \equiv B \cap \exists r. C, \ \exists r. T \subseteq D, \ A \subseteq \neg D\}, \ \{A \equiv B \cap \exists r. C, \ C \cap \exists r^- B \subseteq \bot\}\}$$

$$\{A \equiv B \cap \exists r. C, \ C \cap \exists r^- B \subseteq \bot\}$$

\[\begin{array}{c}
A \equiv B \cap \exists r. C \\
\checkmark \\
\times \\
A \equiv B \cap \exists r. C \\
\exists r. T \subseteq D \\
\checkmark \\
\checkmark \\
\end{array}\]
Hitting Set Trees

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

- Nodes of the tree are labeled with $S \in S$, 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 $S \in S$.
- Every node labeled with some $S \in S$ has an outgoing edge labeled with $e$, for every $e \in S$.
- Every new node $v$ has a label $S \in S$ such that $S \cap H(v) = \emptyset$. If there is no such $S$, then $H(v)$ is a hitting set for $S$ 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 $S$.
- Explore the tree breadth-first to find smaller hitting sets first.
The Hitting Set Tree Algorithm

**Input:** Universe $U$, collection of sets $S$

**Output:** The set $\text{MHS}_U(S)$

- 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 $✓$ such that $H(w) \subseteq H(v)$, then label $v$ with $×$
  - Otherwise, if there is a set $S \in S$ 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 $✓$

- Return the set of all sets $H(v)$ for which $v$ is labeled with $✓$
Correctness of the HST Algorithm

The algorithm is **nondeterministic**: For each node \( x \), there may be several possible labels \( S \in S \) 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 \( S \), we have

\[
\text{HSTAlgorithm}(U, S) = \text{MHS}_U(S).
\]

**Proof**: Blackboard.

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

How can we compute \( \text{Just}_O(\alpha) \) in the first place?

**Black-box algorithms:** Use a reasoner for deciding \( O \models \alpha \) 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 \( O \models \alpha \) to “trace” the axioms from \( O \) that are used to derive \( \alpha \).

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

This algorithm needs \textit{exponentially} many calls to the reasoner. Can we do better?

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

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

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

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

$$\{A \sqsubseteq B_1 \cap C_1, B_1 \sqcup C_1 \sqsubseteq B_2 \cap C_2, \ldots, B_{n-1} \sqcup C_{n-1} \sqsubseteq B_n \cap C_n, B_n \sqcup C_n \sqsubseteq D\}$$
A Black-Box Algorithm for Single Justifications (I)

“Binary search” for a justification for $\alpha$ in $\mathcal{O}$: \texttt{SingleJustification}(\emptyset, \mathcal{O}, \alpha)

Algorithm (SingleJustification)

\begin{itemize}
  \item Input: Ontologies $\mathcal{O}_1$, $\mathcal{O}_2$, axiom $\alpha$ such that $\mathcal{O}_1 \not\models \alpha$ and $\mathcal{O}_1 \cup \mathcal{O}_2 \models \alpha$
  \item Output: A minimal subset $\mathcal{O}_2'$ $\subseteq \mathcal{O}_2$ such that in $\mathcal{O}_1 \cup \mathcal{O}_2' \models \alpha$
    \begin{itemize}
      \item If $|\mathcal{O}_2| = 1$, then return $\mathcal{O}_2$
      \item Split $\mathcal{O}_2$ into $\mathcal{O}_l$ and $\mathcal{O}_r$
      \item If $\mathcal{O}_1 \cup \mathcal{O}_x \models \alpha$ for $x \in \{l, r\}$, return \texttt{SingleJustification}(\mathcal{O}_1, \mathcal{O}_x, \alpha)
      \item $\mathcal{O}_l' := \texttt{SingleJustification}(\mathcal{O}_1 \cup \mathcal{O}_r, \mathcal{O}_l, \alpha)$
      \item $\mathcal{O}_r' := \texttt{SingleJustification}(\mathcal{O}_1 \cup \mathcal{O}_l', \mathcal{O}_r, \alpha)$
      \item Return $\mathcal{O}_l' \cup \mathcal{O}_r'$
    \end{itemize}
\end{itemize}

(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 $\alpha$, it first finds a minimal subset of $\mathcal{O}_l$ that, together with $\mathcal{O}_r$, still entails $\alpha$, 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 $\alpha$ with $\mathcal{O} \models \alpha$, we have

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

**Proof:** Blackboard.
Computing a single justification for $\alpha$ is not enough for repairing the error, because there may be other causes for the entailment of $\alpha$.

An efficient algorithm to compute all justifications in $\text{Just}_\mathcal{O}(\alpha)$ based on $\text{HSTAlgorithm}$ (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 $\text{HSTAlgorithm}$ for justifications and diagnoses:

- Nodes $\nu$ are now labeled with justifications $\hat{\gamma}$
- Edges are labeled with axioms $\alpha'$
- $\mathcal{D}(\nu)$ denotes the set of all edge labels on the path from the root to $\nu$
- We are not interested in diagnoses, but in justifications
A Black-Box Algorithm for All Justifications (II)

Algorithm (AllJustifications)

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

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

- 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 $\checkmark$ such that $\mathcal{D}(w) \subseteq \mathcal{D}(v)$, then label $v$ with $\times$
  - Otherwise, if $\mathcal{O} \setminus \mathcal{D}(v) \models \alpha$, then
    - Label $v$ with $\text{SingleJustification}(\emptyset, \mathcal{O} \setminus \mathcal{D}(v), \alpha)$
    - For each axiom $\alpha' \in \bar{\mathcal{J}}$, create a successor $w$ of $v$ in $T$ and label the edge from $v$ to $w$ with $\alpha'$
  - Otherwise, label $v$ with $\checkmark$
- Return the set of all node labels in $T$ (except $\checkmark$ and $\times$)
A Black-Box Algorithm for All Justifications (III)

Lemma (Correctness of AllJustifications)

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

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

Proof: Blackboard.

- If the tree already contains a justification $\tilde{\mathcal{J}}$ with $\mathcal{D}(\nu) \cap \tilde{\mathcal{J}} = \emptyset$, it can be reused as the label for $\nu$
- **SingleJustification** is then only called once for each justification $\tilde{\mathcal{J}} \in \text{Just}_\mathcal{O}(\alpha)$
- 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 \( \alpha \).

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 \( \alpha \).

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

\[
\{ A \equiv B \sqcap \exists r.C, \ B \sqsubseteq C, \ \exists r.\top \sqsubseteq D, \ D \sqsubseteq \neg C, \ A \sqsubseteq \neg D, \ C \sqcap \exists r.\neg B \sqsubseteq \bot \} \\
\downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \\
\{ \ p_1, \ p_2, \ p_3, \ p_4, \ p_5, \ p_6 \ \}
\]
Monotone Boolean Formulas

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

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

A valuation over $lab(O)$ is a subset $V \subseteq lab(O)$.

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

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

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

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

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

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

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

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

$$\{ \neg A \equiv B \land \exists r. C, \quad B \subseteq C, \quad \exists r. \top \subseteq D, \quad D \subseteq \neg C, \quad A \subseteq \neg D, \quad C \land \exists r. \neg B \subseteq \bot \}$$

$$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow$$

$$\{ p_1, \quad p_2, \quad p_3, \quad p_4, \quad p_5, \quad p_6 \}$$

$$p_1 \land ((p_3 \land p_5) \lor p_6)$$

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

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

These valuations induce the two justifications

$$\mathcal{O}_{V_1} = \{ \neg A \equiv B \land \exists r. C, \quad \exists r. \top \subseteq D, \quad A \subseteq \neg D \} \quad \text{and}$$

$$\mathcal{O}_{V_2} = \{ \neg A \equiv B \land \exists r. C, \quad C \land \exists r. \neg B \subseteq \bot \}.$$
From Pinpointing Formulas to Justifications

Recall:

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

Lemma (Pinpointing Formulas)

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

Proof: Exercise.
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 $A, B \in C$
- 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

\[
A_1 \sqcap \cdots \sqcap A_n \sqsubseteq B \quad A \sqsubseteq \exists r.B \quad \exists r.A \sqsubseteq B
\]

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

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 $A, B, C, D, A_1, \ldots, A_n \in \mathbf{C}(\mathcal{O})$ and $r, s \in \mathbf{R}(\mathcal{O})$:

- **(CR1)** \( A \sqsubseteq A \)
- **(CR2)** \( A \sqsubseteq \top \)
- **(CR3)** \[
\begin{align*}
    & A \sqsubseteq A_1 \\
    & \ldots \\
    & A \sqsubseteq A_n \\
\hline
    & A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B
\end{align*}
\]
- **(CR4)** \[
\begin{align*}
    & A \sqsubseteq \exists r.B \\
    & B \sqsubseteq C \\
    & \exists r.C \sqsubseteq D
\end{align*}
\]
- **(CR5)** \[
\begin{align*}
    & A \sqsubseteq \exists r.B \\
    & r \sqsubseteq s
\end{align*}
\]

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 $A, B \in \mathcal{C}(\mathcal{O})$, 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 $\text{lab}(\mathcal{O})$, with the goal of computing pinpointing formulas for all GCIs $A \sqsubseteq B$ between concept names $A, B \in \mathcal{C}(\mathcal{O})$.

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

Initially, the labeled ontology $\mathcal{O}_\ell$ contains all labeled axioms $\alpha^{\text{lab}(\alpha)}$, where $\alpha \in \mathcal{O}$. 
The Pinpointing Algorithm for \( \mathcal{ELH} \)

\[
\begin{align*}
(A \sqsubseteq A)^{\text{true}} & \quad (\text{CR1}) \\
(A \sqsubseteq \top)^{\text{true}} & \quad (\text{CR2}) \\
(A \sqsubseteq A_1)^{\varphi_1} & \quad \ldots \quad (A \sqsubseteq A_n)^{\varphi_n} & (A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B)^{\varphi} & \quad (\text{CR3}) \\
(A \sqsubseteq B)^{\varphi_1 \land \ldots \land \varphi_n \land \varphi} & \quad (A \sqsubseteq \exists r.B)^{\varphi_1} & (B \sqsubseteq C)^{\varphi_2} & (\exists r.C \sqsubseteq D)^{\varphi_3} & \quad (\text{CR4}) \\
(A \sqsubseteq D)^{\varphi_1 \land \varphi_2 \land \varphi_3} & \quad (A \sqsubseteq \exists r.B)^{\varphi_1} & (r \sqsubseteq s)^{\varphi_2} & \quad (\text{CR5}) \\
(A \sqsubseteq \exists s.B)^{\varphi_1 \land \varphi_2} & \quad (A \sqsubseteq \exists r.B)^{\varphi_1} & (r \sqsubseteq s)^{\varphi_2} & \quad (\text{CR5}) \end{align*}
\]

For a new conclusion \( \alpha^\varphi \):

- if \( \mathcal{O}_\ell \) does not already contain a labeled axiom \( \alpha^\psi \), then add \( \alpha^\varphi \) to \( \mathcal{O}_\ell \).
- if \( \mathcal{O}_\ell \) already contains \( \alpha^\psi \) and \( \varphi \) does not imply \( \psi \), then replace \( \alpha^\psi \) in \( \mathcal{O}_\ell \) with \( \alpha^\psi \lor \varphi \).
Example of Pinpointing for $\mathcal{ELH}$

Ontology $\mathcal{O}$:

\[
\begin{align*}
A & \sqsubseteq \exists r. A \\
A & \sqsubseteq Y \\
\exists r. Y & \sqsubseteq B \\
Y & \sqsubseteq B
\end{align*}
\]

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

\[
\begin{align*}
(A \sqsubseteq \exists r. A)^{p_1} & \quad (A \sqsubseteq Y)^{p_2} & \quad (\exists r. Y \sqsubseteq B)^{p_3} & \quad (Y \sqsubseteq B)^{p_4}
\end{align*}
\]

Rule applications:

\[
\frac{(A \sqsubseteq Y)^{p_2} \quad (Y \sqsubseteq B)^{p_4}}{(A \sqsubseteq B)^{p_2 \land p_4}} \quad \text{(CR3)}
\]

\[
\frac{(A \sqsubseteq \exists r. A)^{p_1} \quad (A \sqsubseteq Y)^{p_2} \quad (\exists r. Y \sqsubseteq B)^{p_3}}{(A \sqsubseteq B)^{(p_2 \land p_4) \lor (p_1 \land p_2 \land p_3)}} \quad \text{(CR4)}
\]

$(p_2 \land p_4) \lor (p_1 \land p_2 \land p_3)$ 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 $A, B \in \mathcal{C}(\mathcal{O})$, 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.
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):

\[
\begin{align*}
\hat{C} \sqsubseteq \hat{D} & \quad \rightarrow \quad \hat{C} \sqsubseteq A, \ A \sqsubseteq \hat{D} \\
B \sqsubseteq C_1 \cap \cdots \cap C_n & \quad \rightarrow \quad B \sqsubseteq C_1, \ldots, B \sqsubseteq C_n \\
B_1 \cap \cdots \cap B_n \cap \hat{C} \sqsubseteq D & \quad \rightarrow \quad \hat{C} \sqsubseteq A, \ B_1 \cap \cdots \cap B_n \cap A \sqsubseteq D \\
B \sqsubseteq \exists r.\hat{C} & \quad \rightarrow \quad B \sqsubseteq \exists r.A, \ A \sqsubseteq \hat{C} \\
\exists r.\hat{C} \sqsubseteq D & \quad \rightarrow \quad \hat{C} \sqsubseteq A, \ \exists r.A \sqsubseteq D
\end{align*}
\]

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)

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

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

The sources of an axiom $\beta \in \mathcal{O}'$ are all axioms of $\mathcal{O}$ from which $\beta$ was obtained via a sequence of normalization rules.

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

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

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

Dealing with Normalization (III)

**Lemma (Pinpointing and Normalization)**

Let $A, B \in \mathcal{C}(\mathcal{O})$, and $\varphi'$ be a pinpointing formula for $A \sqsubseteq B$ in $\mathcal{O}'$. Let $\varphi$ be obtained from $\varphi'$ by replacing each $lab(\beta), \beta \in \mathcal{O}'$, by $lab(\alpha_1) \lor \cdots \lor lab(\alpha_n)$, where $\alpha_1, \ldots, \alpha_n$ are all sources of $\beta$ in $\mathcal{O}$. Then $\varphi$ 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:

\[
\{ A \sqsubseteq B_1 \sqcap C_1, B_1 \sqsubseteq B_2 \sqcap C_2, \ldots, B_{n-1} \sqsubseteq B_n \sqcap C_n, B_n \sqsubseteq D \\
C_1 \sqsubseteq B_2 \sqcap C_2, \ldots, C_{n-1} \sqsubseteq B_n \sqcap C_n, C_n \sqsubseteq D \}
\]

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)

\textbf{Input}: Ontology } \mathcal{O}, \text{ axiom } \alpha \text{ with } \mathcal{O} \models \alpha \\
\textbf{Output}: The set } \text{Just}_{\mathcal{O}}(\alpha)

Is there an output polynomial algorithm for \text{EnumerateAllJustifications}?
**Enumerating vs. Checking Justifications**

**Problem (CheckJustification)**

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

**Output**: “yes” if $\mathcal{J} = \text{Just}_\mathcal{O}(\alpha)$, 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 $P = NP$:

**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:

\[
\{\text{Cow} \sqsubseteq \text{Mammal}, \ \text{Mammal} \sqsubseteq \text{Animal}, \\
\text{Cow} \equiv \forall \text{eats.Grass}, \ \text{Dom(eats)} \sqsubseteq \text{Animal}\} \models \text{Grass} \sqsubseteq \text{Animal}
\]

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

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

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

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

Part 1: Introduction
Part 2: Ontology Creation
Part 3: Ontology Integration
Part 4: Ontology Maintenance
   4.1 Debugging
   4.2 Modularization
4.2 Modularization
Reuse of Ontologies

We want to develop a new ontology

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

and reuse the knowledge from an existing ontology

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

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

\[ \mathcal{O} \models \text{DL} \sqsubseteq \text{Logic}. \]

For convenience, we only consider \( \mathcal{ALC} \) in this section.
Modular Reuse of Ontologies

\[ \mathcal{P} = \{ \text{LogicCourse} \equiv \text{Course} \cap \exists \text{focus}. \text{Logic}, \ \exists \text{focus}. \top \subseteq \text{Course}, \ \text{DLSeminar} \equiv \text{Seminar} \cap \exists \text{focus}. \text{DL}, \ \text{Seminar} \subseteq \text{Course} \} \]

\[ \mathcal{O} = \{ \text{DL} \equiv \text{Language} \cap \exists \text{semantics}. \text{FOSStructure} \cap \exists \text{quantifier}. \text{Forall}, \ \text{FOL} \equiv \text{Language} \cap \exists \text{semantics}. \text{FOSStructure}, \ \text{FOL} \subseteq \text{Logic}, \ \text{Language} \cap \exists \text{quantifier}. \text{Forall} \subseteq \text{FOL}, \ \text{OWL} \subseteq \text{W3CStandard} \cap \exists \text{basedOn}. \text{DL} \} \]

We consider only a subvocabulary \( \Sigma \subseteq C(\mathcal{O}) \cup R(\mathcal{O}) \cup I(\mathcal{O}) \), 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 \( \Sigma \) (it is a \( \Sigma \)-module of \( \mathcal{O} \)).
- \( \mathcal{P} \) does not affect the semantics of the names in \( \Sigma \) (it is \( \Sigma \)-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 \( C \cup R \cup I \) that contains \( \top \) and \( \bot \).

For an ontology \( \mathcal{O} \) or axiom \( \alpha \), \( \text{sig}(\mathcal{O})/\text{sig}(\alpha) \) 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 \( \Sigma \).

Let \( \mathcal{O}_1 \subseteq \mathcal{O}_2 \) be two ontologies and \( \Sigma \) be a signature.

Then \( \mathcal{O}_2 \) is a **\( \Sigma \)-conservative extension (\( \Sigma \)-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 \( \Sigma \).

Note that \( \mathcal{O}_1 \) can contain more names than those in \( \Sigma \), and their semantics is allowed to change.
Safety and Modules

Let $O_1 \subseteq O_2$ be two ontologies and $\Sigma$ be a signature. Then $O_2$ is a $\Sigma$-conservative extension ($\Sigma$-CE) of $O_1$ if, for every model $I$ of $O_1$, there is a model $J$ of $O_2$ such that $I|_{\Sigma} = J|_{\Sigma}$.

Suppose we want to import the knowledge about $\Sigma$ from $O$ into $P$.

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

$P$ does not affect the semantics of the names in $\Sigma$ given by $O$.

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

When we are only interested in $\Sigma$, we can import $M$ instead of $O$. 
Robustness under Replacement

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

**Lemma (Replacement)**

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

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

**Proof:** Blackboard.

**Lemma (Characterization of Safety)**

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

**Lemma (Characterization of Modules)**

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

**Proof:** Exercise.
Example: Safety

\[ \mathcal{P} = \{ \text{LogicCourse} \equiv \text{Course} \land \exists \text{focus.Logic}, \quad \exists \text{focus.}\top \subseteq \text{Course}, \]
\[ \text{DLSeminar} \equiv \text{Seminar} \land \exists \text{focus.DL}, \quad \text{Seminar} \subseteq \text{Course} \} \]

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

\[ \mathcal{P}_2 = \{ \text{LogicCourse} \equiv \text{Course} \land \exists \text{focus.Logic}, \quad \exists \text{focus.}\top \subseteq \text{Course}, \]
\[ \text{DLSeminar} \equiv \text{Seminar} \land \exists \text{focus.DL}, \quad \text{Seminar} \subseteq \text{Course}, \]
\[ \text{Course} \land (\text{Logic} \land \text{DL}) \subseteq \bot, \]
\[ \forall \text{focus.DL} \subseteq \exists \text{focus.Logic} \} \]

is not \( \Sigma \)-safe (see blackboard).
Example: Modules

\[ \mathcal{O} = \{ \text{DL } \equiv \text{Language } \cap \exists \text{ semantics. FOStructure } \cap \exists \text{ quantifier. Forall}, \]
\[ \text{FOL } \equiv \text{Language } \cap \exists \text{ semantics. FOStructure, FOL } \subseteq \text{ Logic}, \]
\[ \text{Language } \cap \exists \text{ quantifier. Forall } \subseteq \text{ FOL}, \]
\[ \text{OWL } \subseteq \text{ W3CStandard } \cap \exists \text{basedOn. DL} \} \]

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

All justifications of \( \text{DL } \subseteq \text{ Logic!} \) (see blackboard)

\[ \mathcal{M}_1 = \{ \text{DL } \equiv \text{Language } \cap \exists \text{ semantics. FOStructure } \cap \exists \text{ quantifier. Forall,} \]
\[ \text{FOL } \equiv \text{Language } \cap \exists \text{ semantics. FOStructure, FOL } \subseteq \text{ Logic} \} \]
\[ \mathcal{M}_2 = \{ \text{DL } \equiv \text{Language } \cap \exists \text{ semantics. FOStructure } \cap \exists \text{ quantifier. Forall,} \]
\[ \text{Language } \cap \exists \text{ quantifier. Forall } \subseteq \text{ FOL, FOL } \subseteq \text{ Logic} \} \]

\( \Sigma \)-modules for \( \mathcal{O} \) are related to justifications for all possible axioms over \( \Sigma \).
Example: “Depleting” Modules

The remaining axioms in

\[ \mathcal{O} \setminus \mathcal{M}_1 = \{ \text{FOL} \equiv \text{Language} \cap \exists \text{semantics}.\text{FOStructure}, \] 

can still influence the interpretation of \( \text{sig}(\mathcal{M}_1) = \{ \text{DL}, \text{Logic}, \text{Language}, \text{FOStructure}, \text{Forall}, \text{FOL}, \text{semantics}, \text{quantifier} \} \). To avoid this, we consider so-called depleting modules of \( \mathcal{O} \):

\[ \mathcal{M}_3 = \{ \text{DL} \equiv \text{Language} \cap \exists \text{semantics}.\text{FOStructure} \cap \exists \text{quantifier}.\text{Forall}, \]
\[ \text{FOL} \equiv \text{Language} \cap \exists \text{semantics}.\text{FOStructure}, \quad \text{FOL} \subseteq \text{Logic}, \]
\[ \text{Language} \cap \exists \text{quantifier}.\text{Forall} \subseteq \text{FOL} \} \]

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

In other words,

\[ \{ \text{OWL} \subseteq \text{W3CStandard} \cap \exists \text{basedOn.DL} \} \text{ is } [\Sigma \cup \text{sig}(\mathcal{M}_3)]\text{-safe}. \]
Properties of $\Sigma$-Conservative Extensions

Before we formally define depleting modules, we introduce some important properties of $\Sigma$-CEs:

- **Lemma (Monotonicity)**
  
  Let $\mathcal{O}_2$ be a $\Sigma$-CE of $\mathcal{O}_1$.
  
  a) If $\Sigma' \subseteq \Sigma$, then $\mathcal{O}_2$ is a $\Sigma'$-CE of $\mathcal{O}_1$.
  
  b) If $\mathcal{O}_1 \subseteq \mathcal{O}'_2 \subseteq \mathcal{O}_2$, then $\mathcal{O}'_2$ is a $\Sigma$-CE of $\mathcal{O}_1$.

- **Lemma (Transitivity)**
  
  If $\mathcal{O}_2$ is a $\Sigma$-CE of $\mathcal{O}_1$ and $\mathcal{O}_3$ is a $\Sigma$-CE of $\mathcal{O}_2$, then $\mathcal{O}_3$ is a $\Sigma$-CE of $\mathcal{O}_1$.

- **Lemma (Reflexivity)**
  
  For every ontology $\mathcal{O}$ and signature $\Sigma$, $\mathcal{O}$ is a $\Sigma$-CE of itself.

**Proof**: Exercise.
Depleting Modules

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

Lemma (Depleting Modules are Modules)

If \( \mathcal{M} \) is a depleting \( \Sigma \)-module of \( \mathcal{O} \), then \( \mathcal{M} \) is a \( \Sigma \)-module of \( \mathcal{O} \).

Proof: Blackboard.

Lemma (Depleting Modules Contain All Minimal Modules)

Every depleting \( \Sigma \)-module of \( \mathcal{O} \) contains all minimal \( \Sigma \)-modules of \( \mathcal{O} \).

Proof: Blackboard.

- A (minimal) depleting module \( \mathcal{M} \) can be very large, because it includes all axioms relevant for the semantics of \( \Sigma \cup \text{sig}(\mathcal{M}) \).
- 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 $\Sigma$-Conservative Extensions

In $\mathcal{E}L$, deciding whether $\mathcal{O}_2$ is a $\Sigma$-CE of $\mathcal{O}_1$ is

- **undecidable**, even if $\mathcal{O}_1 = \emptyset$
- **co-NEXPSPACE-complete** if $\Sigma \subseteq \mathcal{C}$
- **in P** if $\mathcal{O}_1 = \emptyset$ and $\Sigma \subseteq \mathcal{C}$

Due to this high complexity, we consider *approximations*.

**Goal**: Find another definition of $\sigma$-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., $\text{DL} \subseteq \text{Logic}$).
Let $\mathcal{O}_1 \subseteq \mathcal{O}_2$ be two ontologies and $\Sigma$ be a signature.

Then $\mathcal{O}_2$ is a **deductive $\Sigma$-conservative extension (d-$\Sigma$-CE)** of $\mathcal{O}_1$ if, for every GCI $\alpha$ with $\text{sig}(\alpha) \subseteq \Sigma$,

$\mathcal{O}_2 \models \alpha$ implies $\mathcal{O}_1 \models \alpha$.

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

**d-$\Sigma$-safety** and **d-$\Sigma$-modules** are defined as before, by replacing m-$\Sigma$-CE with d-$\Sigma$-CE.
Lemma

If $O_2$ is an m-$\Sigma$-CE of $O_1$, then it is also a d-$\Sigma$-CE of $O_1$.

Proof: Exercise.

In particular, finding an axiom $\alpha$ with $\text{sig}(\alpha) \subseteq \Sigma$, $O_2 \models \alpha$, and $O_1 \not\models \alpha$ proves that $O_2$ is not an m-$\Sigma$-CE of $O_1$.

The other direction does not hold:

$M = \{ T \sqsubseteq \exists r. T \land \exists s. T \}$ is a d-$\Sigma$-module of

$O = M \cup \{ T \sqsubseteq \exists r. A \land \exists s. \neg A \}$ for $\Sigma = \{ r, s \}$.

But there are models of $M$ that cannot be extended to models of $O$ 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

\[ \mathcal{P}_2 = \{ LogicCourse \equiv \text{Course} \cap \exists \text{focus. Logic}, \quad \exists \text{focus. } \top \subseteq \text{Course}, \]
\[ \quad \text{DLSeminar} \equiv \text{Seminar} \cap \exists \text{focus. DL}, \quad \text{Seminar} \subseteq \text{Course}, \]
\[ \quad \text{Course} \cap (\text{Logic} \cap \text{DL}) \subseteq \bot, \]
\[ \quad \forall \text{focus. DL} \subseteq \exists \text{focus. Logic} \} \]

entails \( \text{Logic} \cap \text{DL} \subseteq \bot \), which is not entailed by \( \emptyset \). Thus, it is not d-\( \Sigma \)-safe, and therefore not m-\( \Sigma \)-safe.

All d-\( \Sigma \)-modules (and all m-\( \Sigma \)-modules) of

\[ \mathcal{O} = \{ \text{DL} \equiv \text{Language} \cap \exists \text{semantics. FOStructure} \cap \exists \text{quantifier. Forall}, \]
\[ \quad \text{FOL} \equiv \text{Language} \cap \exists \text{semantics. FOStructure}, \quad \text{FOL} \subseteq \text{Logic}, \]
\[ \quad \text{Language} \cap \exists \text{quantifier. Forall} \subseteq \text{FOL}, \]
\[ \quad \text{OWL} \subseteq \text{W3CStandard} \cap \exists \text{basedOn. DL} \} \]

must entail \( \text{DL} \subseteq \text{Logic} \).
Properties of d-Σ-CE

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

- Monotonicity
- Transitivity
- Reflexivity
- Results about depleting modules

The following properties do not hold:

- Robustness under replacement
- Characterizations of safety and modules

\[ O_2 = \{ A \subseteq \exists r.B \} \text{ is a d-Σ-CE of } \emptyset, \text{ for } \Sigma = \{ A, B \}. \]

For \( O = \{ A \equiv \top, B \equiv \bot \} \), we have \( O \cup O_2 \models \top \subseteq \bot \), but \( O \not\models \top \subseteq \bot \), i.e., \( O \cup O_2 \) is not a d-Σ-CE of \( O \).
Complexity of d-$\Sigma$-CE

Deciding d-$\Sigma$-CE:

- **undecidable** for $\mathit{SROIQ}$, i.e., OWL 2 DL
- **2-EXPTIME-complete** for $\mathit{ALC}$
- **EXPTIME-complete** for $\mathit{EL}$

Deciding d-$\Sigma$-modules:

- **undecidable** for $\mathit{ALC}$ 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 $\Sigma$-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 $[\Sigma \cup \text{sig}(\mathcal{M})]$-safe?

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

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

An axiom $\alpha$ is $\emptyset$-$\Sigma$-local if it is satisfied in every $\emptyset$-$\Sigma$-local interpretation.

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

We only need to check each axiom $\alpha \in \mathcal{O}$ separately!

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

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

$\emptyset$-\(\Sigma\)$-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 $\emptyset$-$\Sigma$-safe iff it is $\emptyset$-$\Sigma$-local.

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

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

\[\text{OWL} \subseteq \text{W3CStandard} \cap \exists \text{basedOn.DL} \]

is $\emptyset$-$\Sigma$-local for $\Sigma = \{\text{DL, Logic, Language, FOSstructure, Forall, FOL, semantics, quantifier}\}$
Theorem 1: If \( \mathcal{O}_2 \) is an \( \emptyset \)-\( \Sigma \)-CE of \( \mathcal{O}_1 \), then \( \mathcal{O}_2 \) is an \( m \)-\( \Sigma \)-CE of \( \mathcal{O}_1 \).

Proof: Blackboard.

Again, the other direction does not hold:

\[
M_1 = \{ \text{DL} \equiv \text{Language} \cap \exists \text{semantics}. \text{FOS} \text{Structure} \cap \exists \text{quantifier}. \text{Forall}, \text{FOL} \equiv \text{Language} \cap \exists \text{semantics}. \text{FOS} \text{Structure}, \text{FOL} \subseteq \text{Logic} \}
\]

is an \( m \)-\( \Sigma \)-module of \( \mathcal{O} \) for \( \Sigma = \{ \text{DL}, \text{Logic} \} \), but it is not depleting. The axiom \( \text{Language} \cap \exists \text{quantifier}. \text{Forall} \subseteq \text{FOL} \) is not an \( \emptyset \)-[\( \Sigma \cup \text{sig}(M_1) \)]-local.

Thus, \( M_1 \) is not an \( \emptyset \)-\( \Sigma \)-module of \( \mathcal{O} \).
Testing Locality

Idea: To test locality, we replace all concept names outside of $\Sigma$ with $\bot$ (because in $\emptyset$-$\Sigma$-local interpretations they are equivalent to $\bot$), and similarly for role names.

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

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

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

Lemma (Testing Locality)

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

Proof: Exercise.
Complexity of $\emptyset$-$\Sigma$-CE

Example:

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

$\text{OWL} \subseteq \text{W3CStandard} \cap \exists \text{basedOn} \cdot \text{DL}$ is replaced by $\bot \subseteq \bot \cap \bot$, which is entailed by the empty ontology $\emptyset$.

The complexity of this check is the same as for axiom entailment:

- **2-NEXPTIME-complete** for $\text{SROIQ}$, i.e., OWL 2 DL
- **EXPTIME-complete** for $\text{ALC}$
- **P-complete** for $\mathcal{EL}$

Can we compute “useful” (small, depleting) modules even faster?
Third Approximation: Syntactic-Locality-Based CE

Idea: To check that $\emptyset \models \bot \sqsubseteq \bot \cap \bot$, we do not need a SROIQ reasoner. We can instead use a simple syntactic check on the axiom $\text{OWL} \sqsubseteq \text{W3C Standard} \sqcap \exists \text{basedOn}.\text{DL}$ to determine that it is local.

Intuition: We collect in $C^\bot_\Sigma$ ($C^\top_\Sigma$) all concepts for which it is “easy to prove” that they are equivalent to $\bot$ ($\top$). For example, if $C \in C^\bot_\Sigma$ and $D$ is another concept, then $C \cap D$ also belongs to $C^\bot_\Sigma$.

$$
\begin{align*}
C^\bot_\Sigma & := \bot \mid A^\bot \mid \neg C^\top \mid C \cap C^\bot \mid C^\bot_1 \cup C^\bot_2 \mid \exists r^\bot.C \mid \exists r.C^\bot \\
C^\top_\Sigma & := \top \mid \neg C^\bot \mid C \cup C^\top \mid C^\top_1 \cap C^\top_2 \mid \forall r^\bot.C \mid \forall r.C^\top
\end{align*}
$$

where $A^\bot \in C \setminus \Sigma$, $C^\bot_{(i)} \in C^\bot_\Sigma$, $C^\top_{(i)} \in C^\top_\Sigma$, $r^\bot \in R \setminus \Sigma$, $r \in \Sigma$, and $C$ is a concept.

An axiom $\alpha$ is $\bot$-$\Sigma$-local if it is of the form $C \subseteq C^\top$, $C^\bot \subseteq C$, or $\alpha : C^\top$.

An ontology $\mathcal{O}$ is $\bot$-$\Sigma$-local if all axioms $\alpha \in \mathcal{O}$ are $\bot$-$\Sigma$-local.
\(-\)-Locality-Based vs. \(\emptyset\)-Locality-Based CE

We can define \(-\Sigma\)-CE, \(-\Sigma\)-safety, and \(-\Sigma\)-modules as we did for \(\emptyset\)-locality. \(-\Sigma\)-CEs are monotone, transitive, reflexive, and robust under replacement.

**Lemma**

If \(O_2\) is a \(-\Sigma\)-CE of \(O_1\), then \(O_2\) is an \(\emptyset\)-\(\Sigma\)-CE of \(O_1\).

**Proof**: Blackboard.

As usual, the other direction does not hold:

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

\[\Sigma = \{\text{DL, Logic, Language, FOStructure, Forall, FOL, semantics, quantifier}\}\]

\(\text{OWL} \sqsubseteq \text{W3CStandard} \cap \exists \text{basedOn. DL}\) is \(-\Sigma\)-local since it is of the form \(C \perp \sqsubseteq C\), because \(\text{OWL} \notin \Sigma\).
## Overview of Conservative Extensions

<table>
<thead>
<tr>
<th>Name</th>
<th>synt. locality ⇒ sem. locality ⇒ model-based ⇒ deductive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbol</td>
<td>⊥</td>
</tr>
<tr>
<td>Monotonicity, Transitivity, Reflexivity</td>
<td>✓</td>
</tr>
<tr>
<td>Robustness under Replacement</td>
<td>✓</td>
</tr>
<tr>
<td>All Modules are Depleting</td>
<td>✓</td>
</tr>
<tr>
<td>Complexity (for $\mathcal{ALC}$)</td>
<td>P</td>
</tr>
</tbody>
</table>
Extracting Depleting Modules

- Instead of deciding whether $\mathcal{M}$ is an $x$-$\Sigma$-module of $\mathcal{O}$, we want to extract a minimal $x$-$\Sigma$-module from $\mathcal{O}$ (for some $x \in \{m, d, \emptyset, \bot\}$).

- For depleting modules, we only need to ensure that $\mathcal{O} \setminus \mathcal{M}$ is $x$-[$\Sigma \cup \text{sig}(\mathcal{M})$]-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 $x$-$\Sigma$-safety.
An Algorithm for Extracting Depleting Modules

Algorithm (Black-Box Module Extraction)

Input: Ontology $\mathcal{O}$, signature $\Sigma$, $x \in \{m, d, \emptyset, \bot\}$
Output: A depleting $x$-$\Sigma$-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 $x$-$[\Sigma \cup \text{sig} (\mathcal{M})]$-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 $\alpha$) may not be $x$-$[\Sigma \cup \text{sig}(\mathcal{M})]$-safe anymore.
Correctness

Lemma (Correctness of Black-Box Module Extraction)
If x-safety is decidable, the black-box module extraction algorithm computes a depleting $x$-$\Sigma$-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 $x$-$\Sigma$-module of $\mathcal{O}$, which is computed by the black-box module extraction algorithm.

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

**Algorithm (Locality-Based Module Extraction)**

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

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

- $\mathcal{M} := \emptyset$; $\mathcal{W} := \emptyset$
- **While** $\mathcal{M} \cup \mathcal{W} \neq \emptyset$:
  - Choose an axiom $\alpha \in \mathcal{O} \setminus (\mathcal{M} \cup \mathcal{W})$
  - If $\alpha$ is $x$-$[\Sigma \cup \text{sig}(\mathcal{M})]$-safe, then
    - $\mathcal{W} := \mathcal{W} \cup \{\alpha\}$
  - Otherwise,
    - $\mathcal{M} := \mathcal{M} \cup \{\alpha\}$; $\mathcal{W} := \emptyset$
- **Return** $\mathcal{M}$
Correctness

Lemma (Correctness of Locality-Based Module Extraction)

The locality-based module extraction algorithm computes the unique minimal (depleting) $x$-$\Sigma$-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) $\perp$-$\Sigma$-module of $\mathcal{O}$ can be computed in polynomial time in the size of $\mathcal{O}$. 
Extracting Modules in Practice

- $m$-$\Sigma$-modules can only be computed in special cases, e.g., for $\mathcal{E}\mathcal{L}$ and $\Sigma \subseteq \mathcal{C}$
- $\perp$-$\Sigma$-modules can be computed much faster, even for large and expressive ontologies
- Minimal $\perp$-$\Sigma$-modules are also $m$-$\Sigma$-modules, but not necessarily minimal $m$-$\Sigma$-modules
- In an evaluation, minimal $\perp$-$\Sigma$-modules differed from the minimal depleting $m$-$\Sigma$-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)