Principles of Knowledge Representation and Reasoning

Description Logics – Reasoning Services and Reductions

Bernhard Nebel, Stefan Wölfl, and Marco Ragni

Albert-Ludwigs-Universität Freiburg

July 12, 2010
1 Motivation

2 Basic Reasoning Services

3 Eliminating the TBox

4 General TBox Reasoning Services

5 General ABox Reasoning Services

6 Summary and Outlook
Example TBox & ABox

Male ⊑ ¬Female
Human ⊑ Living_entity
Woman ⊑ Human ⊓ Female
Man ⊑ Human ⊓ Male
Mother ⊑ Woman ⊓ ∃has-child.Human
Father ⊑ Man ⊓ ∃has-child.Human
Parent ⊑ Father ⊓ Mother
Grandmother
    ⊑ Woman ⊓ ∃has-child.Parent
Mother-without-daughter
    ⊑ Mother ⊓ ∀has-child.Male
Mother-with-many-children
    ⊑ Mother ⊓ (∨ 3 has-child)

DIANA: Woman
ELIZABETH: Woman
CHARLES: Man
EDWARD: Man
ANDREW: Man

DIANA: Mother-without-daughter
(ELIZABETH, CHARLES): has-child
(ELIZABETH, EDWARD): has-child
(ELIZABETH, ANDREW): has-child
(DIANA, WILLIAM): has-child
(CHARLES, WILLIAM): has-child
Motivation: Reasoning Services

- **What do we want to know?**
  - We want to check whether the *knowledge base* is reasonable
    - Is each defined concept in a TBox satisfiable?
    - Is a given TBox satisfiable?
    - Is a given ABox satisfiable?
  - What can we **conclude** from the represented knowledge?
    - Is concept $X$ subsumed by concept $Y$?
    - Is an object *a* instance of a concept $X$?

- These problems can be **reduced** to logical satisfiability or implication – using the logical semantics.

- We take a different route: We will try to simplify these problems and then we specify *direct inference methods*. 
Motivation: Reasoning Services

- What do we want to know?
  - We want to check whether the *knowledge base* is reasonable
    - Is each defined concept in a TBox satisfiable?
    - Is a given TBox satisfiable?
    - Is a given ABox satisfiable?

- What can we **conclude** from the represented knowledge?
  - Is concept $X$ **subsumed** by concept $Y$?
  - Is an object *a instance* of a concept $X$?

- These problems can be **reduced** to logical satisfiability or implication – using the logical semantics.

- We take a different route: We will try to simplify these problems and then we specify *direct inference methods*. 
Motivation: Reasoning Services

What do we want to know?

We want to check whether the *knowledge base* is reasonable
- Is each defined concept in a TBox satisfiable?
- Is a given TBox satisfiable?
- Is a given ABox satisfiable?

What can we *conclude* from the represented knowledge?
- Is concept $X$ subsumed by concept $Y$?
- Is an object a instance of a concept $X$?

These problems can be *reduced* to logical satisfiability or implication – using the logical semantics.

We take a different route: We will try to simplify these problems and then we specify *direct inference methods*. 
Satisfiability of Concept Descriptions in a TBox

- **Motivation**: Given a TBox $\mathcal{T}$ and a concept description $C$, does $C$ make sense, i.e., is $C$ satisfiable?
- **Test**:
  - Does there exist a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^\mathcal{I} \neq \emptyset$?
  - Is the formula $\exists x : C(x)$ together with the formulas resulting from the translation of $\mathcal{T}$ satisfiable?
- **Example**: Mother-without-daughter $\sqcap$ $\forall$ has-child.Female is unsatisfiable.
Satisfiability of Concept Descriptions in a TBox

**Motivation:** Given a TBox $\mathcal{T}$ and a concept description $C$, does $C$ make sense, i.e., is $C$ satisfiable?

**Test:**
- Does there exist a *model* $\mathcal{I}$ of $\mathcal{T}$ such that $C^\mathcal{I} \neq \emptyset$?
- Is the formula $\exists x : C(x)$ together with the formulas resulting from the translation of $\mathcal{T}$ satisfiable?

**Example:** *Mother-without-daughter $\sqcap$ Female* is unsatisfiable.
Satisfiability of Concept Descriptions (without a TBox)

**Motivation:** Given a concept description $C$ in “isolation”, i.e., in an empty TBox, does $C$ make sense, i.e., is $C$ satisfiable?

**Test:**
- Does there exist an interpretation $\mathcal{I}$ such that $C^\mathcal{I} \neq \emptyset$?
- Is the formula $\exists x : C(x)$ satisfiable?

**Example:** Woman $\sqcap (\leq 0 \text{ has-child}) \sqcap (\geq 1 \text{ has-child})$ is unsatisfiable.
**Satisfiability of Concept Descriptions (without a TBox)**

- **Motivation:** Given a concept description $C$ in “isolation”, i.e., in an empty TBox, does $C$ make sense, i.e., is $C$ satisfiable?

- **Test:**
  - Does there exist an interpretation $I$ such that $C^I \neq \emptyset$?
  - Is the formula $\exists x : C(x)$ satisfiable?

- **Example:** Woman $\sqcap (\leq 0 \text{ has-child}) \sqcap (\geq 1 \text{ has-child})$ is unsatisfiable.
Reduction: Getting Rid of the TBox

- We can reduce satisfiability in a TBox to simple satisfiability.

  **Idea:**
  - Since TBoxes are *cycle-free*, one can understand a concept definition as a kind of “macro”
  - For a given TBox $\mathcal{T}$ and a given concept description $C$, all defined concept symbols appearing in $C$ can be *expanded* until $C$ contains only undefined concept symbols
  - An *expanded* concept description is then satisfiable iff $C$ is satisfiable in $\mathcal{T}$
  - **Problem:** What do we do with partial definitions (using $\sqsubseteq$)?

Reduction: Getting Rid of the TBox

We can reduce satisfiability in a TBox to simple satisfiability.

**Idea:**
- Since TBoxes are *cycle-free*, one can understand a concept definition as a kind of “macro”
- For a given TBox $\mathcal{T}$ and a given concept description $C$, all defined concept symbols appearing in $C$ can be *expanded* until $C$ contains only undefined concept symbols
- An *expanded* concept description is then satisfiable iff $C$ is satisfiable in $\mathcal{T}$
- **Problem:** What do we do with partial definitions (using $\sqsubseteq$)?
A terminology is called normalized when it does not contain definitions using $\sqsubseteq$.

In order to normalize a terminology, replace

$$A \sqsubseteq C$$

by

$$A \equiv A^* \cap C,$$

where $A^*$ is a fresh concept symbol (not appearing elsewhere in $\mathcal{T}$).

If $\mathcal{T}$ is a terminology, the normalized terminology is denoted by $\tilde{\mathcal{T}}$. 

Normalized Terminologies
Normalizing is Reasonable

Theorem (Normalization Invariance)

If $\mathcal{I}$ is a model of the terminology $\mathcal{T}$, then there exists a model $\mathcal{I}'$ of $\tilde{\mathcal{T}}$ (and vice versa) such that for all concept symbols $A$ appearing in $\mathcal{T}$ we have:

$$A^\mathcal{I} = A^{\mathcal{I}'}.$$ 

Proof.

“$\Rightarrow$”: Let $\mathcal{I}$ be a model of $\mathcal{T}$. This model should be extended to $\mathcal{I}'$ so that the freshly introduced concept symbols also get interpretations. Assume $(A \subseteq C) \in \mathcal{T}$, i.e., we have $(A \equiv A^* \cap C) \in \tilde{\mathcal{T}}$. Then set $A^{*\mathcal{I}'} = A^{\mathcal{I}}$. $\mathcal{I}'$ obviously satisfies $\tilde{\mathcal{T}}$ and has the same interpretation for all symbols in $\mathcal{T}$.

“$\Leftarrow$”: Given a model $\mathcal{I}'$ of $\tilde{\mathcal{T}}$, its restriction to symbols of $\mathcal{T}$ is the interpretation we looked for.
# Normalizing is Reasonable

## Theorem (Normalization Invariance)

If $\mathcal{I}$ is a model of the terminology $\mathcal{T}$, then there exists a model $\mathcal{I}'$ of $\tilde{\mathcal{T}}$ (and vice versa) such that for all concept symbols $A$ appearing in $\mathcal{T}$ we have:

$$A^\mathcal{I} = A^{\mathcal{I}'}.$$ 

### Proof.

"$\Rightarrow$": Let $\mathcal{I}$ be a model of $\mathcal{T}$. This model should be extended to $\mathcal{I}'$ so that the freshly introduced concept symbols also get interpretations. Assume $(A \subseteq C') \in \mathcal{T}$, i.e., we have $(A \equiv A^* \cap C') \in \tilde{\mathcal{T}}$. Then set $A^{*\mathcal{I}'} = A^\mathcal{I}$. $\mathcal{I}'$ obviously satisfies $\tilde{\mathcal{T}}$ and has the same interpretation for all symbols in $\mathcal{T}$.

"$\Leftarrow$": Given a model $\mathcal{I}'$ of $\tilde{\mathcal{T}}$, its restriction to symbols of $\mathcal{T}$ is the interpretation we looked for.
Normalizing is Reasonable

**Theorem (Normalization Invariance)**

*If* \( \mathcal{I} \) *is a model of the terminology* \( \mathcal{T} \), *then there exists a model* \( \mathcal{I}' \) *of* \( \tilde{\mathcal{T}} \) *and vice versa* such that for all concept symbols* \( A \) *appearing in* \( \mathcal{T} \) *we have:*

\[
A^\mathcal{I} = A^{\mathcal{I}'}.
\]

**Proof.**

\( \Rightarrow \) : Let \( \mathcal{I} \) be a model of \( \mathcal{T} \). This model should be extended to \( \mathcal{I}' \) so that the freshly introduced concept symbols also get interpretations. Assume \( (A \sqsubseteq C) \in \mathcal{T} \), i.e., we have \( (A \sqsupseteq A^* \sqcap C) \in \tilde{\mathcal{T}} \). Then set \( A^{*\mathcal{I}'} = A^\mathcal{I} \). \( \mathcal{I}' \) obviously satisfies \( \tilde{\mathcal{T}} \) and has the same interpretation for all symbols in \( \mathcal{T} \).

\( \Leftarrow \) : Given a model \( \mathcal{I}' \) of \( \tilde{\mathcal{T}} \), its restriction to symbols of \( \mathcal{T} \) is the interpretation we looked for.
Normalizing is Reasonable

**Theorem (Normalization Invariance)**

*If* $\mathcal{I}$ *is a model of the terminology* $\mathcal{T}$, *then there exists a model* $\mathcal{I}'$ *of* $\tilde{\mathcal{T}}$ *(and vice versa) such that for all concept symbols* $A$ *appearing in* $\mathcal{T}$ *we have:*

$$A^\mathcal{I} = A^{\mathcal{I}'}.$$  

**Proof.**

"$\Rightarrow$": Let $\mathcal{I}$ be a model of $\mathcal{T}$. This model should be *extended* to $\mathcal{I}'$ so that the freshly introduced concept symbols also get interpretations. Assume $(A \sqsubseteq C) \in \mathcal{T}$, i.e., we have $(A \vDash A^* \cap C) \in \tilde{\mathcal{T}}$. Then set $A^{*\mathcal{I}'} = A^\mathcal{I}$. $\mathcal{I}'$ obviously satisfies $\tilde{\mathcal{T}}$ and has the same interpretation for all symbols in $\mathcal{T}$.

"$\Leftarrow$": Given a model $\mathcal{I}'$ of $\tilde{\mathcal{T}}$, its restriction to symbols of $\mathcal{T}$ is the interpretation we looked for.
Normalizing is Reasonable

Theorem (Normalization Invariance)

If $I$ is a model of the terminology $\mathcal{T}$, then there exists a model $I'$ of $\tilde{\mathcal{T}}$ (and vice versa) such that for all concept symbols $A$ appearing in $\mathcal{T}$ we have:

$$A^I = A^{I'}.$$ 

Proof.

“$\Rightarrow$”: Let $I$ be a model of $\mathcal{T}$. This model should be extended to $I'$ so that the freshly introduced concept symbols also get interpretations. Assume $(A \sqsubseteq C) \in \mathcal{T}$, i.e., we have $(A \sqsubseteq A^* \sqcap C) \in \tilde{\mathcal{T}}$. Then set $A^{*I'} = A^I$. $I'$ obviously satisfies $\tilde{\mathcal{T}}$ and has the same interpretation for all symbols in $\mathcal{T}$.

“$\Leftarrow$” Given a model $I'$ of $\tilde{\mathcal{T}}$, its restriction to symbols of $\mathcal{T}$ is the interpretation we looked for.
We say that a *normalized TBox* is **unfolded by one step** when all defined concept symbols on the right sides are replaced by their defining terms.

**Example:** Mother $\equiv$ Woman $\sqcap \ldots$ is unfolded to Mother $\equiv$ (Human $\sqcap$ Female) $\sqcap \ldots$

We write $U(\mathcal{T})$ to denote a one-step unfolding and $U^n(\mathcal{T})$ to denote an *$n$-step unfolding*.

We say $\mathcal{T}$ is **unfolded** if $U(\mathcal{T}) = \mathcal{T}$.

We say that $U^n(\mathcal{T})$ is the **unfolding** of $\mathcal{T}$ if $U^n(\mathcal{T}) = U^{n+1}(\mathcal{T})$. If such an unfolding exists, it is denoted by $\hat{\mathcal{T}}$. 

---

**Motivation**

Basic Reasoning Services

Eliminating the TBox Normalization Unfolding

General TBox Reasoning Services

General ABox Reasoning Services

Summary and Outlook

---

**KRR**

Nebel, Wölf, Ragni
We say that a *normalized TBox* is unfolded by one step when all defined concept symbols on the right sides are replaced by their defining terms.

**Example:** Mother ≡ Woman ⊓ ... is unfolded to Mother ≡ (Human ⊓ Female) ⊓ ...

We write $U(T)$ to denote a one-step unfolding and $U^n(T)$ to denote an *n-step unfolding*.

We say $T$ is unfolded if $U(T) = T$.

We say that $U^n(T)$ is the *unfolding* of $T$ if $U^n(T) = U^{n+1}(T)$. If such an unfolding exists, it is denoted by $\hat{T}$.
Properties of Unfoldings (1): Existence

Theorem (Existence of unfolded terminology)

For each normalized terminology \( \mathcal{T} \), there exists its unfolding \( \hat{\mathcal{T}} \).

Proof idea.

The main reason is that terminologies have to be *cycle-free*. The proof can be done by induction of the *definition depth* of concepts.
Properties of Unfoldings (2): Equivalence

**Theorem (Model equivalence for unfolded terminologies)**

\( \mathcal{I} \) is a model of a normalized terminology \( \mathcal{T} \) iff it is a model of \( \hat{\mathcal{T}} \).

**Proof Sketch.**

\( \Rightarrow \) : Let \( \mathcal{I} \) be a model of \( \mathcal{T} \). Then it is also a model of \( U(\mathcal{T}) \), since on the right side of the definitions only terms with identical interpretations are substituted. However, then it must also be a model of \( \hat{\mathcal{T}} \).

\( \Leftarrow \) : Let \( \mathcal{I} \) be a model for \( U(\mathcal{T}) \). Clearly, this is also a model of \( \mathcal{T} \) (with the same argument as above). This means that any model \( \hat{\mathcal{T}} \) is also a model of \( \mathcal{T} \).
Properties of Unfoldings (2): Equivalence

**Theorem (Model equivalence for unfolded terminologies)**

\[ I \text{ is a model of a normalized terminology } \mathcal{T} \iff \text{ it is a model of } \hat{\mathcal{T}}. \]

**Proof Sketch.**

"⇒": Let \( I \) be a model of \( \mathcal{T} \). Then it is also a model of \( U(\mathcal{T}) \), since on the right side of the definitions only terms with identical interpretations are substituted. However, then it must also be a model of \( \hat{\mathcal{T}} \).

"⇐": Let \( I \) be a model for \( U(\mathcal{T}) \). Clearly, this is also a model of \( \mathcal{T} \) (with the same argument as above). This means that any model \( \hat{\mathcal{T}} \) is also a model of \( \mathcal{T} \).
Properties of Unfoldings (2): Equivalence

Theorem (Model equivalence for unfolded terminologies)

\( \mathcal{I} \) is a model of a normalized terminology \( T \) iff it is a model of \( \hat{T} \).

Proof Sketch.

“⇒”: Let \( \mathcal{I} \) be a model of \( T \). Then it is also a model of \( U(T) \), since on the right side of the definitions only terms with identical interpretations are substituted. However, then it must also be a model of \( \hat{T} \).

“⇐”: Let \( \mathcal{I} \) be a model for \( U(T) \). Clearly, this is also a model of \( T \) (with the same argument as above). This means that any model \( \hat{T} \) is also a model of \( T \).
Theorem (Model equivalence for unfolded terminologies)

$I$ is a model of a normalized terminology $\mathcal{T}$ iff it is a model of $\hat{\mathcal{T}}$.

Proof Sketch.

$⇒$: Let $I$ be a model of $\mathcal{T}$. Then it is also a model of $U(\mathcal{T})$, since on the right side of the definitions only terms with identical interpretations are substituted. However, then it must also be a model of $\hat{\mathcal{T}}$.

$⇐$: Let $I$ be a model for $U(\mathcal{T})$. Clearly, this is also a model of $\mathcal{T}$ (with the same argument as above). This means that any model $\hat{\mathcal{T}}$ is also a model of $\mathcal{T}$.
Properties of Unfoldings (2): Equivalence

Theorem (Model equivalence for unfolded terminologies)

\[ \mathcal{I} \text{ is a model of a normalized terminology } \mathcal{T} \text{ iff it is a model of } \hat{\mathcal{T}}. \]

Proof Sketch.

“⇒”: Let \( \mathcal{I} \) be a model of \( \mathcal{T} \). Then it is also a model of \( U(\mathcal{T}) \), since on the right side of the definitions only terms with identical interpretations are substituted. However, then it must also be a model of \( \hat{\mathcal{T}} \).

“⇐”: Let \( \mathcal{I} \) be a model for \( U(\mathcal{T}) \). Clearly, this is also a model of \( \mathcal{T} \) (with the same argument as above). This means that any model \( \hat{\mathcal{T}} \) is also a model of \( \mathcal{T} \).
Generating Models

- All concept and role names not appearing on the left hand side in a terminology $\mathcal{T}$ are called primitive components.
- Interpretations restricted to primitive components are called initial interpretations.

**Theorem (Model extension)**

For each initial interpretation $\mathcal{I}$ of a normalized TBox, there exists a unique interpretation $\mathcal{I}$ extending $\mathcal{I}$ and satisfying $\mathcal{T}$.

**Proof idea.**

Use $\hat{\mathcal{T}}$ and compute an interpretation for all defined symbols.

**Corollary (Model existence for TBoxes)**

Each TBox has at least one model.
Similar to the unfolding of TBoxes, we can define unfolding of concept descriptions.

We write $\hat{C}$ for the unfolded version of $C$.

**Theorem (Satisfiability of unfolded concepts)**

An concept description $C$ is satisfiable in a terminology $\mathcal{T}$ iff $\hat{C}$ satisfiable in an empty terminology.

**Proof.**

"$\Rightarrow$": trivial.

"$\Leftarrow$": Use the interpretation for all the symbols in $\hat{C}$ to generate an initial interpretation of $\mathcal{T}$. Then extend it to a full model $\mathcal{I}$ of $\mathcal{T}$. This satisfies $\mathcal{T}$ as well as $\hat{C}$. Since $\hat{C}^\mathcal{I} = C^\mathcal{I}$, it satisfies also $C$. 

\[ \square \]
Unfolding of Concept Descriptions

- Similar to the unfolding of TBoxes, we can define unfolding of concept descriptions.
- We write $\hat{C}$ for the unfolded version of $C$.

Theorem (Satisfiability of unfolded concepts)

An concept description $C$ is satisfiable in a terminology $T$ iff $\hat{C}$ satisfiable in an empty terminology.

Proof.

“$\Rightarrow$”: trivial.

“$\Leftarrow$”: Use the interpretation for all the symbols in $\hat{C}$ to generate an initial interpretation of $T$. Then extend it to a full model $I$ of $T$. This satisfies $T$ as well as $\hat{C}$. Since $\hat{C}^I = C^I$, it satisfies also $C$.\qed
Unfolding of Concept Descriptions

- Similar to the unfolding of TBoxes, we can define unfolding of concept descriptions.
- We write $\hat{C}$ for the unfolded version of $C$.

**Theorem (Satisfiability of unfolded concepts)**

An concept description $C$ is satisfiable in a terminology $T$ iff $\hat{C}$ satisfiable in an empty terminology.

**Proof.**

$\Rightarrow$: trivial.

$\Leftarrow$: Use the interpretation for all the symbols in $\hat{C}$ to generate an initial interpretation of $T$. Then extend it to a full model $I$ of $T$. This satisfies $T$ as well as $\hat{C}$. Since $\hat{C}^I = C^I$, it satisfies also $C$. □
Similar to the unfolding of TBoxes, we can define unfolding of concept descriptions.

We write $\hat{C}$ for the unfolded version of $C$.

**Theorem (Satisfiability of unfolded concepts)**

An concept description $C$ is satisfiable in a terminology $T$ iff $\hat{C}$ satisfiable in an empty terminology.

**Proof.**

"$\Rightarrow$": trivial.

"$\Leftarrow$": Use the interpretation for all the symbols in $\hat{C}$ to generate an initial interpretation of $T$. Then extend it to a full model $I$ of $T$. This satisfies $T$ as well as $\hat{C}$. Since $\hat{C}^I = C^I$, it satisfies also $C$. □
Motivation: Given a terminology $\mathcal{T}$ and two concept descriptions $C$ and $D$, is $C$ subsumed by (or a sub-concept of) $D$ in $\mathcal{T}$ ($C \sqsubseteq \mathcal{T} D$)?

Test:

- Is $C$ interpreted as a subset of $D$ for all models $\mathcal{I}$ of $\mathcal{T}$ ($C^\mathcal{I} \subseteq D^\mathcal{I}$)?
- Is the formula $\forall x : (C(x) \rightarrow D(x))$ a logical consequence of the translation of $\mathcal{T}$ to predicate logic?

Example: Grandmother $\sqsubseteq \mathcal{T}$ Mother
Motivation: Given a terminology $\mathcal{T}$ and two concept descriptions $C$ and $D$, is $C$ subsumed by (or a sub-concept of) $D$ in $\mathcal{T}$ ($C \sqsubseteq_\mathcal{T} D$)?

Test:
- Is $C$ interpreted as a subset of $D$ for all models $\mathcal{I}$ of $\mathcal{T}$ ($C^\mathcal{I} \subseteq D^\mathcal{I}$)?
- Is the formula $\forall x : (C(x) \rightarrow D(x))$ a logical consequence of the translation of $\mathcal{T}$ to predicate logic?

Example: Grandmother $\sqsubseteq_\mathcal{T}$ Mother
Subsumption (Without a TBox)

**Motivation:** Given two concept descriptions $C$ and $D$, is $C$ *subsumed by* $D$ regardless of a TBox (or in an *empty TBox*), written $C \sqsubseteq D$?

**Test:**
- Is $C$ interpreted as a subset of $D$ for *all interpretations* $\mathcal{I}$ ($C^\mathcal{I} \subseteq D^\mathcal{I}$)?
- Is the formula $\forall x : (C(x) \rightarrow D(x))$ *logically valid*?

**Example:** Human $\sqcap$ Female $\sqsubseteq$ Human
Motivation: Given two concept descriptions $C$ and $D$, is $C$ subsumed by $D$ regardless of a TBox (or in an empty TBox), written $C \sqsubseteq D$?

Test:

- Is $C$ interpreted as a subset of $D$ for all interpretations $I$ ($C^I \subseteq D^I$)?
- Is the formula $\forall x : (C(x) \rightarrow D(x))$ logically valid?

Example: Human $\sqcap$ Female $\sqsubseteq$ Human
Reductions

- Subsumption in a TBox can be reduced to subsumption in the empty TBox
  - Normalize and unfold TBox and concept descriptions.
- Subsumption in the empty TBox can be reduced to unsatisfiability
  - $C \sqsubseteq D$ iff $C \cap \neg D$ is unsatisfiable
- Unsatisfiability can be reduced to subsumption
  - $C$ is unsatisfiable iff $C \sqsubseteq (C \cap \neg C)$
Subsumption in a TBox can be reduced to subsumption in the empty TBox

Normalize and unfold TBox and concept descriptions.

Subsumption in the empty TBox can be reduced to unsatisfiability

\[ C \sqsubseteq D \text{ iff } C \cap \neg D \text{ is unsatisfiable} \]

Unsatisfiability can be reduced to subsumption

\[ C \text{ is unsatisfiable iff } C \sqsubseteq (C \cap \neg C) \]
Reductions

- Subsumption in a TBox can be reduced to subsumption in the empty TBox
- Normalize and unfold TBox and concept descriptions.
- Subsumption in the empty TBox can be reduced to unsatisfiability
  - \( C \subseteq D \) iff \( C \cap \neg D \) is unsatisfiable
  - Unsatisfiability can be reduced to subsumption
  - \( C \) is unsatisfiable iff \( C \subseteq (C \cap \neg C) \)
Subsumption in a TBox can be reduced to subsumption in the empty TBox

- **Normalize** and **unfold** TBox and concept descriptions.

Subsumption in the empty TBox can be reduced to unsatisfiability

- \( C \sqsubseteq D \) iff \( C \cap \neg D \) is unsatisfiable

Unsatisfiability can be reduced to subsumption

- \( C \) is unsatisfiable iff \( C \sqsubseteq (C \cap \neg C) \)
Reductions

- Subsumption in a TBox can be reduced to subsumption in the empty TBox
- *Normalize* and *unfold* TBox and concept descriptions.
- Subsumption in the empty TBox can be reduced to unsatisfiability
  
  \[ C \subseteq D \text{ iff } C \sqcap \neg D \text{ is unsatisfiable} \]

- Unsatisfiability can be reduced to subsumption
  
  \[ C \text{ is unsatisfiable iff } C \subseteq (C \sqcap \neg C) \]
Reductions

- Subsumption in a TBox can be reduced to subsumption in the empty TBox
- *Normalize* and *unfold* TBox and concept descriptions.
- Subsumption in the empty TBox can be reduced to unsatisfiability
  - \( C \sqsubseteq D \) iff \( C \sqcap \neg D \) is unsatisfiable
- Unsatisfiability can be reduced to subsumption
  - \( C \) is unsatisfiable iff \( C \sqsubseteq (C \sqcap \neg C) \)
Motivation: Compute all subsumption relationships (and represent them using only a minimal number of relationships) in order to

○ check the modeling – does the terminology make sense?
○ use the precomputed relations later when subsumption queries have to be answered
○ reduce to subsumption
○ it is a generalized sorting problem!
**Motivation**: Compute all subsumption relationships (and represent them using only a minimal number of relationships) in order to

- check the modeling – does the terminology make sense?
- use the precomputed relations later when subsumption queries have to be answered
- reduce to subsumption
- it is a generalized sorting problem!
**Motivation**: Compute all subsumption relationships (and represent them using only a minimal number of relationships) in order to

- check the modeling – does the terminology make sense?
- use the precomputed relations later when subsumption queries have to be answered
- reduce to subsumption
- it is a *generalized sorting* problem!
**Motivation:** Compute all subsumption relationships (and represent them using only a minimal number of relationships) in order to
- check the modeling – does the terminology make sense?
- use the precomputed relations later when subsumption queries have to be answered
- reduce to subsumption
- it is a *generalized sorting* problem!
**Motivation**: Compute all subsumption relationships (and represent them using only a minimal number of relationships) in order to

- check the modeling – does the terminology make sense?
- use the precomputed relations later when subsumption queries have to be answered
- reduce to subsumption
- it is a *generalized sorting* problem!
Motivation: Compute all subsumption relationships (and represent them using only a minimal number of relationships) in order to
- check the modeling – does the terminology make sense?
- use the precomputed relations later when subsumption queries have to be answered
- reduce to subsumption
- it is a generalized sorting problem!
**Motivation**: An ABox should *model* the real world, i.e., it should have a *model*.

**Test**: Check for a model

**Example**:

\[ X : (\forall r. \neg C) \]
\[ Y : C \]
\[ (X, Y) : r \]

is not satisfiable.
**Motivation:** An ABox should *model* the real world, i.e., it should have a *model*.

**Test:** Check for a model

**Example:**

\[ X : (\forall r. \neg C) \]
\[ Y : C \]
\[ (X, Y) : r \]

is not satisfiable.
Motivation: An ABox should model the real world, i.e., it should have a model.

Test: Check for a model

Example:

\[
\begin{align*}
X & : (\forall r. \neg C) \\
Y & : C \\
(X, Y) & : r
\end{align*}
\]

is not satisfiable.
**Motivation:** Is a given ABox $\mathcal{A}$ compatible with the terminology introduced in $\mathcal{T}$?

**Test:** Is $\mathcal{T} \cup \mathcal{A}$ satisfiable?

**Example:** If we extend our example with

\[
\text{MARGRET: Woman (DIANA, MARGRET): has-child,}
\]

then the ABox becomes unsatisfiable in the given TBox.

**Reduction:**

- to satisfiability of an ABox
- *Normalize* terminology, then *unfold* all concept and role descriptions in the ABox
ABox Satisfiability in a TBox

- **Motivation**: Is a given ABox $\mathcal{A}$ compatible with the terminology introduced in $\mathcal{T}$?
- **Test**: Is $\mathcal{T} \cup \mathcal{A}$ satisfiable?
- **Example**: If we extend our example with MARGRET: Woman
  (DIANA,MARGRET): has-child,
then the ABox becomes unsatisfiable in the given TBox.
- **Reduction**:
  - to satisfiability of an ABox
  - *Normalize* terminology, then *unfold* all concept and role descriptions in the ABox
**Motivation**: Is a given ABox $\mathcal{A}$ compatible with the terminology introduced in $\mathcal{T}$?

**Test**: Is $\mathcal{T} \cup \mathcal{A}$ satisfiable?

**Example**: If we extend our example with
- MARGRET: Woman
- (DIANA, MARGRET): has-child,

then the ABox becomes unsatisfiable in the given TBox.

**Reduction**:
- to satisfiability of an ABox
- *Normalize* terminology, then *unfold* all concept and role descriptions in the ABox
ABox Satisfiability in a TBox

- **Motivation**: Is a given ABox $\mathcal{A}$ compatible with the terminology introduced in $\mathcal{T}$?
- **Test**: Is $\mathcal{T} \cup \mathcal{A}$ satisfiable?
- **Example**: If we extend our example with
  MARGRET: Woman
  (DIANA,MARGRET): has-child,
then the ABox becomes unsatisfiable in the given TBox.
- **Reduction**:
  - to satisfiability of an ABox
  - *Normalize* terminology, then *unfold* all concept and role descriptions in the ABox
ABox Satisfiability in a TBox

- **Motivation**: Is a given ABox $\mathcal{A}$ compatible with the terminology introduced in $\mathcal{T}$?
- **Test**: Is $\mathcal{T} \cup \mathcal{A}$ satisfiable?
- **Example**: If we extend our example with
  MARGRET: Woman
  (DIANA,MARGRET): has-child,

  then the ABox becomes unsatisfiable in the given TBox.
- **Reduction**:
  - to satisfiability of an ABox
  - *Normalize* terminology, then *unfold* all concept and role descriptions in the ABox
**Motivation:** Which additional ABox formulas of the form $a : C$ follow logically from a given ABox and TBox?

**Test:**
- Is $a^I \in C^I$ true in all models of $I$ of $\mathcal{T} \cup \mathcal{A}$?
- Does the formula $C(a)$ logically follow from the translation of $\mathcal{A}$ and $\mathcal{T}$ to predicate logic?

**Reductions:**
- Instance relations wrt. an ABox and a TBox can be reduced to instance relations wrt. ABox.
- Use *normalization* and *unfolding*
- Instance relations in an ABox can be reduced to ABox unsatisfiability:

  $$a : C \text{ holds in } \mathcal{A} \text{ iff } \mathcal{A} \cup \{a : \neg C\} \text{ is unsatisfiable}$$
Instance Relations

**Motivation:** Which additional ABox formulas of the form \( a: C \) follow logically from a given ABox and TBox?

**Test:**
- Is \( a^I \in C^I \) true in all models of \( I \) of \( T \cup A \)?
- Does the formula \( C(a) \) logically follow from the translation of \( A \) and \( T \) to predicate logic?

**Reductions:**
- Instance relations wrt. an ABox and a TBox can be reduced to instance relations wrt. ABox.
- Use normalization and unfolding
- Instance relations in an ABox can be reduced to ABox unsatisfiability:

  \[ a: C \text{ holds in } A \text{ iff } A \cup \{a: \neg C\} \text{ is unsatisfiable} \]
Motivation: Which additional ABox formulas of the form $a: C$ follow logically from a given ABox and TBox?

Test:
- Is $a^\mathcal{I} \in C^\mathcal{I}$ true in all models of $\mathcal{I}$ of $\mathcal{T} \cup \mathcal{A}$?
- Does the formula $C(a)$ logically follow from the translation of $\mathcal{A}$ and $\mathcal{T}$ to predicate logic?

Reductions:
- Instance relations wrt. an ABox and a TBox can be reduced to instance relations wrt. ABox.
  - Use normalization and unfolding
- Instance relations in an ABox can be reduced to ABox unsatisfiability:
  $$a: C \text{ holds in } \mathcal{A} \text{ iff } \mathcal{A} \cup \{a: \neg C\} \text{ is unsatisfiable}$$
Motivation: Which additional ABox formulas of the form $a : C$ follow logically from a given ABox and TBox?

Test:
- Is $a^I \in C^I$ true in all models of $\mathcal{I}$ of $\mathcal{T} \cup \mathcal{A}$?
- Does the formula $C(a)$ logically follow from the translation of $\mathcal{A}$ and $\mathcal{T}$ to predicate logic?

Reductions:
- Instance relations wrt. an ABox and a TBox can be reduced to instance relations wrt. ABox.
  - Use normalization and unfolding
- Instance relations in an ABox can be reduced to ABox unsatisfiability:
  
  $$a : C \text{ holds in } \mathcal{A} \iff \mathcal{A} \cup \{a : \neg C\} \text{ is unsatisfiable}$$
Motivation: Which additional ABox formulas of the form \( a : C \) follow logically from a given ABox and TBox?

Test:
- Is \( a^I \in C^I \) true in all models of \( I \) of \( T \cup A \)?
- Does the formula \( C(a) \) logically follow from the translation of \( A \) and \( T \) to predicate logic?

Reductions:
- Instance relations wrt. an ABox and a TBox can be reduced to instance relations wrt. ABox.
- Use normalization and unfolding
- Instance relations in an ABox can be reduced to ABox unsatisfiability:

\[
a : C \text{ holds in } A \iff A \cup \{a : \neg C\} \text{ is unsatisfiable}
\]
Motivation: Which additional ABox formulas of the form \( a: C \) follow logically from a given ABox and TBox?

Test:
- Is \( a^\mathcal{I} \in C^\mathcal{I} \) true in all models of \( \mathcal{I} \) of \( T \cup A \)?
- Does the formula \( C(a) \) logically follow from the translation of \( A \) and \( T \) to predicate logic?

Reductions:
- Instance relations wrt. an ABox and a TBox can be reduced to instance relations wrt. ABox.
  - Use normalization and unfolding
- Instance relations in an ABox can be reduced to ABox unsatisfiability:
  \[
a: C \text{ holds in } A \iff A \cup \{a: \neg C\} \text{ is unsatisfiable}
\]
Examples

• ELIZABETH: Mother-with-many-children?

• WILLIAM: ­ Female?

• ELIZABETH: Mother-without-daughter?

• ELIZABETH: Grandmother?
Examples

- ELIZABETH: Mother-with-many-children?
  - yes
- WILLIAM: \( \neg \) Female?
- ELIZABETH: Mother-without-daughter?
- ELIZABETH: Grandmother?
Examples

- ELIZABETH: Mother-with-many-children?
  - yes
- WILLIAM: $\neg$ Female?
- ELIZABETH: Mother-without-daughter?
- ELIZABETH: Grandmother?
Examples

- ELIZABETH: Mother-with-many-children?
  - yes
- WILLIAM: $\neg$ Female?
  - yes
- ELIZABETH: Mother-without-daughter?
- ELIZABETH: Grandmother?
Examples

- ELIZABETH: Mother-with-many-children?
  - yes
- WILLIAM: ¬ Female?
  - yes
- ELIZABETH: Mother-without-daughter?
- ELIZABETH: Grandmother?
Examples

- ELIZABETH: Mother-with-many-children?
  - yes
- WILLIAM: ¬ Female?
  - yes
- ELIZABETH: Mother-without-daughter?
  - no (no CWA!)
- ELIZABETH: Grandmother?
Examples

- ELIZABETH: Mother-with-many-children?
  - yes
- WILLIAM: \( \neg \) Female?
  - yes
- ELIZABETH: Mother-without-daughter?
  - no (no CWA!)
- ELIZABETH: Grandmother?
Examples

- **ELIZABETH:** Mother-with-many-children?
  - **yes**

- **WILLIAM:** ¬ Female?
  - **yes**

- **ELIZABETH:** Mother-without-daughter?
  - **no** (no CWA!)

- **ELIZABETH:** Grandmother?
  - **no** (only male, but not necessarily human!)
Realization

- **Idea**: For a given object $a$, determine the most specialized concept symbols such that $a$ is an instance of these concepts.

- **Motivation**:
  - Similar to *classification*
  - Is the minimal representation of the instance relations (in the set of concept symbols)
  - Will give us faster answers for instance queries!

- **Reduction**: Can be reduced to (a sequence of) instance relation tests.
Realization

- **Idea**: For a given object $a$, determine the most specialized concept symbols such that $a$ is an instance of these concepts.

- **Motivation**:
  - Similar to *classification*
  - Is the minimal representation of the instance relations (in the set of concept symbols)
  - Will give us faster answers for instance queries!

- **Reduction**: Can be reduced to (a sequence of) instance relation tests.
Realization

**Idea:** For a given object $a$, determine the most specialized concept symbols such that $a$ is an instance of these concepts.

**Motivation:**
- Similar to *classification*
- Is the minimal representation of the instance relations (in the set of concept symbols)
- Will give us faster answers for instance queries!

**Reduction:** Can be reduced to (a sequence of) instance relation tests.
Realization

- **Idea:** For a given object $a$, determine the most specialized concept symbols such that $a$ is an instance of these concepts.

- **Motivation:**
  - Similar to *classification*
  - Is the minimal representation of the instance relations (in the set of concept symbols)
  - Will give us faster answers for instance queries!

- **Reduction:** Can be reduced to (a sequence of) instance relation tests.
Idea: For a given object $a$, determine the most specialized concept symbols such that $a$ is an instance of these concepts.

Motivation:
- Similar to classification
- Is the minimal representation of the instance relations (in the set of concept symbols)
- Will give us faster answers for instance queries!

Reduction: Can be reduced to (a sequence of) instance relation tests.
**Motivation**: Sometimes, we want to get the set of instances of a concept (as in database queries).

**Example**: Asking for all instances of the concept Male, we will get the answer CHARLES, ANDREW, EDWARD, WILLIAM.

**Reduction**: Compute the set of instances by testing the instance relation for each object.

**Implementation**: Realization can be used to speed this up.
Retrieval

- **Motivation**: Sometimes, we want to get the set of instances of a concept (as in database queries)
- **Example**: Asking for all instances of the concept Male, we will get the answer CHARLES, ANDREW, EDWARD, WILLIAM.
- **Reduction**: Compute the set of instances by testing the instance relation for each object
- **Implementation**: Realization can be used to speed this up
**Motivation:** Sometimes, we want to get the set of instances of a concept (as in database queries)

**Example:** Asking for all instances of the concept Male, we will get the answer CHARLES, ANDREW, EDWARD, WILLIAM.

**Reduction:** Compute the set of instances by testing the instance relation for each object

**Implementation:** Realization can be used to speed this up
Motivation: Sometimes, we want to get the set of instances of a concept (as in database queries)

Example: Asking for all instances of the concept Male, we will get the answer CHARLES, ANDREW, EDWARD, WILLIAM.

Reduction: Compute the set of instances by testing the instance relation for each object

Implementation: Realization can be used to speed this up
Reasoning Services – Summary

- Satisfiability of concept descriptions
  - in a given TBox or in an empty TBox
- Subsumption between concept descriptions
  - in a given TBox or in an empty TBox
- Classification
- Satisfiability of an ABox
  - in a given TBox or in an empty TBox
- Instance relations in an ABox
  - in a given TBox or in an empty TBox
- Realization
- Retrieval
Reasoning Services – Summary

- Satisfiability of concept descriptions
  - in a given TBox or in an empty TBox
- Subsumption between concept descriptions
  - in a given TBox or in an empty TBox
- Classification
- Satisfiability of an ABox
  - in a given TBox or in an empty TBox
- Instance relations in an ABox
  - in a given TBox or in an empty TBox
- Realization
- Retrieval
Reasoning Services – Summary

- Satisfiability of concept descriptions
  - in a given TBox or in an empty TBox
- Subsumption between concept descriptions
  - in a given TBox or in an empty TBox
- Classification
  - Satisfiability of an ABox
    - in a given TBox or in an empty TBox
  - Instance relations in an ABox
    - in a given TBox or in an empty TBox
- Realization
- Retrieval
Reasoning Services – Summary

- Satisfiability of concept descriptions
  - in a given TBox or in an empty TBox
- Subsumption between concept descriptions
  - in a given TBox or in an empty TBox
- Classification
- Satisfiability of an ABox
  - in a given TBox or in an empty TBox
- Instance relations in an ABox
  - in a given TBox or in an empty TBox
- Realization
- Retrieval
Reasoning Services – Summary

- Satisfiability of concept descriptions
  - in a given TBox or in an empty TBox
- Subsumption between concept descriptions
  - in a given TBox or in an empty TBox
- Classification
- Satisfiability of an ABox
  - in a given TBox or in an empty TBox
- Instance relations in an ABox
  - in a given TBox or in an empty TBox
- Realization
- Retrieval
Reasoning Services – Summary

- Satisfiability of concept descriptions
  - in a given TBox or in an empty TBox
- Subsumption between concept descriptions
  - in a given TBox or in an empty TBox
- Classification
- Satisfiability of an ABox
  - in a given TBox or in an empty TBox
- Instance relations in an ABox
  - in a given TBox or in an empty TBox
- Realization
  - Retrieval
### Reasoning Services – Summary

- **Satisfiability of concept descriptions**
  - in a given TBox or in an empty TBox
- **Subsumption between concept descriptions**
  - in a given TBox or in an empty TBox
- **Classification**
- **Satisfiability of an ABox**
  - in a given TBox or in an empty TBox
- **Instance relations in an ABox**
  - in a given TBox or in an empty TBox
- **Realization**
- **Retrieval**
Outlook

- How to determine *subsumption* between two concept description (in the empty TBox)?
- How to determine *instance relations/ABox satisfiability*?
- How to implement the mentioned reductions *efficiently*?
- Does normalization and unfolding introduce another source of *computational complexity*?
How to determine *subsumption* between two concept description (in the empty TBox)?

How to determine *instance relations/ABox satisfiability*?

How to implement the mentioned reductions *efficiently*?

Does normalization and unfolding introduce another source of *computational complexity*?
Outlook

- How to determine *subsumption* between two concept description (in the empty TBox)?
- How to determine *instance relations/ABox satisfiability*?
- How to implement the mentioned reductions *efficiently*?
- Does normalization and unfolding introduce another source of *computational complexity*?
Outlook

- How to determine *subsumption* between two concept description (in the empty TBox)?
- How to determine *instance relations/ABox satisfiability*?
- How to implement the mentioned reductions *efficiently*?
- Does normalization and unfolding introduce another source of *computational complexity*?