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

Semantic Networks and Description Logics III: Description Logics – Reasoning Services and Reductions

Example TBox & ABox

$$
\begin{aligned}
\texttt{Male} &\doteq \neg\texttt{Female}\\
\texttt{Human} &\sqsubseteq \texttt{Living\_entity}\\
\texttt{Woman} &\doteq \texttt{Human} \sqcap \texttt{Female}\\
\texttt{Man} &\doteq \texttt{Human} \sqcap \texttt{Male}\\
\texttt{Mother} &\doteq \texttt{Woman} \sqcap \exists\texttt{has-child.Human}\\
\texttt{Father} &\doteq \texttt{Man} \sqcap \exists\texttt{has-child.Human}\\
\texttt{Parent} &\doteq \texttt{Father} \sqcup \texttt{Mother}\\
\texttt{Grandmother} &\\
&\doteq \texttt{Woman} \sqcap \exists\texttt{has-child.Parent}\\
\texttt{Mother-without-daughter} &\\
&\doteq \texttt{Mother} \sqcap \forall\texttt{has-child.Male}\\
\texttt{Mother-with-many-children} &\\
&\doteq \texttt{Mother} \sqcap (\geq 3\, \texttt{has-child})
\end{aligned}
$$

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 $\exists x\colon C(x)$ 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 (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 $\exists x \colon C(x)$ satisfiable?
- Example: $\texttt{Woman} \sqcap (\leq 0\,\texttt{has-child}) \sqcap (\geq 1\,\texttt{has-child})$ 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* $\mathcal{I}$ such that $C^{\mathcal{I}} \neq \emptyset$?
  - Is the formula $\exists x : C(x)$ satisfiable?
- Example: $\texttt{Woman} \sqcap (\leq 0\, \texttt{has-child}) \sqcap (\geq 1\, \texttt{has-child})$ 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 $\sqsubseteq$)?

Normalized Terminologies

- A terminology is called normalized when it does not contain definitions using $\sqsubseteq$.
- In order to *normalize* a terminology, replace

$$A \sqsubseteq C$$

by

$$A \doteq A^* \sqcap C,$$

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 $\widetilde{\mathcal{T}}$.

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 $\widetilde{\mathcal{T}}$ (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 $(A \doteq A^* \sqcap C) \in \widetilde{\mathcal{T}}$. Then set $A^{*\mathcal{I}'} = A^{\mathcal{I}}$. $\mathcal{I}'$ obviously satisfies $\widetilde{\mathcal{T}}$ and has the same interpretation for all symbols in $\mathcal{T}$.
$\Leftarrow$ Given a model $\mathcal{I}'$ of $\widetilde{\mathcal{T}}$, 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 $\widetilde{\mathcal{T}}$ (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 $(A \doteq A^* \sqcap C) \in \widetilde{\mathcal{T}}$. Then set ${A^*}^{\mathcal{I}'} = A^{\mathcal{I}}$. $\mathcal{I}'$ obviously satisfies $\widetilde{\mathcal{T}}$ and has the same interpretation for all symbols in $\mathcal{T}$.

$\Leftarrow$ Given a model $\mathcal{I}'$ of $\widetilde{\mathcal{T}}$, 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 $\widetilde{\mathcal{T}}$ (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 $(A \doteq A^* \sqcap C) \in \widetilde{\mathcal{T}}$. Then set $A^{*\mathcal{I}'} = A^{\mathcal{I}}$. $\mathcal{I}'$ obviously satisfies $\widetilde{\mathcal{T}}$ and has the same interpretation for all symbols in $\mathcal{T}$.

$\Leftarrow$ Given a model $\mathcal{I}'$ of $\widetilde{\mathcal{T}}$, its restriction to symbols of $\mathcal{T}$ is the interpretation we looked for. $\square$

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 $\widetilde{\mathcal{T}}$ (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 $(A \doteq A^* \sqcap C) \in \widetilde{\mathcal{T}}$. Then set $A^{*\mathcal{I}'} = A^{\mathcal{I}}$. $\mathcal{I}'$ obviously satisfies $\widetilde{\mathcal{T}}$ and has the same interpretation for all symbols in $\mathcal{T}$.

$\Leftarrow$ Given a model $\mathcal{I}'$ of $\widetilde{\mathcal{T}}$, its restriction to symbols of $\mathcal{T}$ is the interpretation we looked for. $\square$

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 $\widetilde{\mathcal{T}}$ (and vice versa*) such that for all concept symbols A appearing in $\mathcal{T}$ we have:*

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

Proof.

"⇒": 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 $(A \doteq A^* \sqcap C) \in \widetilde{\mathcal{T}}$. Then set $A^{*\mathcal{I}'} = A^{\mathcal{I}}$. $\mathcal{I}'$ obviously satisfies $\widetilde{\mathcal{T}}$ and has the same interpretation for all symbols in $\mathcal{T}$.
⇐ Given a model $\mathcal{I}'$ of $\widetilde{\mathcal{T}}$, its restriction to symbols of $\mathcal{T}$ is the interpretation we looked for. □

TBox Unfolding

- 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: $\texttt{Mother} \doteq \texttt{Woman} \sqcap \ldots$ is unfolded to $\texttt{Mother} \doteq (\texttt{Human} \sqcap \texttt{Female}) \sqcap \ldots$
- 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 $\widehat{\mathcal{T}}$

TBox Unfolding

- 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: $\texttt{Mother} \doteq \texttt{Woman} \sqcap \ldots$ is unfolded to $\texttt{Mother} \doteq (\texttt{Human} \sqcap \texttt{Female}) \sqcap \ldots$
- 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 $\widehat{\mathcal{T}}$

Properties of Unfoldings (1): Existence

Theorem (Existence of unfolded terminology)

For each normalized terminology $\mathcal{T}$, there exists its unfolding $\widehat{\mathcal{T}}$.

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 $\widehat{\mathcal{T}}$.

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 $\widehat{\mathcal{T}}$.

"$\Leftarrow$": 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 $\widehat{\mathcal{T}}$ 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 $\mathcal{T}$ iff it is a model of $\widehat{\mathcal{T}}$.

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 $\widehat{\mathcal{T}}$.

"$\Leftarrow$": 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 $\widehat{\mathcal{T}}$ 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 $\mathcal{T}$ iff it is a model of $\widehat{\mathcal{T}}$.

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 $\widehat{\mathcal{T}}$.

"$\Leftarrow$": 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 $\widehat{\mathcal{T}}$ 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 $\mathcal{T}$ iff it is a model of $\widehat{\mathcal{T}}$.

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 $\widehat{\mathcal{T}}$.

"$\Leftarrow$": 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 $\widehat{\mathcal{T}}$ 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 $\mathcal{T}$ iff it is a model of $\widehat{\mathcal{T}}$.

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 $\widehat{\mathcal{T}}$.

"$\Leftarrow$": 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 $\widehat{\mathcal{T}}$ 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 $\mathcal{J}$ of a normalized TBox, there exists a unique interpretation $\mathcal{I}$ extending $\mathcal{J}$ and satisfying $\mathcal{T}$.

Proof idea.

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

Corollary (Model existence for TBoxes)

Each TBox has at least one model.

Unfolding of Concept Descriptions

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

Theorem (Satisfiability of unfolded concepts)

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

Proof.

"$\Rightarrow$": trivial.

"$\Leftarrow$": Use the interpretation for all the symbols in $\widehat{C}$ 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 $\widehat{C}$. Since $\widehat{C}^{\mathcal{I}} = C^{\mathcal{I}}$, it satisfies also C. □

Unfolding of Concept Descriptions

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

Theorem (Satisfiability of unfolded concepts)

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

Proof.

"$\Rightarrow$": trivial.
"$\Leftarrow$": Use the interpretation for all the symbols in $\widehat{C}$ 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 $\widehat{C}$. Since $\widehat{C}^{\mathcal{I}} = C^{\mathcal{I}}$, it satisfies also C. $\square$

Unfolding of Concept Descriptions

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

Theorem (Satisfiability of unfolded concepts)

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

Proof.

"$\Rightarrow$": trivial.
"$\Leftarrow$": Use the interpretation for all the symbols in $\widehat{C}$ 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 $\widehat{C}$. Since $\widehat{C}^{\mathcal{I}} = C^{\mathcal{I}}$, it satisfies also C. $\square$

Unfolding of Concept Descriptions

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

Theorem (Satisfiability of unfolded concepts)

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

Proof.

"$\Rightarrow$": trivial.
"$\Leftarrow$": Use the interpretation for all the symbols in $\widehat{C}$ 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 $\widehat{C}$. Since $\widehat{C}^{\mathcal{I}} = C^{\mathcal{I}}$, it satisfies also C. □

Subsumption in a TBox

- 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 $\forall x : \big(C(x) \rightarrow D(x)\big)$ a logical consequence of the translation of $\mathcal{T}$ to predicate logic?
- Example: `Grandmother` $\sqsubseteq_{\mathcal{T}}$ `Mother`

Subsumption in a TBox

- 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 $\forall x : \big(C(x) \rightarrow D(x)\big)$ a logical consequence of the translation of $\mathcal{T}$ to predicate logic?
- Example: $\texttt{Grandmother} \sqsubseteq_{\mathcal{T}} \texttt{Mother}$

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 $\forall x : \big(C(x) \rightarrow D(x)\big)$ *logically valid*?
- Example: $\texttt{Human} \sqcap \texttt{Female} \sqsubseteq \texttt{Human}$

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 $\forall x : \big(C(x) \rightarrow D(x)\big)$ *logically valid*?
- Example: $\texttt{Human} \sqcap \texttt{Female} \sqsubseteq \texttt{Human}$

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 $C \sqcap \neg D$ is unsatisfiable
- Unsatisfiability can be reduced to subsumption
- C is unsatisfiable iff $C \sqsubseteq (C \sqcap \neg C)$

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 $C \sqsubseteq (C \sqcap \neg C)$

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 $C \sqsubseteq (C \sqcap \neg C)$

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 $C \sqsubseteq (C \sqcap \neg C)$

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 $C \sqsubseteq (C \sqcap \neg C)$

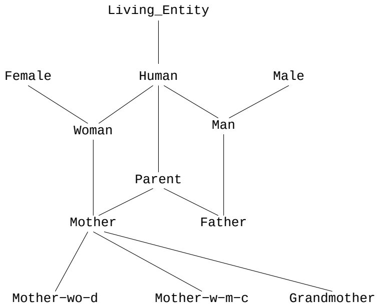
Reductions

- 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 $C \sqsubseteq (C \sqcap \neg C)$

Classification

- 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!

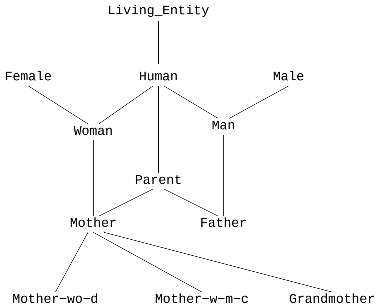
Example



Classification

- 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!

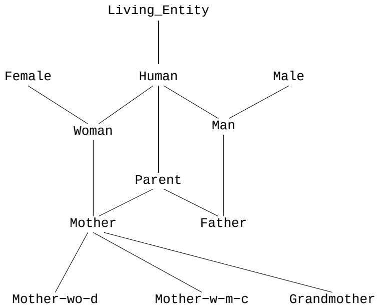
Example



Classification

- 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!

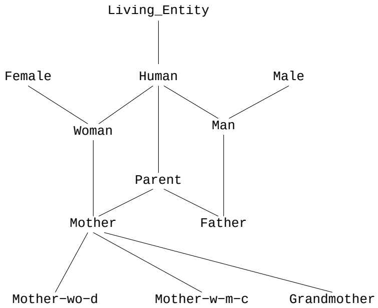
Example



Classification

- 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!

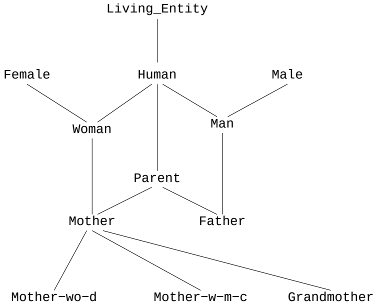
Example



Classification

- 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!

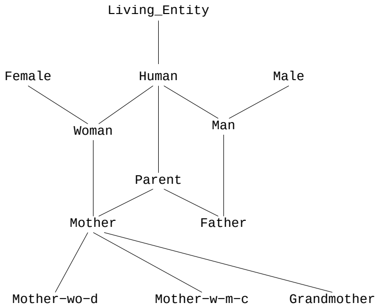
Example



Classification

- 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!

Example



ABox Satisfiability

- Motivation: An ABox should *model* the real world, i.e., it should have a model.
- Test: Check for a model
- Example:

$$
\begin{aligned}
X &: (\forall r.\neg C) \\
Y &: C \\
(X, Y) &: r
\end{aligned}
$$

is not satisfiable.

ABox Satisfiability

- Motivation: An ABox should *model* the real world, i.e., it should have a model.
- Test: Check for a model
- Example:

$$\begin{aligned} X &: (\forall r.\neg C) \\ Y &: C \\ (X, Y) &: r \end{aligned}$$

is not satisfiable.

ABox Satisfiability

- Motivation: An ABox should *model* the real world, i.e., it should have a model.
- Test: Check for a model
- Example:

$$\begin{array}{rcl} X & : & (\forall r.\neg C) \\ Y & : & C \\ (X, Y) & : & r \end{array}$$

is not satisfiable.

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

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

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

Instance Relations

- Motivation: Which additional ABox formulas of the form $a\colon C$ 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:

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

Instance Relations

- Motivation: Which additional ABox formulas of the form $a\colon C$ 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:

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

Instance Relations

- Motivation: Which additional ABox formulas of the form $a\colon C$ 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:

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

Instance Relations

- Motivation: Which additional ABox formulas of the form $a\colon C$ 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:

$$a\colon C \text{ holds in } \mathcal{A} \quad \text{iff} \quad \mathcal{A} \cup \{a\colon \neg C\} \text{ is unsatisfiable}$$

Instance Relations

- Motivation: Which additional ABox formulas of the form $a\colon C$ 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:

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

Instance Relations

- Motivation: Which additional ABox formulas of the form $a\colon C$ 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:

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

Examples

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

Examples

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

Examples

- ELIZABETH: Mother-with-many-children?
- yes
- WILLIAM: ¬ 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: ¬ 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.

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.

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

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

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

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

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*?

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*?

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*?