Knowledge Representation for the Semantic Web

Winter Quarter 2010

Slides 9 – 03/01/2010

Pascal Hitzler
Kno.e.sis Center
Wright State University, Dayton, OH
http://www.knoesis.org/pascal/
Slides are based on

Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph

Foundations of Semantic Web Technologies

Chapman & Hall/CRC, 2010

Flyer with special offer is available.

http://www.semantic-web-book.org
Today: Reasoning with OWL
A is a logical consequence of $K$
written $K \models A$
if and only if
every model of $K$ is a model of $A$.

• To show an entailment, we need to check all models?
• But that’s infinitely many!!!
A Reasoning Problem

We need algorithms which do not apply the model-based definition of logical consequence in a naive manner.

These algorithms should be syntax-based. (Computers can only do syntax manipulations.)

Luckily, such algorithms exist!

However, their correctness (soundness and completeness) needs to be proven formally. Which is often a non-trivial problem requiring substantial mathematical build-up.

We won't do the proofs here.
Contents

- Important inference problems
- Tableaux algorithm for ALC
- Tableaux algorithm for SHIQ
Important Inference Problems

- Global consistency of a knowledge base. $\text{KB \models false}$?
  - Is the knowledge base meaningful?
- Class consistency $\equiv \bot$?
  - Is C necessarily empty?
- Class inclusion (Subsumption) $C \subseteq D$?
  - Structuring knowledge bases
- Class equivalence $C \equiv D$?
  - Are two classes in fact the same class?
- Class disjointness $C \cap D = \bot$?
  - Do they have common members?
- Class membership $C(a)$?
  - Is a contained in C?
- Instance Retrieval „find all x with C(x)”
  - Find all (known!) individuals belonging to a given class.
Reduction to Unsatisfiability

- Global consistency of a knowledge base. \( \text{KB unsatisfiable} \)
  - Failure to find a model.
- Class consistency \( C \equiv \bot ? \)
  - \( \text{KB} \cup \{C(a)\} \) unsatisfiable
- Class inclusion (Subsumption) \( C \subseteq D ? \)
  - \( \text{KB} \cup \{C \cap \neg D(a)\} \) unsatisfiable (a new)
- Class equivalence \( C \equiv D ? \)
  - \( C \subseteq D \text{ and } D \subseteq C \)
- Class disjointness \( C \cap D = \bot ? \)
  - \( \text{KB} \cup \{(C \cap D)(a)\} \) unsatisfiable (a new)
- Class membership \( C(a) ? \)
  - \( \text{KB} \cup \{\neg C(a)\} \) unsatisfiable
- Instance Retrieval „find all x with C(x)“
  - Check class membership for all individuals.
We will present so-called tableaux algorithms.

They attempt to construct a model of the knowledge base in a „general, abstract“ manner.
- If the construction fails, then (provably) there is no model – i.e. the knowledge base is unsatisfiable.
- If the construction works, then it is satisfiable.

Hence the reduction of all inference problems to the checking of unsatisfiability of the knowledge base!
Contents

• Important inference problems
• Tableaux algorithm for ALC
• Tableaux algorithm for SHIQ
ALC tableaux: contents

- Transformation to negation normal form
- Naive tableaux algorithm
- Tableaux algorithm with blocking
Given a knowledge base K.

- Replace $C \equiv D$ by $C \subseteq D$ and $D \subseteq C$.
- Replace $C \subseteq D$ by $\neg C \sqcup D$.
- Apply the equations on the next slide exhaustively.

Resulting knowledge base: $\text{NNF}(K)$

*Negation normal form of K.*

Negation occurs only directly in front of atomic classes.
\[
\begin{align*}
\text{NNF}(C') &= C \quad \text{if } C \text{ is a class name} \\
\text{NNF}(\neg C') &= \neg C \quad \text{if } C \text{ is a class name} \\
\text{NNF}(\neg\neg C') &= \text{NNF}(C') \\
\text{NNF}(C \sqcup D) &= \text{NNF}(C') \sqcup \text{NNF}(D) \\
\text{NNF}(C \sqcap D) &= \text{NNF}(C') \sqcap \text{NNF}(D) \\
\text{NNF}(\neg(C \sqcup D)) &= \text{NNF}(\neg C') \sqcap \text{NNF}(\neg D) \\
\text{NNF}(\neg(C \sqcap D)) &= \text{NNF}(\neg C') \sqcup \text{NNF}(\neg D) \\
\text{NNF}(\forall R.C') &= \forall R.\text{NNF}(C') \\
\text{NNF}(\exists R.C') &= \exists R.\text{NNF}(C') \\
\text{NNF}(\neg\forall R.C') &= \exists R.\text{NNF}(\neg C') \\
\text{NNF}(\neg\exists R.C') &= \forall R.\text{NNF}(\neg C')
\end{align*}
\]

K and NNF(K) have the same models (are \textit{logically equivalent}).
Example

\[ P \subseteq (E \cap U) \cup \neg(\neg E \cup D). \]

In negation normal form:

\[ \neg P \cup (E \cap U) \cup (E \cap \neg D). \]
ALC tableaux: contents

• Transformation to negation normal form
• Naive tableaux algorithm
• Tableaux algorithm with blocking
Reduction to (un)satisfiability.

Idea:

• Given knowledge base $K$
• Attempt construction of a tree (called *Tableau*), which represents a model of $K$.
  (It’s actually rather a *Forest*.)
• If attempt fails, $K$ is unsatisfiable.
The Tableau

- Nodes represent elements of the domain of the model
  - Every node $x$ is labeled with a set $L(x)$ of class expressions. $C \in L(x)$ means: "$x$ is in the extension of $C$"

- Edges stand for role relationships:
  - Every edge $<x,y>$ is labeled with a set $L(<x,y>)$ of role names. $R \in L(<x,y>)$ means: "$(x,y)$ is in the extension of $R$"
Simple example

C(a)
C ⊆ ∃R.D
D ⊆ E

Does this entail (∃R.E)(a)?

(add ∀R.¬E(a) and show unsatisfiability)
Another example

C(a)
C ⊆ ∃R.D
D ⊆ E ∪ F
F ⊆ E

Does this entail (∃R.E)(a)?

(add ∀R.¬E(a)
and show unsatisfiability)

C
∃R.D
∀R.¬E

a
R

D
¬E (because ∀R.¬E(a))
choice: (D ⊆ E ∪ F):
1. E (contradiction!)
2. F
   E (contradiction!)
Formal Definition

- Input: K=TBox + ABox (in NNF)
- Output: Whether or not K is satisfiable.

- A tableau is a directed labeled graph
  - nodes are individuals or (new) variable names
  - nodes x are labeled with sets L(x) of classes
  - edges <x,y> are labeled with sets L(<x,y>) of role names
Initialisation

- Make a node for every individual in the ABox.
- Every node is labeled with the corresponding class names from the ABox.
- There is an edge, labeled with R, between a and b, if R(a,b) is in the ABox.

- (If there is no ABox, the initial tableau consists of a node x with empty label.)
Example initialisation

\[
\begin{align*}
\text{Human} & \subseteq \exists \text{hasParent.Human} \\
\text{Orphan} & \subseteq \text{Human} \cap \neg \exists \text{hasParent.Alive} \\
\text{Orphan(harrypotter)} & \\
\text{hasParent(harrypotter, jamespotter)} &
\end{align*}
\]
Careful: need NNF!

\[ \neg \text{Human} \lor \exists \text{hasParent.Human} \]
\[ \neg \text{Orphan} \lor (\text{Human} \land \forall \text{hasParent.} \neg \text{Alive}) \]

Orphan(harrypotter)
hasParent(harrypotter,jamespotter)
Constructing the tableau

- Non-deterministically extend the tableau using the rules on the next slide.

- Terminate, if
  - there is a contradiction in a node label (i.e., it contains classes C and \( \neg C \), or it contains \( \bot \)), or
  - none of the rules is applicable.

- If the tableau does not contain a contradiction, then the knowledge base is satisfiable.
  Or more precisely: If you can make a choice of rule applications such that no contradiction occurs and the process terminates, then the knowledge base is satisfiable.
Naive ALC tableaux rules

\(\square\)-rule: If \(C \cap D \in \mathcal{L}(x)\) and \(\{C, D\} \not\subseteq \mathcal{L}(x)\), then set \(\mathcal{L}(x) \leftarrow \{C, D\}\).

\(\square\)-rule: If \(C \cup D \in \mathcal{L}(x)\) and \(\{C, D\} \cap \mathcal{L}(x) = \emptyset\), then set \(\mathcal{L}(x) \leftarrow C\) or \(\mathcal{L}(x) \leftarrow D\).

\(\exists\)-rule: If \(\exists R.C \in \mathcal{L}(x)\) and there is no \(y\) with \(R \in L(x, y)\) and \(C \in \mathcal{L}(y)\), then

1. add a new node with label \(y\) (where \(y\) is a new node label),
2. set \(\mathcal{L}(x, y) = \{R\}\), and
3. set \(\mathcal{L}(y) = \{C\}\).

\(\forall\)-rule: If \(\forall R.C \in \mathcal{L}(x)\) and there is a node \(y\) with \(R \in L(x, y)\) and \(C \notin \mathcal{L}(y)\), then set \(\mathcal{L}(y) \leftarrow C\).

TBox-rule: If \(C\) is a TBox statement and \(C \notin \mathcal{L}(x)\), then set \(\mathcal{L}(x) \leftarrow C\).
Example

\neg \text{Human} \cup \exists \text{hasParent. Human}

\neg \text{Orphan} \cup (\text{Human} \cap \forall \text{hasParent. \neg Alive})

\text{Orphan(harrypotter)}

\text{hasParent(harrypotter, jamespotter)}

\neg \text{Alive(jamespotter)}

i.e. add: \text{Alive(jamespotter)}

and search for contradiction
ALC tableaux: contents

- Transformation to negation normal form
- Naive tableaux algorithm
- Tableaux algorithm with blocking
There’s a termination problem

TBox: $\exists R. T$

ABox: $T(a_1)$

- Obviously satisfiable:
  Model $M$ with domain elements $a_1^M, a_2^M, ...$
  and $R^M(a_i^M, a_{i+1}^M)$ for all $i \geq 1$

- but tableaux algorithm does not terminate!
Solution?

Actually, things repeat!
Idea: it is not necessary to expand $x$, since it’s simply a copy of $a$.

⇒ Blocking
Blocking

- $x$ is *blocked* (by $y$) if
  - $x$ is not an individual (but a variable)
  - $y$ is a predecessor of $x$ and $L(x) \subseteq L(y)$
  - or a predecessor of $x$ is blocked

Here, $x$ is blocked by $a$. 
Constructing the tableau

- Non-deterministically extend the tableau using the rules on the next slide, **but only apply a rule if x is not blocked!**

- **Terminate, if**
  - there is a contradiction in a node label (i.e., it contains classes C and \( \neg C \)), or
  - none of the rules is applicable.

- If the tableau does not contain a contradiction, then the knowledge base is satisfiable.
  Or more precisely: If you can make a choice of rule applications such that no contradiction occurs and the process terminates, then the knowledge base is satisfiable.
Naive ALC tableaux rules

\( \cap\)-rule: If \( C \cap D \in \mathcal{L}(x) \) and \( \{C, D\} \not\subseteq \mathcal{L}(x) \), then set \( \mathcal{L}(x) \leftarrow \{C, D\} \).

\( \sqcap\)-rule: If \( C \sqcap D \in \mathcal{L}(x) \) and \( \{C, D\} \cap \mathcal{L}(x) = \emptyset \), then set \( \mathcal{L}(x) \leftarrow C \) or \( \mathcal{L}(x) \leftarrow D \).

\( \exists\)-rule: If \( \exists R.C \in \mathcal{L}(x) \) and there is no \( y \) with \( R \in \mathcal{L}(x, y) \) and \( C \in \mathcal{L}(y) \), then

1. add a new node with label \( y \) (where \( y \) is a new node label),
2. set \( \mathcal{L}(x, y) = \{R\} \), and
3. set \( \mathcal{L}(y) = \{C\} \).

\( \forall\)-rule: If \( \forall R.C \in \mathcal{L}(x) \) and there is a node \( y \) with \( R \in \mathcal{L}(x, y) \) and \( C \not\in \mathcal{L}(y) \), then set \( \mathcal{L}(y) \leftarrow C \).

TBox-rule: If \( C \) is a TBox statement and \( C \not\in \mathcal{L}(x) \), then set \( \mathcal{L}(x) \leftarrow C \).

Apply only if \( x \) is not blocked!
Example (0)

- Knowledge base \{Human \sqsubseteq \exists \text{hasParent}\cdot \text{Human}, \text{Bird(tweety)}\}
- We want to show that Human(tweety) does \textit{not} hold, i.e. that \neg \text{Human(tweety)} is entailed.
- We will not be able to show this. I.e. Human(tweety) is \textit{possible}.

- Shorter notation:
  \begin{align*}
  &H \sqsubseteq \exists p.H \\
  &B(t)
  \\
  \neg H(t) \text{ entailed?}
  \end{align*}
Knowledge base \{\neg H \sqcup \exists p.H, B(t), H(t)\}

expansion stops. Cannot find contradiction!

2.: H blocked by t!
Example (0) the other case

Knowledge base \{\neg H \uplus \exists p.H, B(t), \neg H(t)\}

no further expansion possible – knowledge base is satisfiable!
Example(1)

Show, that

Professor $\subseteq (\text{Person} \cap \text{Universitymember})$
$\cup (\text{Person} \cap \neg \text{PhDstudent})$

entails that every Professor is a Person.

Find contradiction in:

$\neg P \cup (E \cap U) \cup (E \cap \neg S)$
$P \cap \neg E(x)$

\[
\begin{align*}
P \cap \neg E \\
P \\
\neg E \\
\neg P \cup (E \cap U) \cup (E \cap \neg S) \\
1. \quad \neg P \text{ (contradiction)} \\
2. \quad (E \cap U) \cup (E \cap \neg S) \\
1. \quad E \cap U \\
\quad \quad E \text{ (contradiction)} \\
2. \quad E \cap \neg S \\
\quad \quad E \text{ (contradiction)}
\end{align*}
\]
Example (2)

Show that
hasChild(john, peter)
hasChild(john, paul)
male(peter)
male(paul)
does not entail ∀ hasChild.male(john).

\[ \neg \forall \text{hasChild.male} \equiv \exists \text{hasChild.} \neg \text{male} \]
Example (3)

Show that the knowledge base
Bird $\sqsubseteq$ Flies
Penguin $\sqsubseteq$ Bird
Penguin $\sqcap$ Flies $\sqsubseteq$ $\bot$
Penguin(tweety)
is unsatisfiable.

TBox:
$\neg B \sqcup F$
$\neg P \sqcup B$
$\neg P \sqcup \neg F \sqcup \bot$

---

P
$\neg P \sqcup B$
$\neg B \sqcup F$
$\neg P \sqcup \neg F$
1. $\neg P$ (contradiction)
2. B
   1. $\neg B$ (contradiction)
   2. F
      1. $\neg P$ (contradiction)
      2. $\neg F$ (contradiction)
Example (4)

Show that the knowledge base

\[ C(a) \quad C(c) \]
\[ R(a,b) \quad R(a,c) \]
\[ S(a,a) \quad S(c,b) \]
\[ C \subseteq \forall S.A \]
\[ A \subseteq \exists R.\exists S.A \]
\[ A \subseteq \exists R.C \]

entails \( \exists R.\exists R.\exists S.A(a) \).
Example (4)

TBox:
¬C ⊔ ∀S.A
¬A ⊔ ∃R.∃S.A
¬A ⊔ ∃R.C

¬∃R.∃R.∃S.A ≡ ∀R.∀R.∀S.¬A
Contents

• Important inference problems
• Tableaux algorithm for ALC
• Tableaux algorithm for SHIQ
Tableaux Algorithm for SHIQ

- Basic idea is the same.
- Blocking rule is more complicated
- Other modifications are also needed.
Given a knowledge base $K$.

- Replace $C \equiv D$ by $C \sqsubseteq D$ and $D \sqsubseteq C$.
- Replace $C \sqsubseteq D$ by $\neg C \sqcup D$.
- Apply the equations on the next slide exhaustively.

Resulting knowledge base: $\text{NNF}(K)$

*Negation normal form* of $K$.

Negation occurs only directly in front of atomic classes.
NNF(C) = C  if C is a class name
NNF(¬C) = ¬C  if C is a class name

NNF(¬¬C) = NNF(C)
NNF(C ⊔ D) = NNF(C) ⊔ NNF(D)
NNF(C ⊓ D) = NNF(C) ⊓ NNF(D)
NNF(¬(C ⊔ D)) = NNF(¬C) ⊓ NNF(¬D)
NNF(¬(C ⊓ D)) = NNF(¬C) ⊔ NNF(¬D)

NNF(∀R.C) = ∀R.NNF(C)
NNF(∃R.C) = ∃R.NNF(C)

NNF(¬∀R.C) = ∃R.NNF(¬C)
NNF(¬∃R.C) = ∀R.NNF(¬C)

NNF(≤n R.C) = ≤n R.NNF(C)
NNF(≥n R.C) = ≥n R.NNF(C)
NNF(¬ ≤n R.C) = ≥(n+1)R.NNF(C)
NNF(¬ ≥n R.C) = ≤(n-1)R.NNF(C), where ≤(-1)R.C = ⊥

K and NNF(K) have the same models (are logically equivalent).
Formal Definition

- A tableau is a directed labeled graph
  - nodes are individuals or (new) variable names
  - nodes $x$ are labeled with sets $L(x)$ of classes
  - edges $<x,y>$ are labeled
    - either with sets $L(<x,y>)$ of role names or inverse role names
    - or with the symbol $=$ (for equality)
    - or with the symbol $\neq$ (for inequality)
Initialisation

• Make a node for every individual in the ABox. These nodes are called *root nodes*.
• Every node is labeled with the corresponding class names from the ABox.
• There is an edge, labeled with R, between a and b, if R(a,b) is in the ABox.
• There is an edge, labeled ≠, between a and b if a ≠ b is in the ABox.
• There are no = relations (yet).
Notions

• We write $S^{-}$ as $S$.
• If $R \in L(<x,y>)$ and $R \subseteq S$ (where $R,S$ can be inverse roles), then
  – $y$ is an $S$-successor of $x$ and
  – $x$ is an $S$-predecessor of $y$.
• If $y$ is an $S$-successor or an $S^{-}$-predecessor of $x$, then $y$ is an neighbor of $x$.
• Ancestor is the transitive closure of Predecessor.
Blocking for SHIQ

• x is blocked by y if x,y are not root nodes and
  – the following hold: ["x is directly blocked"]
    • no ancestor of x is blocked
    • there are predecessors y', x' of x
    • y is a successor of y' and x is a successor of x'
    • L(x) = L(y) and L(x') = L(y')
    • L(<x',x>) = L(<y',y>)
  – or the following holds: ["x is indirectly blocked"]
    • an ancestor of x is blocked or
    • x is successor of some y with L(<y,x>) =∅
Constructing the tableau

• Non-deterministically extend the tableau using the rules on the next slide.

• Terminate, if
  – there is a contradiction in a node label, i.e.,
    • it contains ⊥ or classes C and ¬C or
    • it contains a class ≤ nR.C and
      x also has (n+1) R-successors y_i and y_i ≠ y_j (for all i ≠ j)
  – or none of the rules is applicable.

• If the tableau does not contain a contradiction, then the knowledge base is satisfiable.
  Or more precisely: If you can make a choice of rule applications such that no contradiction occurs and the process terminates, then the knowledge base is satisfiable.
*-rule: If $x$ is not indirectly blocked, $C \sqcap D \in \mathcal{L}(x)$, and $\{C, D\} \not\subseteq \mathcal{L}(x)$, then set $\mathcal{L}(x) \leftarrow \{C, D\}$.

□-rule: If $x$ is not indirectly blocked, $C \sqcup D \in \mathcal{L}(x)$ and $\{C, D\} \cap \mathcal{L}(x) = \emptyset$, then set $\mathcal{L}(x) \leftarrow C$ or $\mathcal{L}(x) \leftarrow D$.

∃-rule: If $x$ is not blocked, $\exists R.C \in \mathcal{L}(x)$, and there is no $y$ with $R \in \mathcal{L}(x, y)$ and $C \in \mathcal{L}(y)$, then

1. add a new node with label $y$ (where $y$ is a new node label),
2. set $\mathcal{L}(x, y) = \{R\}$ and $\mathcal{L}(y) = \{C\}$.

∀-rule: If $x$ is not indirectly blocked, $\forall R.C \in \mathcal{L}(x)$, and there is a node $y$ with $R \in \mathcal{L}(x, y)$ and $C \not\in \mathcal{L}(y)$, then set $\mathcal{L}(y) \leftarrow C$.

TBox-rule: If $x$ is not indirectly blocked, $C$ is a TBox statement, and $C \not\in \mathcal{L}(x)$, then set $\mathcal{L}(x) \leftarrow C$. 
**trans-rule:** If $x$ is not indirectly blocked, $\forall S.C \in \mathcal{L}(x)$, $S$ has a transitive subrole $R$, and $x$ has an $R$-neighbor $y$ with $\forall R.C \notin \mathcal{L}(y)$, then set $\mathcal{L}(y) \leftarrow \forall R.C$.

**choose-rule:** If $x$ is not indirectly blocked, $\leq n S.C \in \mathcal{L}(x)$ or $\geq n S.C \in \mathcal{L}(x)$, and there is an $S$-neighbor $y$ of $x$ with $\{C, \text{NNF}(\neg C)\} \cap \mathcal{L}(y) = \emptyset$, then set $\mathcal{L}(y) \leftarrow C$ or $\mathcal{L}(y) \leftarrow \text{NNF}(\neg C)$.

**$\geq$-rule:** If $x$ is not blocked, $\geq n S.C \in \mathcal{L}(x)$, and there are no $n$ $S$-neighbors $y_1, \ldots, y_n$ of $x$ with $C \in \mathcal{L}(y_i)$ and $y_i \not\equiv y_j$ for $i, j \in \{1, \ldots, n\}$ and $i \neq j$, then

1. create $n$ new nodes with labels $y_1, \ldots, y_n$ (where the labels are new),
2. set $\mathcal{L}(x, y_i) = \{S\}$, $\mathcal{L}(y_i) = \{C\}$, and $y_i \not\equiv y_j$ for all $i, j \in \{1, \ldots, n\}$ with $i \neq j$. 
\[ \leq - \text{rule:} \] If \( x \) is not indirectly blocked, \( \leq_n S.C \in \mathcal{L}(x) \), there are more than \( n \) \( S \)-neighbors \( y_i \) of \( x \) with \( C \in \mathcal{L}(y_i) \), and \( x \) has two \( S \)-neighbors \( y, z \) such that \( y \) is neither a root node nor an ancestor of \( z \), \( y \not\approx z \) does not hold, and \( C \in \mathcal{L}(y) \cap \mathcal{L}(z) \), then

1. set \( \mathcal{L}(z) \leftarrow \mathcal{L}(y) \),
2. if \( z \) is an ancestor of \( x \), then \( \mathcal{L}(z, x) \leftarrow \{ \text{Inv}(R) \mid R \in \mathcal{L}(x, y) \} \),
3. if \( z \) is not an ancestor of \( x \), then \( \mathcal{L}(x, z) \leftarrow \mathcal{L}(x, y) \),
4. set \( \mathcal{L}(x, y) = \emptyset \), and
5. set \( u \not\approx z \) for all \( u \) with \( u \not\approx y \).

\[ \leq - \text{root-rule:} \] If \( \leq_n S.C \in \mathcal{L}(x) \), there are more than \( n \) \( S \)-neighbors \( y_i \) of \( x \) with \( C \in \mathcal{L}(y_i) \), and \( x \) has two \( S \)-neighbors \( y, z \) which are both root nodes, \( y \not\approx z \) does not hold, and \( C \in \mathcal{L}(y) \cap \mathcal{L}(z) \), then

1. set \( \mathcal{L}(z) \leftarrow \mathcal{L}(y) \),
2. for all directed edges from \( y \) to some \( w \), set \( \mathcal{L}(z, w) \leftarrow \mathcal{L}(y, w) \),
3. for all directed edges from some \( w \) to \( y \), set \( \mathcal{L}(w, z) \leftarrow \mathcal{L}(w, y) \),
4. set \( \mathcal{L}(y) = \mathcal{L}(w, y) = \mathcal{L}(y, w) = \emptyset \) for all \( w \),
5. set \( u \not\approx z \) for all \( u \) with \( u \not\approx y \), and
6. set \( y \approx z \).
Example (1): cardinalities

Show, that

\[
\text{hasChild(john, peter)} \quad \text{hasChild(john, paul)} \quad \text{male(peter)} \quad \text{male(paul)} \quad \leq 2\text{hasChild.} \top(john)
\]
does not entail \( \forall \text{hasChild.male(john)} \).

\[
\neg \forall \text{hasChild.male} \equiv \exists \text{hasChild.} \neg \text{male}
\]

now apply \( \leq \)
Example (1): cardinalities

Show, that

hasChild(john, peter)
hasChild(john, paul)
male(peter)
male(paul)

≤ 2hasChild(T(john))

does not entail ∀ hasChild.male(john).

¬∀ hasChild.male ≡ ∃ hasChild.¬male

now apply ≤

backtracking!
Example (1): cardinalities – again

Show, that

\[ \exists \text{hasChild} \cdot \neg \text{male}(\text{john}) \leq \exists \text{hasChild} \cdot \neg \text{male} \]

\[ \neg \forall \text{hasChild} \cdot \text{male}(\text{john}) \]

\[ \text{male}(\text{peter}) \]

\[ \text{male}(\text{paul}) \]

\[ \leq \exists \text{hasChild} \cdot \top(\text{john}) \text{ and } \neg \text{peter} \neq \text{paul} \]

does not entail \[ \forall \text{hasChild} \cdot \text{male}(\text{john}) \].
Example (2): cardinalities

Show, that
\[ \geq 2 \text{hasSon}. \top(\text{john}) \]
entails \[ \geq 2 \text{hasChild}. \top(\text{john}). \]

\[ \neg \geq 2 \text{hasSon}. \top \equiv \leq 1 \text{hasChild}. \top \]

hasSon \subseteq hasChild

hasSon-neighbors are also hasChild-neighbors, tableau terminates with contradiction
Example (3): choose

\[ \geq 3 \text{hasSon}(\text{john}) \]
\[ \leq 2 \text{hasSon.male}(\text{john}) \]
Is this contradictory?

No, because the following tableau is complete.
Example (4): inverse roles

\[ \exists \text{hasChild}\cdot \text{human}(\text{john}) \]
\[ \text{human} \sqsubseteq \forall \text{hasParent}\cdot \text{human} \]
\[ \text{hasChild} \sqsubseteq \text{hasParent}^- \]
zu zeigen: human(\text{john})

\[ \exists \text{hasChild}\cdot \text{human} \]
\[ \neg \text{human} \]
\[ \text{human} \]

\[ \text{john} \xrightarrow{\text{hasChild}} \text{x} \]

\[ \text{human} \]
\[ \forall \text{hasParent}\cdot \text{human} \]

john is hP^--predecessor of x, hence hP-neighbor of x
Example (5): Transitivity and Blocking

\[
\begin{align*}
\text{human} & \subseteq \exists \text{hasFather}. \top \\
\text{human} & \subseteq \forall \text{hasAncestor}. \text{human} \\
\text{hasFather} & \subseteq \text{hasAncestor} \quad \text{Trans(}\text{hasAncestor}\text{)} \\
\text{human(john)} & \\
\text{Does this entail } & \leq 1 \text{hasFather.} \top \text{(john)}? \\
\text{Negation: } & \geq 2 \text{hasFather.} \top \text{(john)}
\end{align*}
\]
Example (5): Transitivity and Blocking

\[ \text{human} \subseteq \exists \text{hasFather}. \top \]
\[ \text{hasFather} \subseteq \text{hasAncestor} \]
\[ \forall \text{hasAncestor} . \text{human}(\text{john}) \]
\[ \text{human}(\text{john}) \]

\[ \text{Trans(\text{hasAncestor})} \]
\[ \geq 2 \text{hasFather}. \top(\text{john}) \]

\[
\begin{align*}
\text{h} & \geq 2hF. \top \\
\forall hA.h & \Rightarrow hF. \top
\end{align*}
\]

\[
\begin{align*}
\text{h} & \Rightarrow hF. \top \\
\forall hA.h & \Rightarrow hF. \top
\end{align*}
\]

\[
\begin{align*}
\text{h} & \Rightarrow hF. \top \\
\forall hA.h & \Rightarrow hF. \top
\end{align*}
\]

\[ x \]
\[ x_1 \]
\[ x_2 \]

\[ hF \rightarrow x \]
\[ hF \rightarrow x_1 \]
\[ hF \rightarrow x_2 \]

\[ h \]
\[ \forall hA.h \]

\[ y \quad \ldots \]

same as branch above

\[ \text{x}_2 \text{ now blocked by } \text{x}_1 : \]
\[ \text{Pair } (\text{x}_1,\text{x}_2) \text{ repeats } (\text{x},\text{x}_1) \]
Example (6): Pairwise Blocking

\[ \neg C \sqcap (\leq 1F) \sqcap \exists F^-.D \sqcap \forall R^-.(\exists F^- . D), \text{ where} \]
\[ D = C \sqcap (\leq 1F) \sqcap \exists F^- . \neg C, \text{ Trans}(R), \text{ and } F \subseteq R, \]

is not satisfiable.

\[ \neg C \]
\[ \leq 1F \]
\[ \exists F^- . D \]
\[ \forall R^- . (\exists F^- . D) \]
\[ C \]
\[ \leq 1F \]
\[ \exists F . \neg C \]

Without pairwise blocking, z would be blocked, which shouldn’t happen:
Expansion of \( \exists F . \neg C \) yields \( \neg C \) for node y as required.
Example (7): Dynamic Blocking

\[ A \land \exists S. (\exists R. T \land \exists P. T \land \forall R. C \land \forall P. (\exists R. T) \land \forall P. (\forall R. C)) \]

with \( C = \forall R^-. (\forall P^- . (\forall S^- . \neg A)) \) and \( \text{Trans}(P) \), is not satisfiable.

Part of the tableau:

At this stage, \( z \) would be blocked by \( y \) (assuming the presence of another pair). However, when \( C \) from \( v \) is expanded, \( z \) becomes unblocked, which is necessary in order to label \( w \) with \( C \) which in turn labels \( x \) with \( \neg A \), yielding the required contradiction.
Tableaux Reasoners

- **Fact++**
  - [http://owl.man.ac.uk/factplusplus/](http://owl.man.ac.uk/factplusplus/)

- **Pellet**

- **RacerPro**
  - [http://www.sts.tu-harburg.de/~r.f.moeller/racer/](http://www.sts.tu-harburg.de/~r.f.moeller/racer/)
Please don’t forget the preparations for the interactive class project session.