Knowledge Representation for the Semantic Web

Winter Quarter 2012

Slides 10 – 02/21/2012

Pascal Hitzler

Kno.e.sis Center

Wright State University, Dayton, OH

http://www.knoesis.org/pascal/

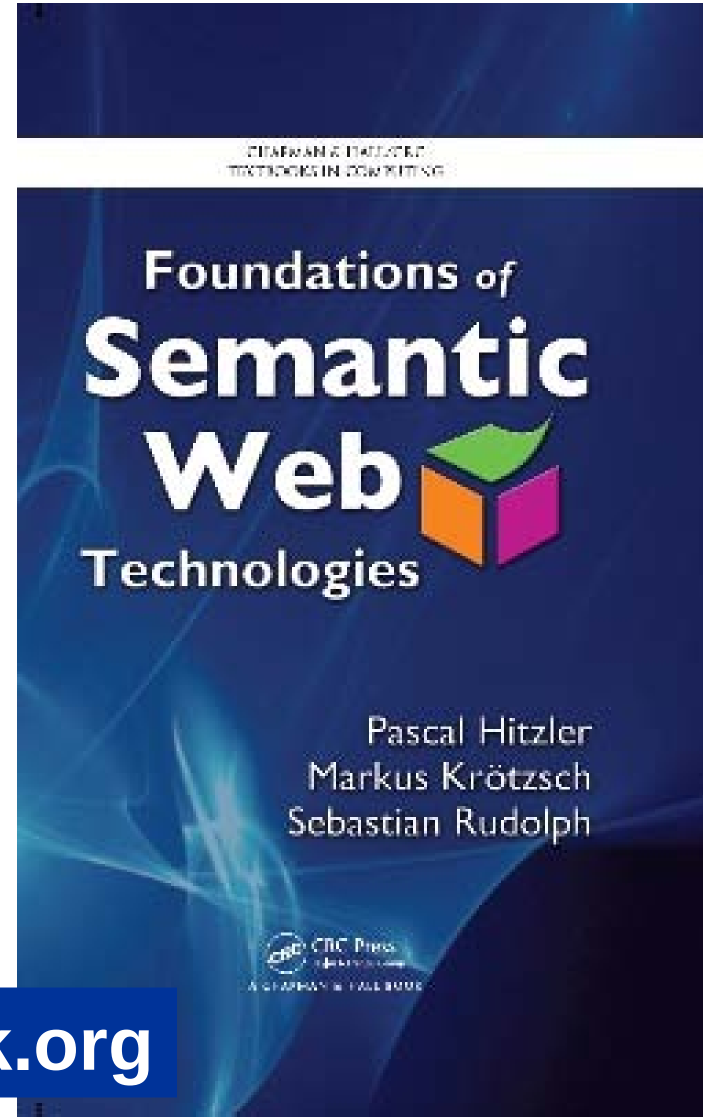
Textbook (required)

Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph

Foundations of Semantic Web Technologies

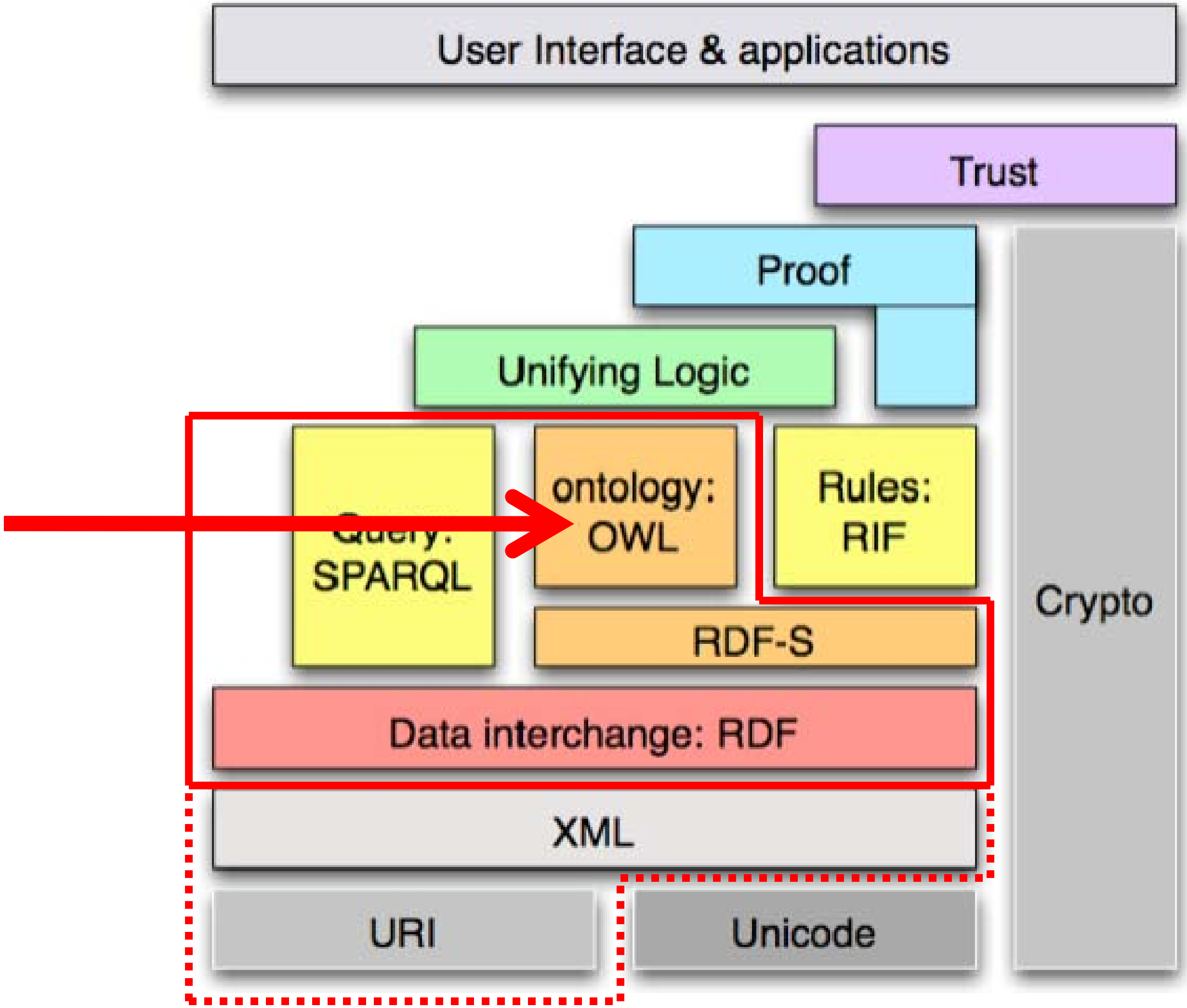
Chapman & Hall/CRC, 2010

Choice Magazine Outstanding Academic Title 2010 (one out of seven in Information & Computer Science)



http://www.semantic-web-book.org

Today: Reasoning with OWL



A Reasoning Problem

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. $KB \models$ **false**?
  - Is the knowledge base meaningful?
- Class consistency $C \equiv \bot$?
  - Is C necessarily empty?
- Class inclusion (Subsumption) $C \sqsubseteq D$?
  - Structuring knowledge bases
- Class equivalence $C \equiv D$?
  - Are two classes in fact the same class?
- Class disjointness $C \sqcap D = \bot$?
  - 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.** — **KB unsatisfiable**
  - **Failure to find a model.**
- **Class consistency** — $C \equiv \bot$?
  - **KB** $\cup$ **{C(a)} unsatisfiable**
- **Class inclusion (Subsumption)** — $C \sqsubseteq D$?
  - **KB** $\cup$ **{**$C \sqcap \neg D(a)$**} unsatisfiable (a new)**
- **Class equivalence** — $C \equiv D$?
  - $C \sqsubseteq D$ **und** $D \sqsubseteq C$
- **Class disjointness** — $C \sqcap D = \bot$?
  - **KB** $\cup$ **{**$(C \sqcap D)(a)$**} unsatisfiable (a new)**
- **Class membership** — **C(a)?**
  - **KB** $\cup$ **{**$\neg C(a)$**} unsatisfiable**
- **Instance Retrieval** — **„find all x with C(x)“**
  - **Check class membership for all individuals.**

Reduction to Satisfiability

- **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

Transform. to negation normal form

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: NNF(K)

Negation normal form of K.

Negation occurs only directly in front of atomic classes.

$$
\begin{aligned}
\mathrm{NNF}(C) &= C \qquad \text{if } C \text{ is a class name} \\
\mathrm{NNF}(\neg C) &= \neg C \qquad \text{if } C \text{ is a class name} \\
\mathrm{NNF}(\neg\neg C) &= \mathrm{NNF}(C) \\
\mathrm{NNF}(C \sqcup D) &= \mathrm{NNF}(C) \sqcup \mathrm{NNF}(D) \\
\mathrm{NNF}(C \sqcap D) &= \mathrm{NNF}(C) \sqcap \mathrm{NNF}(D) \\
\mathrm{NNF}(\neg(C \sqcup D)) &= \mathrm{NNF}(\neg C) \sqcap \mathrm{NNF}(\neg D) \\
\mathrm{NNF}(\neg(C \sqcap D)) &= \mathrm{NNF}(\neg C) \sqcup \mathrm{NNF}(\neg D) \\
\mathrm{NNF}(\forall R.C) &= \forall R.\mathrm{NNF}(C) \\
\mathrm{NNF}(\exists R.C) &= \exists R.\mathrm{NNF}(C) \\
\mathrm{NNF}(\neg\forall R.C) &= \exists R.\mathrm{NNF}(\neg C) \\
\mathrm{NNF}(\neg\exists R.C) &= \forall R.\mathrm{NNF}(\neg C)
\end{aligned}
$$

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

Example

$P \sqsubseteq (E \sqcap U) \sqcup \neg(\neg E \sqcup D).$

In negation normal form:

$\neg P \sqcup (E \sqcap U) \sqcup (E \sqcap \neg D).$

ALC tableaux: contents

- Transformation to negation normal form
- Naive tableaux algorithm
- Tableaux algorithm with blocking

Naive tableaux algorithm

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**
  $\rightarrow$ 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:**
  $\rightarrow$ 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

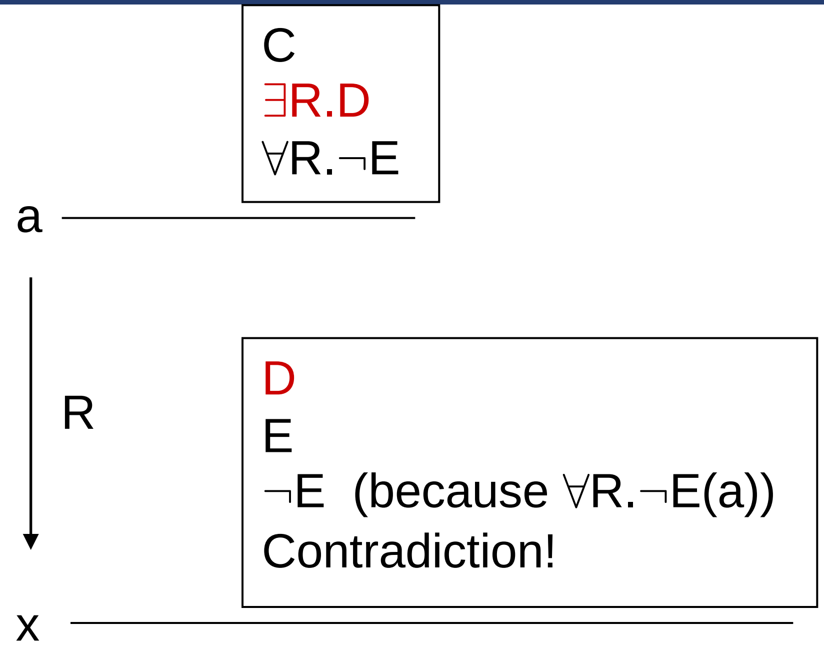
C(a)

$C \sqsubseteq \exists R.D$

$D \sqsubseteq E$

Does this entail $(\exists R.E)(a)$?

(add $\forall R.\neg E(a)$ and show unsatisfiability)



Another example

C(a)

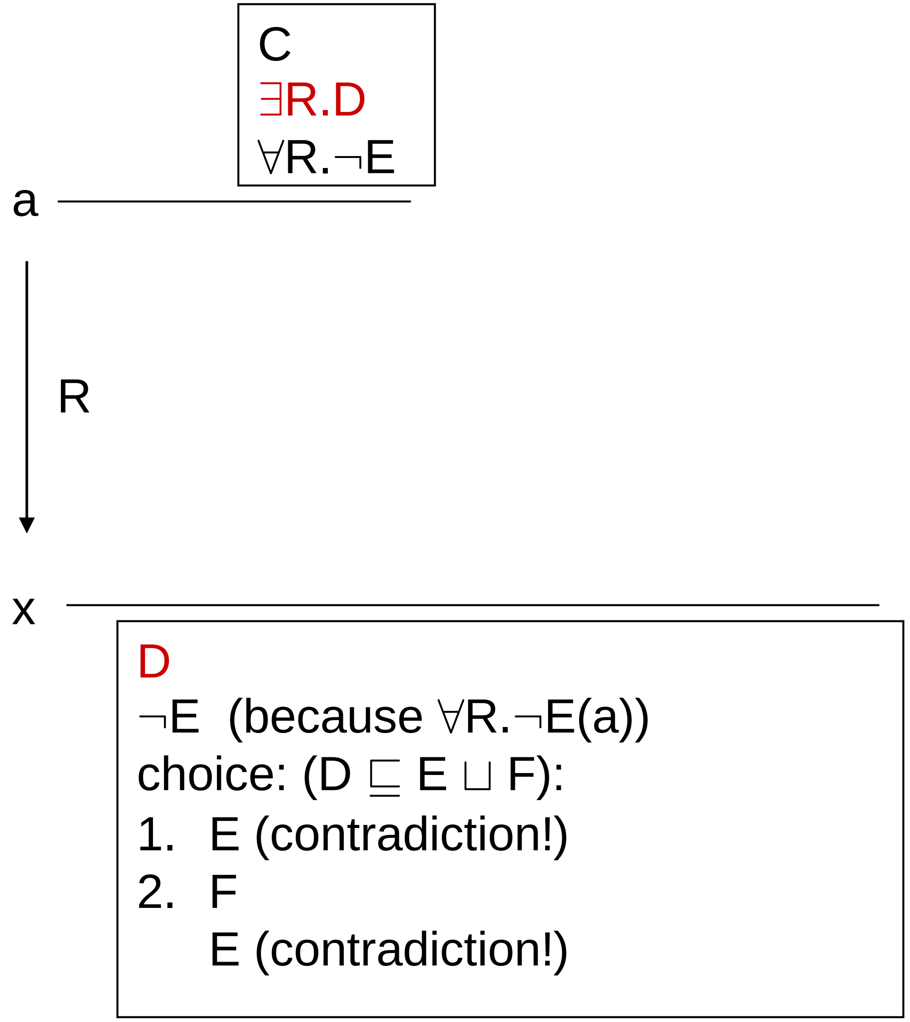
$C \sqsubseteq \exists R.D$

$D \sqsubseteq E \sqcup F$

$F \sqsubseteq E$

Does this entail $(\exists R.E)(a)$?

(add $\forall R.\neg E(a)$ and show unsatisfiability)



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.)

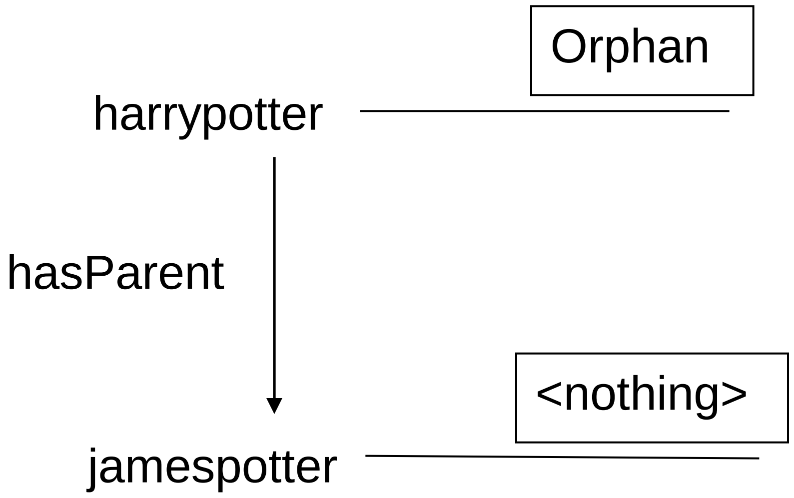
Example initialisation

Human ⊑ ∃hasParent.Human

Orphan ⊑ Human ⊓ ¬∃hasParent.Alive

Orphan(harrypotter)

hasParent(harrypotter,jamespotter)



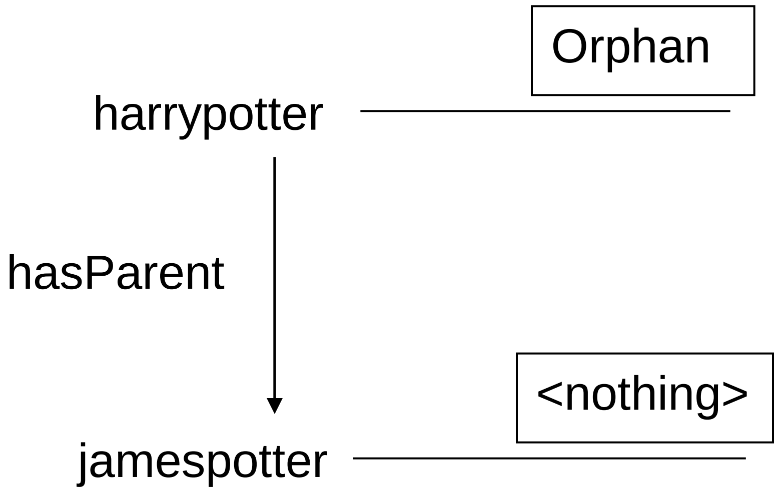
Careful: need NNF!

¬Human ⊔ ∃hasParent.Human

¬Orphan ⊔ (Human ⊓ ∀hasParent.¬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 ¬C, or it contains ⊥), 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

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

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

$\exists$-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 $R \in \mathcal{L}(x, y)$ 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

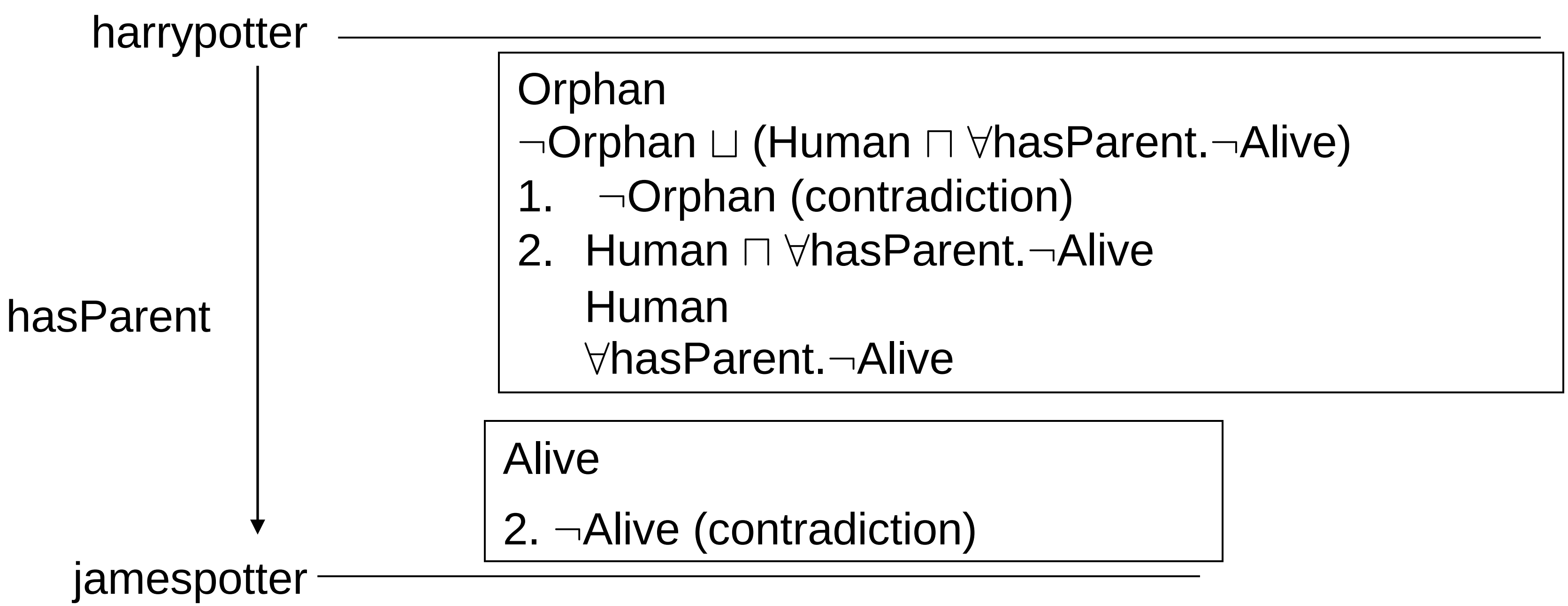
$\neg$Alive(jamespotter)
i.e. add: Alive(jamespotter)
and search for contradiction

$\neg \text{Human} \sqcup \exists \text{hasParent.Human}$

$\neg \text{Orphan} \sqcup (\text{Human} \sqcap \forall \text{hasParent.}\neg \text{Alive})$

Orphan(harrypotter)

hasParent(harrypotter,jamespotter)



ALC tableaux: contents

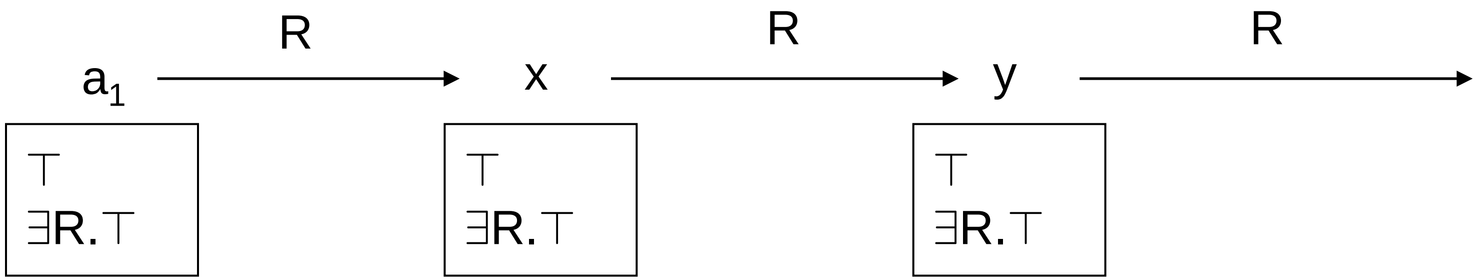
- Transformation to negation normal form
- Naive tableaux algorithm
- Tableaux algorithm with blocking

There's a termination problem

TBox: $\exists R.\top$

ABox: $\top(a_1)$

- **Obviously satisfiable:**
  Model M with domain elements $a_1^M, a_2^M, \ldots$
  and $R^M(a_i^M, a_{i+1}^M)$ for all $i \geq 1$
- **but tableaux algorithm does not terminate!**

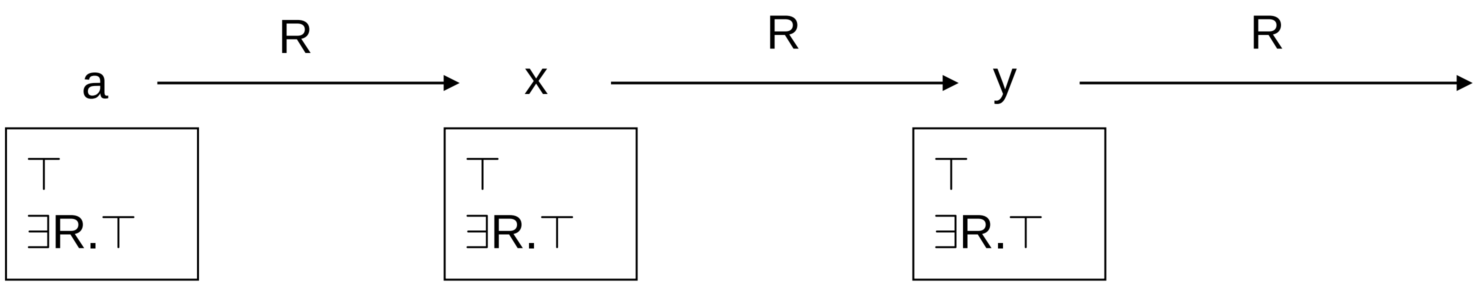


Solution?

Actually, things repeat!

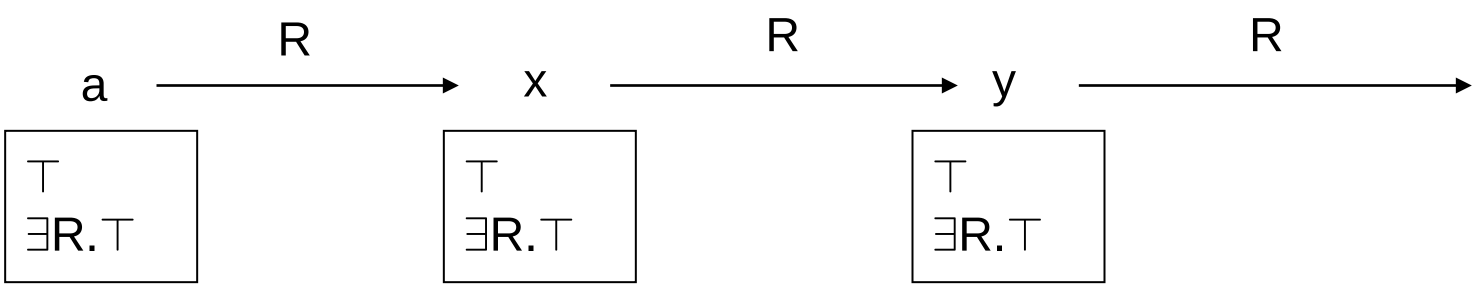
Idea: it is not necessary to expand x, since it‘s simply a copy of a.

⇒ 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 ¬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

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

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

$\exists$-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 $R \in \mathcal{L}(x, y)$ 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 $\mathcal{L}(x) \leftarrow C$.

Apply only if x is not blocked!

Example (0)

- **Knowledge base {Human ⊑ ∃hasParent.Human, Bird(tweety)}**
- **We want to show that Human(tweety) does *not* hold,**
  i.e. that ¬Human(tweety) is entailed.
- **We will not be able to show this.**
  I.e. Human(tweety) is *possible*.

- **Shorter notation:**
  H ⊑ ∃p.H
  B(t)

  ¬H(t) entailed?

Example (0)

Knowledge base {¬H ⊔ ∃p.H, B(t), H(t)}

t —p→ x

t:

H
B
¬H ⊔ ∃p.H
1. ¬H (contradiction)
2. ∃p.H

x:

2.:
H
blocked by t!

expansion stops. Cannot find contradiction!

Example (0) the other case

Knowledge base {¬H ⊔ ∃p.H, B(t), ¬H(t)}

t

¬H B ¬H ⊔ ∃p.H

No further expansion possible: ⊔-rule not applicable!
Hence knowledge base is satisfiable.

Example(1)

Show, that

$$\text{Professor} \sqsubseteq (\text{Person} \sqcap \text{Unversitymember}) \sqcup (\text{Person} \sqcap \neg\text{PhDstudent})$$

entails that every Professor is a Person.

Find contradiction in:
$\neg P \sqcup (E \sqcap U) \sqcup (E \sqcap \neg S)$
$P \sqcap \neg E(x)$

x

$P \sqcap \neg E$
P
$\neg E$
$\neg P \sqcup (E \sqcap U) \sqcup (E \sqcap \neg S)$

1. $\neg P$ (contradiction)
2. $(E \sqcap U) \sqcup (E \sqcap \neg S)$
    1. $E \sqcap U$
       E (contradiction)
    2. $E \sqcap \neg S$
       E (contradiction)

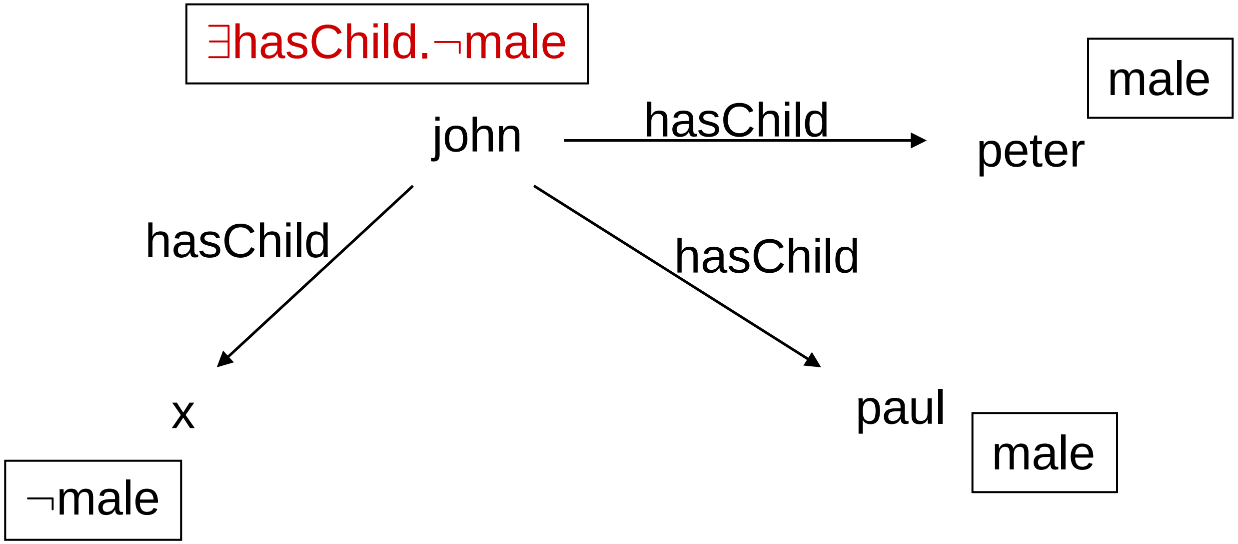
Example (2)

Show that

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

does *not* entail ∀hasChild.male(john).

¬∀hasChild.male ≡ ∃hasChild.¬male



Example (3)

Show that the knowledge base

Bird $\sqsubseteq$ Flies
Penguin $\sqsubseteq$ Bird
Penguin $\sqcap$ Flies $\sqsubseteq \bot$
Penguin(tweety)

is unsatisfiable.

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

tweety

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

$C(a)$

$R(a,b)$

$S(a,a)$

$C \sqsubseteq \forall S.A$

$A \sqsubseteq \exists R.\exists S.A$

$A \sqsubseteq \exists R.C$

$C(c)$

$R(a,c)$

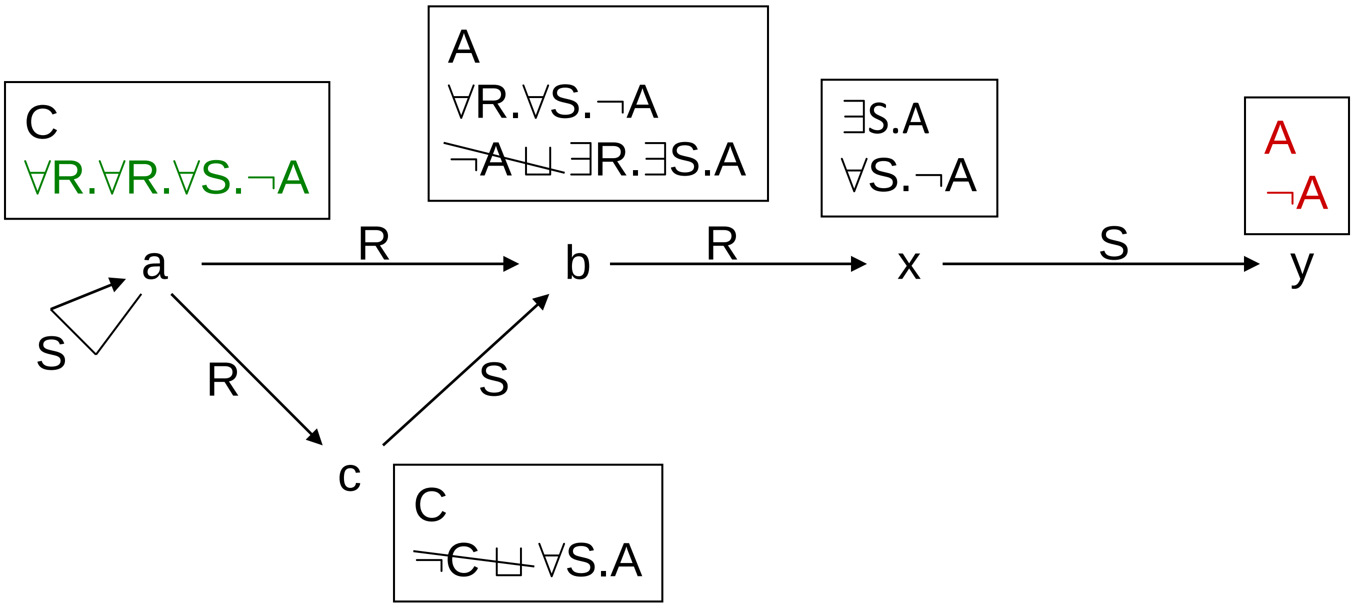
$S(c,b)$

entails $\exists R.\exists R.\exists S.A(a)$.

Example (4)

¬∃R.∃R.∃S.A ≡ ∀R.∀R.∀S.¬A

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



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 modifictions are also needed.

Transform. to negation normal form

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: NNF(K)

Negation normal form of K.

Negation occurs only directly in front of atomic classes.

$$
\begin{aligned}
\mathrm{NNF}(C) &= C \quad \text{if } C \text{ is a class name}\\
\mathrm{NNF}(\neg C) &= \neg C \quad \text{if } C \text{ is a class name}\\
\mathrm{NNF}(\neg\neg C) &= \mathrm{NNF}(C)\\
\mathrm{NNF}(C \sqcup D) &= \mathrm{NNF}(C) \sqcup \mathrm{NNF}(D)\\
\mathrm{NNF}(C \sqcap D) &= \mathrm{NNF}(C) \sqcap \mathrm{NNF}(D)\\
\mathrm{NNF}(\neg(C \sqcup D)) &= \mathrm{NNF}(\neg C) \sqcap \mathrm{NNF}(\neg D)\\
\mathrm{NNF}(\neg(C \sqcap D)) &= \mathrm{NNF}(\neg C) \sqcup \mathrm{NNF}(\neg D)\\
\mathrm{NNF}(\forall R.C) &= \forall R.\mathrm{NNF}(C)\\
\mathrm{NNF}(\exists R.C) &= \exists R.\mathrm{NNF}(C)\\
\mathrm{NNF}(\neg\forall R.C) &= \exists R.\mathrm{NNF}(\neg C)\\
\mathrm{NNF}(\neg\exists R.C) &= \forall R.\mathrm{NNF}(\neg C)
\end{aligned}
$$

NNF($\leq$n R.C) = $\leq$n R.NNF(C)

NNF($\geq$n R.C) = $\geq$n R.NNF(C)

NNF($\neg$ $\leq$n R.C) = $\geq$(n+1)R.NNF(C)

NNF($\neg$ $\geq$n R.C) = $\leq$(n-1)R.NNF(C), where $\leq$(-1)R.C = $\perp$

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 ≠ (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 $\neq$, between a and b if $a \neq b$ is in the ABox.
- There are no = relations (yet).

Notions

- **We write S^{--} as S.**
- **If $R \in L(<x,y>)$ and $R \sqsubseteq S$ (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 $\bot$ or classes C and $\neg$C or**
    - **it contains a class $\leq$ nR.C and x also has (n+1) R-successors y_i and $y_i \neq y_j$ (for all i $\neq$ 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.

SHIQ Tableaux Rules

$\sqcap$-rule: If x is not indirectly blocked, $C \sqcap D \in \mathcal{L}(x)$, and $\{C, D\} \not\subseteq \mathcal{L}(x)$, then set $\mathcal{L}(x) \leftarrow \{C, D\}$.

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

$\exists$-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 $\mathcal{L}(y) = \{C\}$.

$\forall$-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 $C \notin \mathcal{L}(y)$, then set $\mathcal{L}(y) \leftarrow C$.

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

SHIQ Tableaux Rules

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, $\leq nS.C \in \mathcal{L}(x)$ or $\geq nS.C \in \mathcal{L}(x)$, 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, $\geq nS.C \in \mathcal{L}(x)$, and there are no n S-neighbors $y_1, \ldots, y_n$ of x with $C \in \mathcal{L}(y_i)$ and $y_i \not\approx y_j$ 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 $y_i \not\approx y_j$ for all $i, j \in \{1, \ldots, n\}$ with $i \neq j$.

$\leq$-rule: If x is not indirectly blocked, ${\leq}nS.C \in \mathcal{L}(x)$, 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 $\mathcal{L}(z, x) \leftarrow \{\mathrm{Inv}(R) \mid R \in \mathcal{L}(x, y)\}$,
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 $u \not\approx y$.

$\leq$-root-rule: If ${\leq}nS.C \in \mathcal{L}(x)$, 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 $y \approx z$.

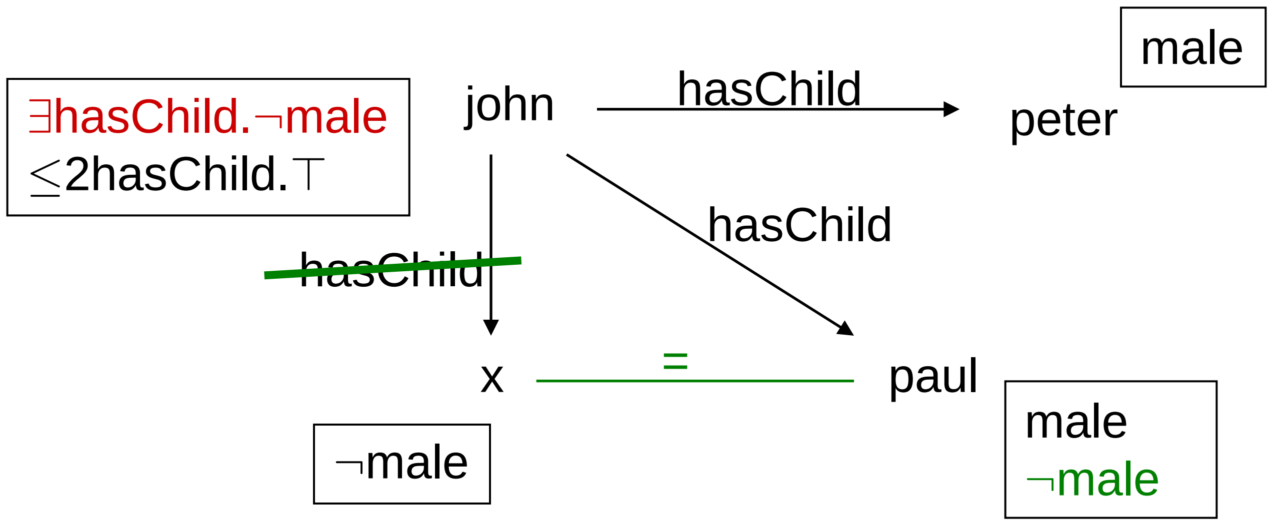
Show, that

hasChild(john, peter)
hasChild(john, paul)
male(peter)
male(paul)
$\leq$2hasChild.$\top$(john)

does *not* entail $\forall$hasChild.male(john).

$\neg\forall$hasChild.male $\equiv$ $\exists$hasChild.$\neg$male

now apply $\leq$



Example (1): cardinalities

Show, that

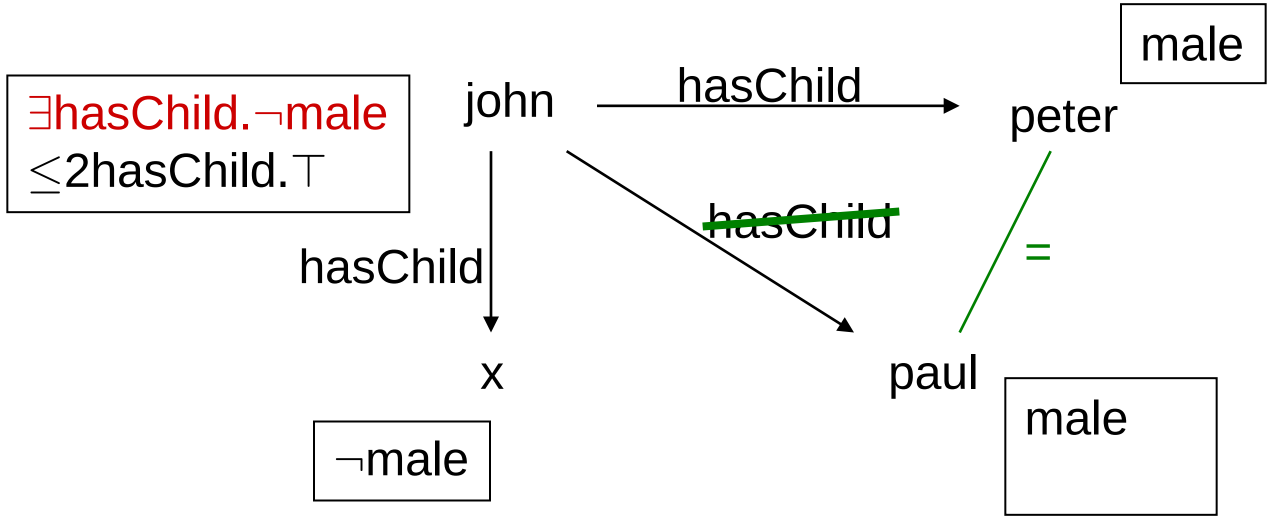
hasChild(john, peter)
hasChild(john, paul)
male(peter)
male(paul)
$\leq$2hasChild.$\top$(john)

does *not* entail $\forall$hasChild.male(john).

$\neg\forall$hasChild.male $\equiv$ $\exists$hasChild.$\neg$male

backtracking!

now apply $\leq$



Example (1): cardinalities – again

Show, that

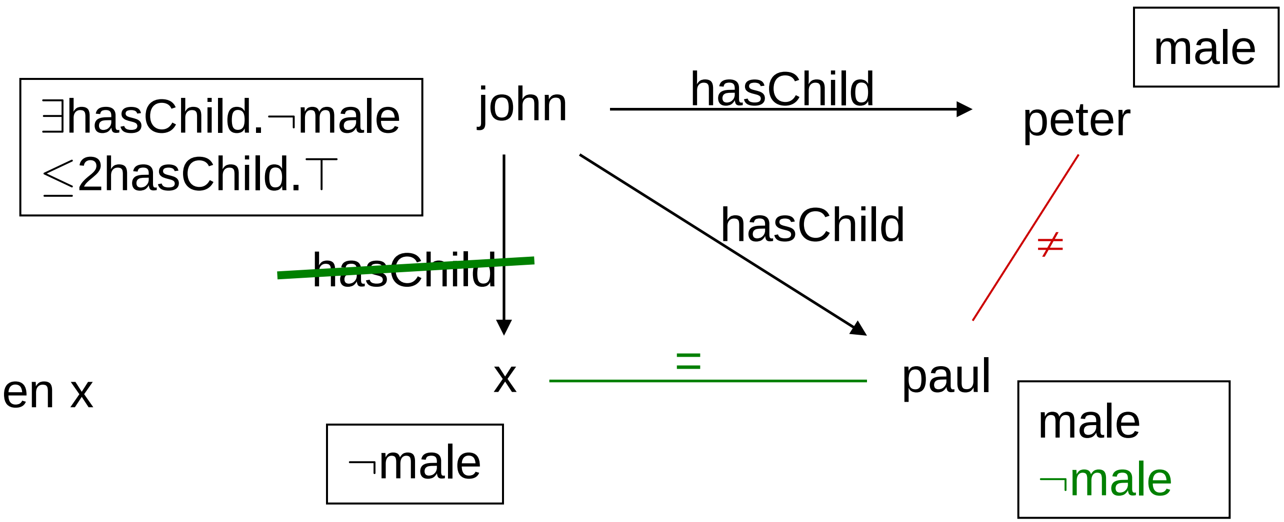
hasChild(john, peter)
hasChild(john, paul)
male(peter)
male(paul)
$\leq$2hasChild.⊤(john) and peter ≠ paul

does *not* entail ∀hasChild.male(john).

¬∀hasChild.male ≡ ∃hasChild.¬male

now apply $\leq$

can backtrack only between x and peter – also leads to contradiction



Example (2): cardinalities

Show, that

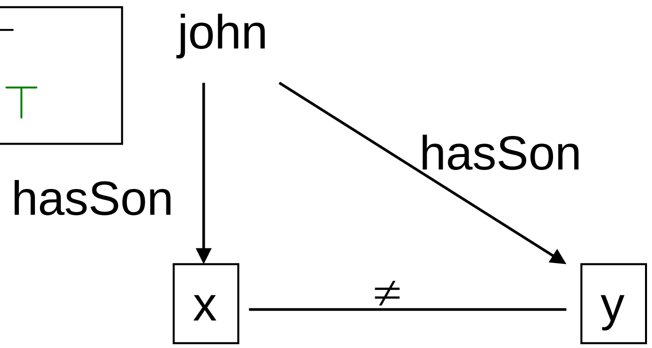
$\geq$2hasSon.$\top$(john)

entails $\geq$2hasChild.$\top$(john).

$\neg\geq 2 hasSon.\top \equiv \leq 1 hasChild.\top$

hasSon $\sqsubseteq$ hasChild

$\geq$2hasSon.$\top$
$\leq$1hasChild.$\top$



hasSon-neighbors are also hasChild-neighbors, tableau terminates with contradiction

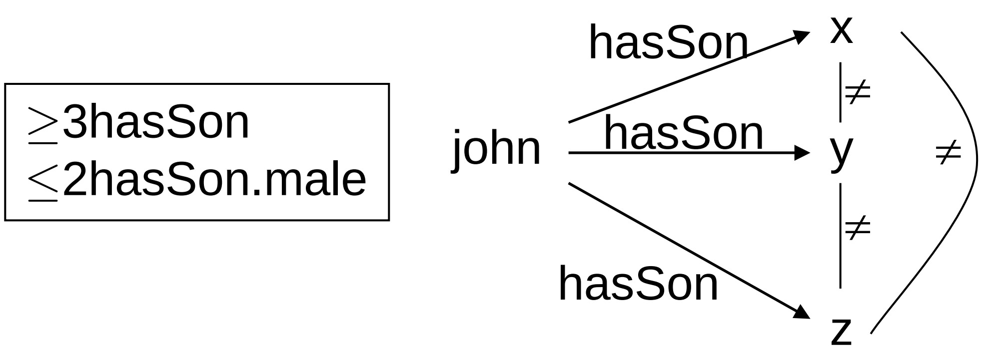
Example (3): choose

$\geq$3hasSon(john)

$\leq$2hasSon.male(john)

Is this contradictory?

No, because the following tableau is complete.



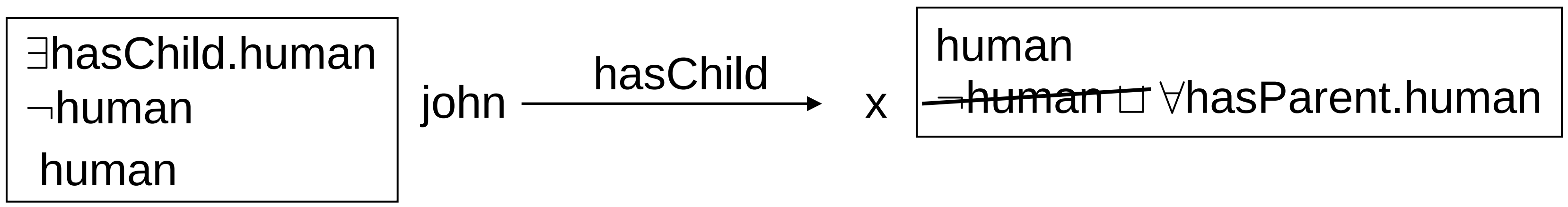
Example (4): inverse roles

∃hasChild.human(john)

human ⊑ ∀hasParent.human

hasChild ⊑ hasParent^{-}

zu zeigen: human(john)



john is hP^{-}-predecessor of x, hence hP-neighbor of x

Example (5): Transitivity and Blocking

human $\sqsubseteq$ $\exists$hasFather.$\top$

human $\sqsubseteq$ $\forall$hasAncestor.human

hasFather $\sqsubseteq$ hasAncestor **Trans(hasAncestor)**

human(john)

Does this entail $\leq$1hasFather.$\top$(john)?

Negation: $\geq$2hasFather.$\top$(john)

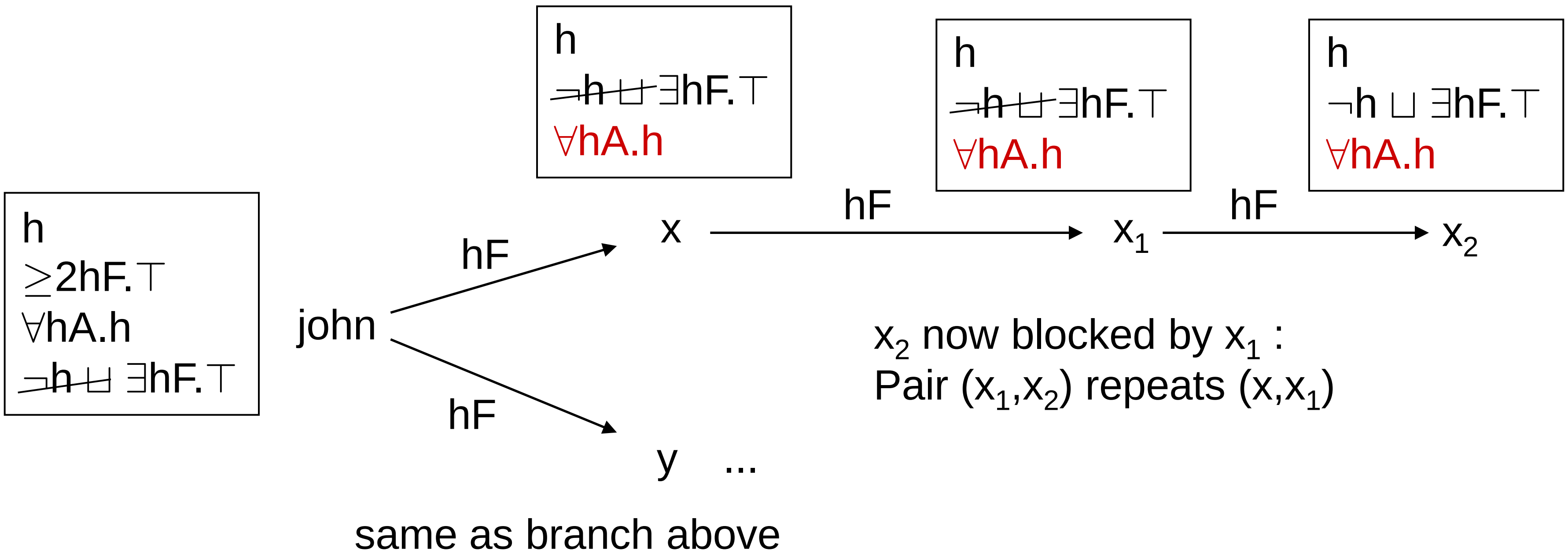
Example (5): Transitivity and Blocking

$\text{human} \sqsubseteq \exists \text{hasFather}.\top$

$\text{hasFather} \sqsubseteq \text{hasAncestor}$ **$\text{Trans}(\text{hasAncestor})$**

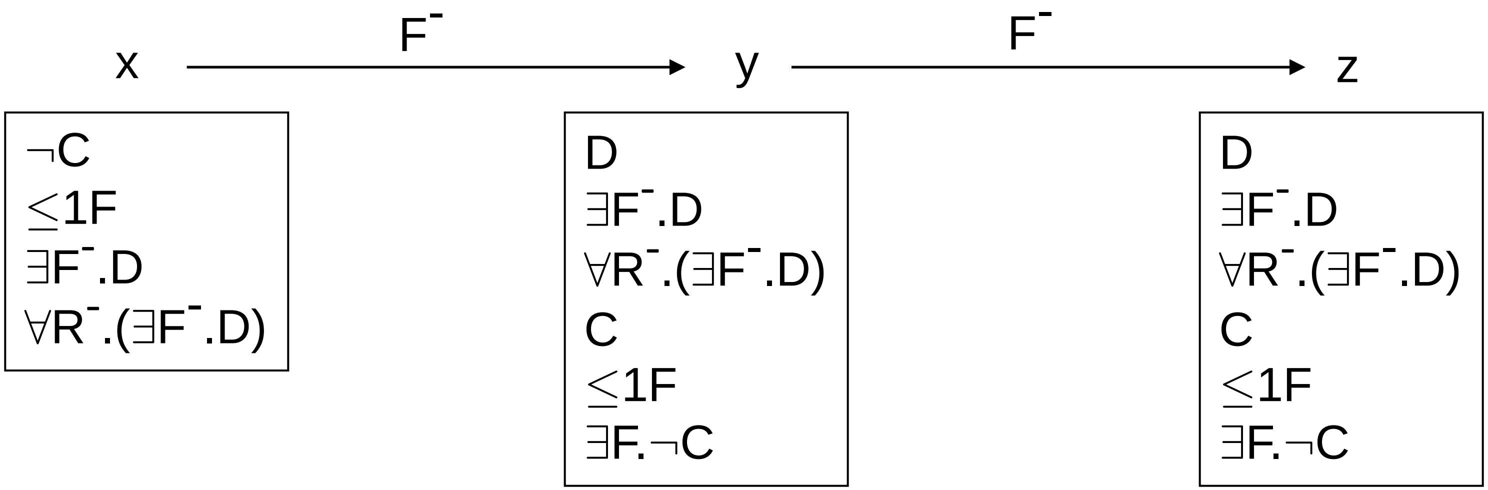
$\forall \text{hasAncestor}.\text{human}(\text{john})$

$\text{human}(\text{john})$ **$\geq 2\text{hasFather}.\top(\text{john})$**



Example (6): Pairwise Blocking

$\neg C \sqcap (\leq 1F) \sqcap \exists F^{-}.D \sqcap \forall R^{-}.(\exists F^{-}.D)$, where
$D = C \sqcap (\leq 1F) \sqcap \exists F.\neg C$, Trans(R), and $F \sqsubseteq R$,
is not satisfiable.



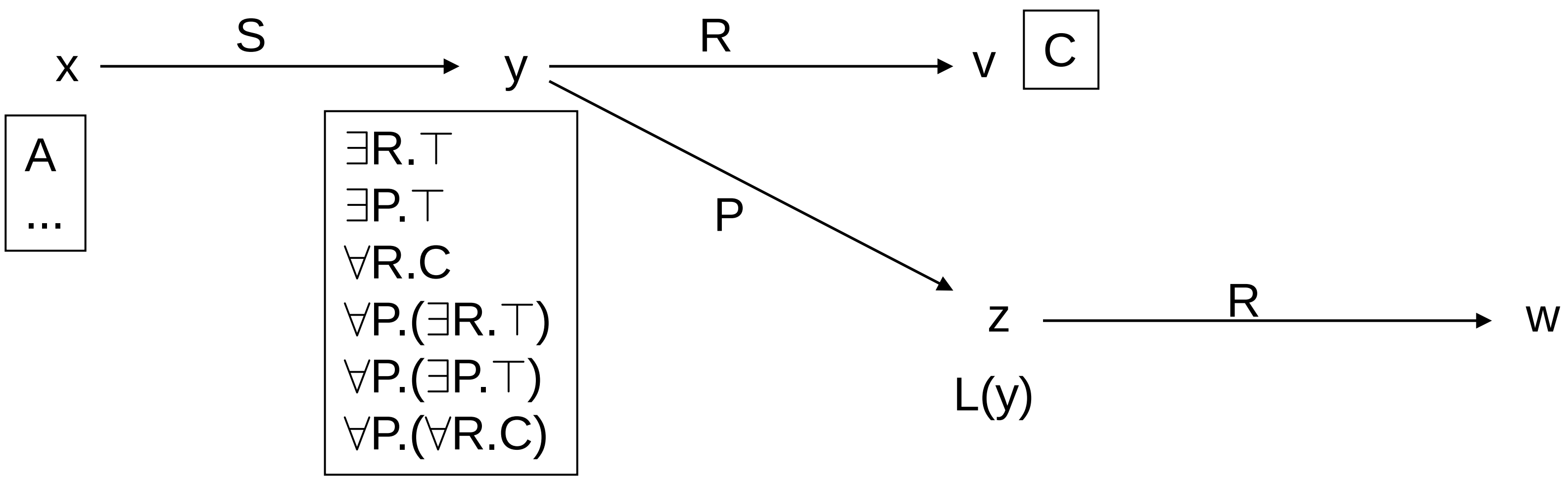
Without pairwise blocking, z would be blocked, which shouldn‘t happen:
Expansion of $\exists F.\neg C$ yields $\neg C$ for node y as required.

Example (7): Dynamic Blocking

$A \sqcap \exists S.(\exists R.\top \sqcap \exists P.\top \sqcap \forall R.C \sqcap \forall P.(\exists R.\top) \sqcap \forall P.(\forall R.C) \sqcap \forall P.(\exists P.\top))$

with $C = \forall R^-.(\forall P^-.(\forall S^-.\neg A))$ and 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/**
- **Pellet**
  - **http://clarkparsia.com/pellet/**
- **RacerPro**
  - **http://www.racer-systems.com/products/racerpro/**
  - **Volker Haarslev, Kay Hidde, Ralf Möller, Michael Wessel, The RacerPro Knowledge Representation and Reasoning System. Semantic Web journal, to appear.**
- **Hermit**
  - **http://hermit-reasoner.com/**