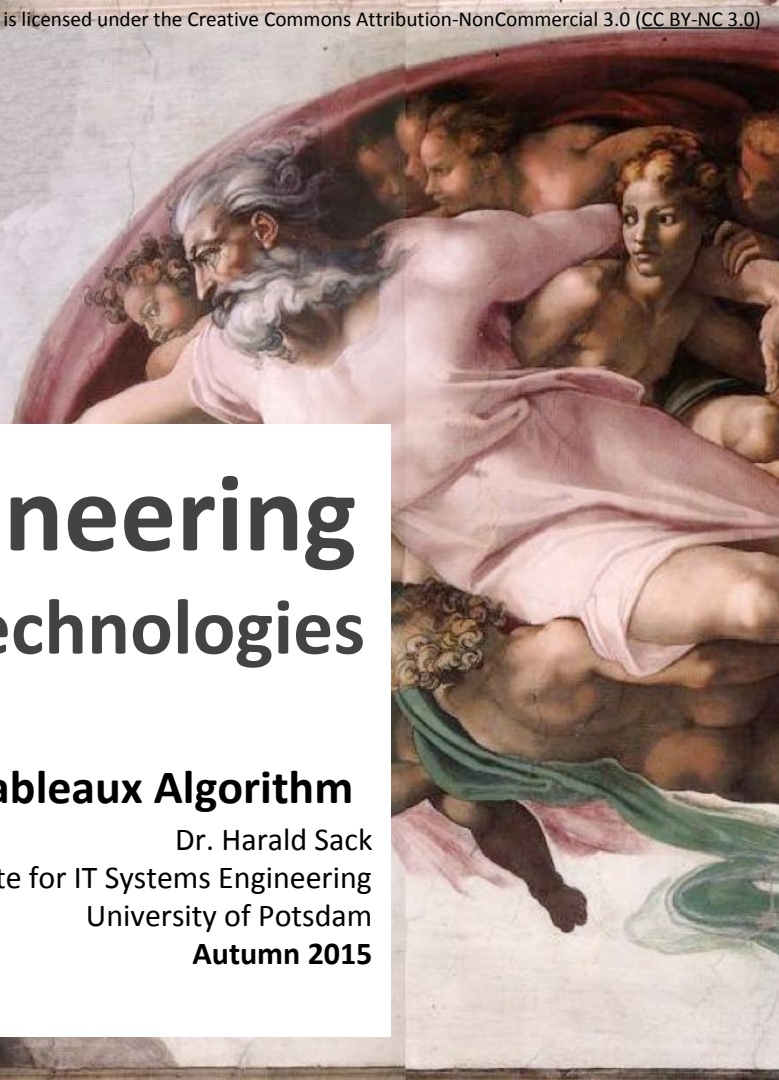
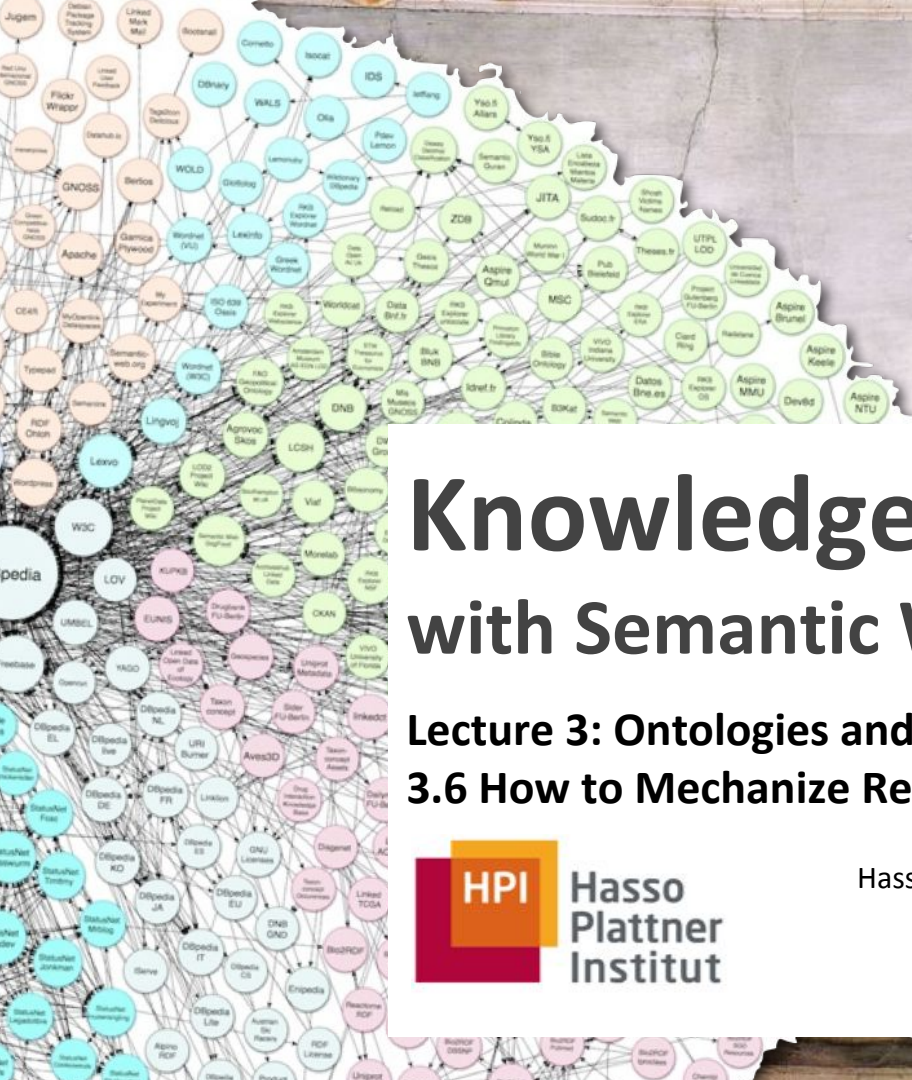




Knowledge Engineering with Semantic Web Technologies

Lecture 3: Ontologies and Logic

3.6 How to Mechanize Reasoning - Tableaux Algorithm



Dr. Harald Sack
Hasso Plattner Institute for IT Systems Engineering
University of Potsdam
Autumn 2015

How to Mechanize Reasoning?

- **Recall:**
 - A formula $\mathbb{F}$ is a **logical consequence** of a theory (knowledge base) $\mathbb{T}$, i.e. $\mathbb{T} \models \mathbb{F}$, iff all models of $\mathbb{T}$ are also models of $\mathbb{F}$
- **Problem:**
 - We have to consider all possible interpretations.
 - How do we do this in practice?

How to Mechanize Reasoning?

- **Problem:**
 - We have to consider all possible interpretations.
 - How do we do this in practice?
- Therefore, **logical consequence** ($\models$) is implemented via **syntactical methods** ($\vdash$) (= **Calculus**).
- For the **logical calculus**, there must be proven:
 - **Correctness:** every syntactic entailment is also a semantic entailment,
if $T \vdash F$ then $T \models F$
 - **Completeness:** all semantic entailments are also syntactic entailments,
if $T \models F$ then $T \vdash F$

Proof by Refutation

- **Idea:**
 - to **check the consistency** of a logical formula we switch the problem to its inverse and show that its **negation is a contradiction**.
- Let F be a formula and T a theory, then
 - instead of showing $T \models F$
 - we show that $T \cup \{\neg F\} \models \perp$
- **Algorithms** for automated reasoning are based on Proof by Refutation:
 - Resolution (cf. EXTRA 3.12)
 - Tableaux Algorithm (cf. 3.6 and 3.9)

Some Logical Equivalences

$$F \wedge G \equiv G \wedge F$$

$$F \vee G \equiv G \vee F$$

$$F \rightarrow G \equiv \neg F \vee G$$

$$F \leftrightarrow G \equiv (F \rightarrow G) \wedge (G \rightarrow F)$$

$$\neg (F \wedge G) \equiv \neg F \vee \neg G$$

$$\neg (F \vee G) \equiv \neg F \wedge \neg G$$

$$\neg \neg F \equiv F$$

$$F \wedge t \equiv F$$

$$F \wedge f \equiv f$$

$$F \wedge \neg F = f$$

$$F \vee t \equiv t$$

$$F \vee f \equiv F$$

$$F \vee \neg F = t$$

$$F \vee (G \wedge H) \equiv (F \vee G) \wedge (F \vee H)$$

$$F \wedge (G \vee H) \equiv (F \wedge G) \vee (F \wedge H)$$

$$\neg (\forall X) F \equiv (\exists X) \neg F$$

$$\neg (\exists X) F \equiv (\forall X) \neg F$$

$$(\forall X) (\forall Y) F \equiv (\forall Y) (\forall X) F$$

$$(\exists X) (\exists Y) F \equiv (\exists Y) (\exists X) F$$

$$(\forall X) (F \wedge G) \equiv (\forall X) F \wedge (\forall X) G$$

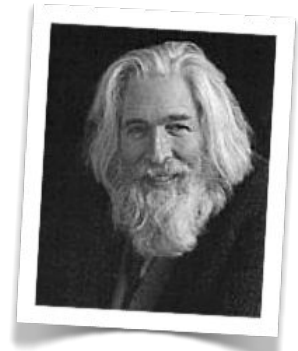
$$(\exists X) (F \vee G) \equiv (\exists X) F \vee (\exists X) G$$

Tableaux Algorithm



Evert Willem Beth
(1908-1964)

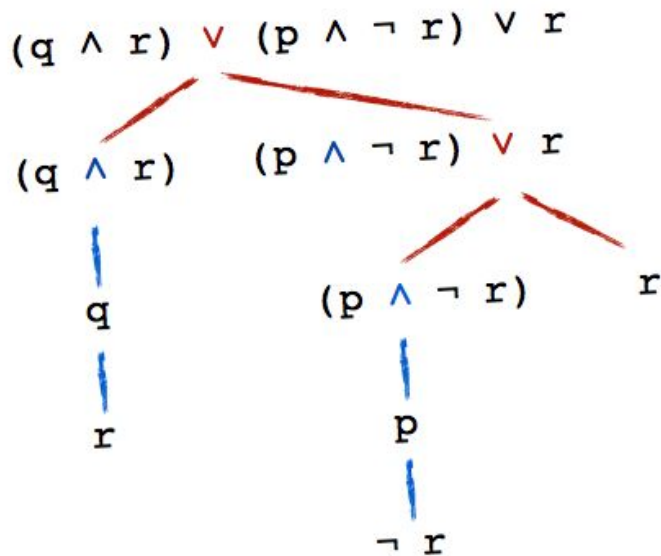
- For **automated reasoning**, we need syntactic algorithms to check the consistency of logical assertions
- The **method of analytical tableaux** is based on disjunctive normal form of logical expressions
 - invented by Dutch logician Evert Willem Beth in 1955 and simplified by Raymond Smullyan.
- **Basic Idea of Tableaux Algorithm:**
 - Proof algorithm (decision procedure) to check the consistency of a logical formula by inferring that its negation is a contradiction (***proof by refutation***).



Raymond Merrill Smullyan

Beth, Evert W., 1955. "Semantic entailment and formal derivability", *Mededelingen van de Koninklijke Nederlandse Akademie van Wetenschappen, Afdeling Letterkunde*, N.R. Vol 18, no 13, 1955, pp 309–342.

Tableaux Algorithm for PL



- (1) Construct **Decision Tree**, where each node is marked with a logical formula.
- A **path from the root to a leaf** is the **conjunction** of all formulas represented within the nodes of the path;
 - a **branch of the path** represents a **disjunction**.

Tableaux Algorithm for PL

1. Construct **Decision Tree**, where each node is marked with a logical formula.
 - A **path from the root to a leaf** is the **conjunction** of all formulas represented within the nodes of the path;
 - a **branch of the path** represents a **disjunction**.
2. The tree is created by successive application of the **Tableaux Extension Rules** (which are selected non deterministically).
3. A **path** in the Tableaux is **closed**,
 - if along the path as well X as $\neg X$ for a formula X occurs (X doesn't have to be atomic) or
 - if **false** occurs .
4. A tableaux is **fully expanded**, if no more extension rules can be applied.
5. A tableaux is called **closed**, if all its paths are closed.
6. A **Tableaux Proof** for a formula X is a closed tableaux for $\neg X$.

Tableaux Extension Rules for PL

- for PL Variables and Truth Values:

$$\frac{\neg\neg X}{X} \quad \frac{\neg T}{F} \quad \frac{\neg F}{T}$$

- for *conjunctive formulas* (α -Rules):

$$\begin{array}{l} \underline{\alpha} \\ \alpha_1 \\ \alpha_2 \end{array} \quad \frac{X \wedge Y}{\begin{array}{l} X \\ Y \end{array}} \quad \frac{\neg (X \vee Y)}{\begin{array}{l} \neg X \\ \neg Y \end{array}} \quad \frac{\neg (X \rightarrow Y)}{\begin{array}{l} X \\ \neg Y \end{array}}$$

- for *disjunctive formulas* (β -Rules):

$$\begin{array}{l} \underline{\beta} \\ \beta_1 \mid \beta_2 \end{array} \quad \frac{X \vee Y}{X \mid Y} \quad \frac{\neg (X \wedge Y)}{\neg X \mid \neg Y} \quad \frac{X \rightarrow Y}{\neg X \mid Y}$$

Tableaux Algorithm (PL) - Example 1

proof: $((q \wedge r) \rightarrow (\neg q \vee r))$

α -Rule

$\neg(X \rightarrow Y)$

X

$\neg Y$

1. $\neg((q \wedge r) \rightarrow (\neg q \vee r))$

2. $\alpha, 1: (q \wedge r)$

3. $\alpha, 1: \neg(\neg q \vee r) = q \wedge \neg r$

Tableaux Algorithm (PL) - Example 1

proof: $((q \wedge r) \rightarrow (\neg q \vee r))$

1. $\neg((q \wedge r) \rightarrow (\neg q \vee r))$

2. $\alpha, 1: (q \wedge r)$

3. $\alpha, 1: \neg(\neg q \vee r) = q \wedge \neg r$

4. $\alpha, 2: q$

5. $\alpha, 2: r$

6. $\alpha, 3: q$

7. $\alpha, 3: \neg r$

- path is closed
- tableaux is closed

α -Rule

x	$\wedge$	y
x		
y		

Tableaux Algorithm (PL) - Example 2

proof: $(p \rightarrow (q \rightarrow r)) \rightarrow ((p \rightarrow q) \rightarrow (p \rightarrow r))$

α -Rule

$\neg(x \rightarrow y)$

x

$\neg y$

1. $\neg((p \rightarrow (q \rightarrow r)) \rightarrow ((p \rightarrow q) \rightarrow (p \rightarrow r)))$
2. $\alpha, 1: (p \rightarrow (q \rightarrow r))$
3. $\alpha, 1: \neg((p \rightarrow q) \rightarrow (p \rightarrow r))$
4. $\alpha, 3: (p \rightarrow q)$
5. $\alpha, 3: \neg(p \rightarrow r)$
6. $\alpha, 5: p$
7. $\alpha, 5: \neg r$

Tableaux Algorithm (PL) - Example 2

proof: $(p \rightarrow (q \rightarrow r)) \rightarrow ((p \rightarrow q) \rightarrow (p \rightarrow r))$

1. $\neg((p \rightarrow (q \rightarrow r)) \rightarrow ((p \rightarrow q) \rightarrow (p \rightarrow r)))$
2. $\alpha, 1: (p \rightarrow (q \rightarrow r))$
3. $\alpha, 1: \neg((p \rightarrow q) \rightarrow (p \rightarrow r))$
4. $\alpha, 3: (p \rightarrow q)$
5. $\alpha, 3: \neg(p \rightarrow r)$
6. $\alpha, 5: p$
7. $\alpha, 5: \neg r$
8. $\beta, 2: \neg p$ | 9. $\beta, 2: (q \rightarrow r)$
10. $\beta, 9: \neg q$ | 11. $\beta, 2: r$
12. $\beta, 4: \neg p$ | 13. $\beta, 4: q$

β -Rule

$X \rightarrow Y$
 $\neg X \mid Y$

Tableaux Algorithm Extension for FOL

- As for propositional logic - $\mathbb{X}$ and $\mathbb{Y}$ stand for arbitrary (FOL) formulas
- Additional Rules for quantified formulas : $\frac{\gamma}{\gamma[t]}$ $\frac{\delta}{\delta[c]}$
- γ for **universally quantified** formulas, δ **existentially quantified** formulas, with:

γ	$\gamma[t]$
$\forall x. \Phi$	$\Phi[x \leftarrow t]$
$\neg \exists x. \Phi$	$\neg \Phi[x \leftarrow t]$

δ	$\delta[c]$
$\exists x. \Phi$	$\Phi[x \leftarrow c]$
$\neg \forall x. \Phi$	$\neg \Phi[x \leftarrow c]$

- t is an arbitrary ground term
(i.e. doesn't contain variables that are bound in Φ),
- c is a „new“ constant

Tableaux Algorithm (FOL) - Example 3

γ	$\gamma[t]$
$\forall x. \Phi$	$\Phi[x \leftarrow t]$
$\neg \exists x. \Phi$	$\neg \Phi[x \leftarrow t]$
δ	$\delta[c]$
$\exists x. \Phi$	$\Phi[x \leftarrow c]$
$\neg \forall x. \Phi$	$\neg \Phi[x \leftarrow c]$

proof: $(\forall x. (P(x) \vee Q(x))) \rightarrow ((\exists x. P(x)) \vee (\forall x. Q(x)))$

1. $\neg((\forall x. (P(x) \vee Q(x))) \rightarrow ((\exists x. P(x)) \vee (\forall x. Q(x))))$
2. $\alpha, 1: (\forall x. (P(x) \vee Q(x)))$
3. $\alpha, 1: \neg((\exists x. P(x)) \vee (\forall x. Q(x)))$
4. $\alpha, 3: \neg(\exists x. P(x))$
5. $\alpha, 3: \neg(\forall x. Q(x))$
6. $\delta, 5: \neg Q(c)$
7. $\gamma, 4: \neg P(c)$
8. $\gamma, 2: P(c) \vee Q(c)$
9. $\beta, 8: P(c) \quad | \quad 10. \beta, 8: Q(c)$



07: Description Logics

OpenHPI - Course Knowledge Engineering with Semantic Web Technologies

Lecture 3: Ontologies and Logic