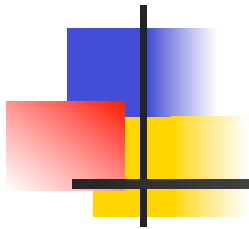


Propositional Logic Proof Theory



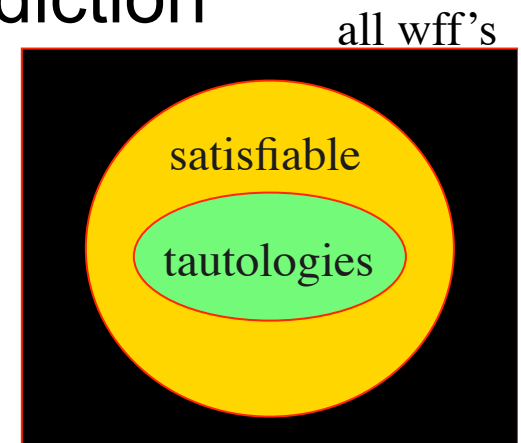
Michael Leuschel

Softwaretechnik und Programmiersprachen



Recap

- Wff's: atomic propositions, \neg , \wedge , \vee , \rightarrow
- Interpretations:
 - propositions \mapsto {true,false}
 - wff's \mapsto {true,false} by truth-tables
- Models
- Satisfiable wff, tautology, contradiction
- Logical equivalence \equiv
- Logical consequence \Rightarrow





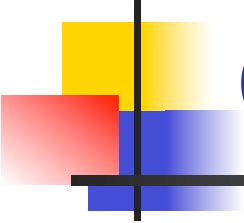
Additions/Clarifications

- New notations:
 - $\alpha \Leftrightarrow \beta$ stands for $(\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$
 - $\alpha \leftarrow \beta$ stands for $(\beta \rightarrow \alpha)$
- Difference between \Leftrightarrow and \equiv
 - $p \Leftrightarrow q$ is a satisfiable wff
 - $p \equiv q$ is not a wff and is simply not true
- Difference between \rightarrow and \Rightarrow
 - $p \rightarrow \neg p$ is a satisfiable wff
 - $p \Rightarrow \neg p$ is not a wff and is simply not true
- \rightarrow , \Leftrightarrow are connectives while \equiv , \Rightarrow talk about wff's



Consequence and Meaning

- Meaning of a logical theory S :
all formulas L_i such that $S \Rightarrow L_i$
 - S = knowledge about the possible states of the world
 L_i = things that must be true in all possible states
 - S = description of our beliefs
 L_i = things we also have to believe
 - S = logic program describing a problem
solutions to the problem contained in the L_i 's



Determining Logical Consequences

- $S \Rightarrow L$?
 - Try out all possible interpretations v_i
 - Check whether v_i is a model of S (truth tables)
 - If it is, check whether v_i is also a model of L
 - If all models of S are also models of L : OK

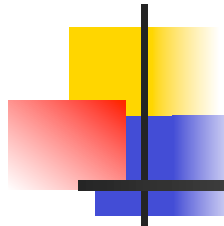


Logic Teaser (adapted)

- Knights: always tell the truth, Knaves: always lie
- Three persons: knights or knaves
- A says: “B is a knave or C is a knave” $A \Leftrightarrow \neg B \vee \neg C$
- B says “A is a knight” $B \Leftrightarrow A$

Theory $\Rightarrow A \wedge B$?

A	B	C	$\neg B \vee \neg C$	Theory	$A \wedge B$
True	True	True	False	False	True
True	True	False	True		
True	False	True			
True	False	False			
False	True	True			
False	True	False			
False	False	True			
False	False	False			



Logic Teaser solution

- Knights: always tell the truth, Knaves: always lie
- Three persons: knights or knaves
- A says: “B is a knave or C is a knave” $A \Leftrightarrow \neg B \vee \neg C$
- B says “A is a knight” $B \Leftrightarrow A$

$$\{A \Leftrightarrow \neg B \vee \neg C, B \Leftrightarrow A\} \Rightarrow A \wedge B$$

A	B	C	$\neg B \vee \neg C$	Theory	$A \wedge B$
<u>True</u>	True	True	<u>False</u>	False	True
True	True	False	True	True	True
True	False	True	True	False	False
True	False	False	True	False	False
<u>False</u>	<u>True</u>	True	False	False	False
<u>False</u>	True	False	<u>True</u>	False	False
<u>False</u>	False	True	<u>True</u>	False	False
<u>False</u>	False	False	<u>True</u>	False	False

Determining Logical Consequences II

- $S \Rightarrow L$?
 - Try out all possible interpretations v_i
 - Check whether v_i is a model of S (truth tables)
 - If it is, check whether v_i is also a model of L
 - If all models of S are also models of L : OK
 - Cumbersome 😞
 - Moving to Predicate Logic:
 - infinite models 😞, infinite number of models 😞
- How to do it without constructing all models ?



Recap: What is a logic ?

1. **Language** of expressions/formulas ✓

2. **Proof Theory**

Axioms = things which are true

Inference Rules to derive new theorems from existing ones (**truth preserving inference**)

3. **Model Theory** ✓

Mapping from formulas to “real” objects



Proof by Refutation

- Theorem: $S \Rightarrow L$ iff $S \wedge \neg L$ is inconsistent
- Idea to prove $S \Rightarrow L$:
 - Start from $S \wedge \neg L$
 - Apply **inconsistency preserving** rules
 - Try to arrive at an “obvious” inconsistency
 - $p \wedge \neg p$
 - true \rightarrow false
 - false
- “Reductio ad absurdum”



Proof by refutation (cont'd)

$S \Rightarrow L$ iff $S \wedge \neg L$ is inconsistent

A	B	C	Theory S	$L = A \wedge B$	$S \wedge \neg L$
<u>True</u>	True	True	False	True	False
True	True	False	True	True	False
True	False	True	False	False	False
True	False	False	False	False	False
<u>False</u>	<u>True</u>	True	False	False	False
<u>False</u>	True	False	False	False	False
<u>False</u>	False	True	False	False	False
<u>False</u>	False	False	False	False	False

$$S = \{A \Leftrightarrow \neg B \vee \neg C, B \Leftrightarrow A\}$$



Resolution

Resolvent

- From $p \vee \alpha$ and $\neg p \vee \beta$ derive $\alpha \vee \beta$
 - p is any atomic proposition
 - α, β any formula, can also be empty
- Note: disjunction is associative and commutative:
 - $s \vee (\neg t \vee r) \equiv (s \vee \neg t) \vee r \equiv \neg t \vee (s \vee r)$



Resolution Examples

- From $p \vee \alpha$ and $\neg p \vee \beta$ derive $\alpha \vee \beta$

$p \vee \alpha$	$\neg p \vee \beta$	$\alpha \vee \beta$
p \vee q	\neg p \vee r	q \vee r
t \vee q	s \vee \neg t \vee r	q \vee s \vee r
rains	wet \vee \neg rains	wet
\neg rains \vee wet	\neg wet	\neg rains



Resolution II

- From $p \vee \alpha$ and $\neg p \vee \beta$ derive $\alpha \vee \beta$
 - If both α and β are empty:
 - p and $\neg p$ hold
 - we have found a contradiction
 - Denoted by \square (empty formula, false)
 - Theorems:
 - $\{p \vee \alpha, \neg p \vee \beta\} \Rightarrow \alpha \vee \beta$
 - If $S \cup \{p \vee \alpha, \neg p \vee \beta\}$ is inconsistent if-and-only-if $S \cup \{p \vee \alpha, \neg p \vee \beta, \alpha \vee \beta\}$ is inconsistent
 - resolution is “inconsistency preserving”
 - We can use it for “proof by refutation”!
- Notation: $\{F_1, F_2, \dots, F_n\}$ shorthand for $F_1 \wedge F_2 \wedge \dots \wedge F_n$

A Proof by Refut

```
/* Prolog example */  
p.  
q.  
  
?- p, q.
```

- $\{p, q\} \Rightarrow p \wedge q$?
 1. Add negation of $p \wedge q$
- $\{p, q, \neg(p \wedge q)\}$ \rightarrow
 2. No disjunction:
 \Rightarrow Resolution cannot be applied
 \Rightarrow Transform into equivalent wff with \vee
- $\{p, q, \neg p \vee \neg q\}$ \leftarrow
 3. Resolve p and $\neg p \vee \neg q$
- $\{p, q, \neg p \vee \neg q, \neg q\}$
 4. Resolve q and $\neg q$
- $\{p, q, \neg p \vee \neg q, \neg q, \text{false}\}$

\searrow Yes: $\{p, q\} \Rightarrow p \wedge q$



Conjunctive Normal Form (CNF)

- Literal = p or $\neg p$ where p is an atomic proposition (p : positive, $\neg p$: negative literal)
- Clause = disjunction of literals
 - Examples: $\neg p \vee q$, $\text{wet} \vee \neg \text{rains} \vee \neg \text{outside}$
 - Horn clause if at most 1 positive literal
 - Denial if no positive literal
- CNF: set of clauses = conjunction of clauses

- Good for resolution!



Prolog & CNF

- Observe: $\neg p \vee q \equiv p \rightarrow q$
- Prolog
 - `wet :- rains, outside.`
 - This represent the logical formula:

wet \leftarrow rains \wedge outside

- Which is logically equivalent to

wet \vee \neg rains \vee \neg outside

- Every Prolog rule can be viewed as a Horn clause!



Converting formulas into CNF

- *Eliminate Implications:*
 - For this you just have to replace
 - $(\alpha \rightarrow \beta)$ by $(\neg\alpha \vee \beta)$
- *Move negations down to the atomic propositions:*
 - For this you have to replace:
 - 1. $(\neg(\alpha \wedge \beta))$ by $(\neg\alpha \vee \neg\beta)$
 - 2. $(\neg(\alpha \vee \beta))$ by $(\neg\alpha \wedge \neg\beta)$
 - 3. $\neg\neg\alpha$ by α



Converting formulas into CNF

- *Move the disjunctions (\vee) down to the literals:*
- For this you should replace
 - $\alpha \vee (\beta \wedge \gamma)$ by $(\alpha \vee \beta) \wedge (\alpha \vee \gamma)$
 - The following, more general rewrite can also be applied:
 - $\alpha \vee (\beta_1 \wedge \beta_2 \wedge \dots \wedge \beta_n)$ by $(\alpha \vee \beta_1) \wedge (\alpha \vee \beta_2) \dots (\alpha \vee \beta_n)$
 - Note that \wedge is associative, meaning that $((\alpha \wedge \beta) \wedge \gamma)$ is logically equivalent to $(\alpha \wedge (\beta \wedge \gamma))$. One therefore often writes $(\alpha \wedge \beta \wedge \gamma)$. The same holds for \vee .
 - This also means that we can replace, e.g.,
 - $(\beta_1 \wedge \beta_2 \wedge \dots \wedge \beta_n) \vee \alpha$ by $(\beta_1 \vee \alpha) \wedge (\beta_2 \vee \alpha) \dots (\beta_n \vee \alpha)$



Converting formulas into CNF

- *Eliminate the conjunctions:*
 - Actually, all you have to do is simply write each part of a conjunction as a separate wff.
 - For example $(p \vee q) \wedge (q \vee r)$ is turned into a set consisting of two wff's: $\{ (p \vee q), (q \vee r) \}$
- *Simplify clauses and eliminate redundant clauses:*
 - Remove extra literals which occur more than once in a clause.
 - For example $(p \vee \neg q \vee p)$ can be simplified into $(p \vee \neg q)$.
 - Remove the clauses that contain a proposition and its negation.
 - For example $(p \vee \neg p \vee q)$ can be removed (the clause is always true and does not add any information).



Exercise

- Do our logic teaser by resolution



Gantō's Ax

- One day Tokusan told his student Gantō, *“I have two monks who have been here for many years. Go and examine them.”*
- Gantō picked up an ax, saying, *“If you say a word I will cut off your heads; and if you do not say a word, I will also cut off your heads.”*

Quelle: D. Hofstadter; Gödel, Escher, Bach.

See also http://en.wikipedia.org/wiki/Ganto's_Ax



Gantõ's Ax Continued

- $G = \{\text{cut_head} \leftarrow \text{say_word}, \text{cut_head} \leftarrow \neg \text{say_word}\}$
- $G \Rightarrow \{\text{cut_head}\}$?
 - 1. $\text{cut_head} \vee \text{say_word}$
 - 2. $\text{cut_head} \vee \neg \text{say_word}$
 - 3. $\neg \text{cut_head}$
 - 4. cut_head (by resolving 1,2)
 - 5. \square (by resolving 3,4)
- Yes: $G \Rightarrow \{\text{cut_head}\}$



Gantō's Ax Finished

- Both monks continued their meditation as if he had not spoken. Gantō dropped the ax and said, “*you are true Zen students.*”
- He returned to Tokusan and related the incident. “*I see your side well,*” Tokusan agreed, “*but tell me, how is their side?*”
- “*Tōzan may admit them,*” replied Gantō, “*but they should not be admitted under Tokusan.*”



What to know for the exam:

- Proof by refutation
- Resolution
- Conjunctive Normal Form