Introduction to Automated Theorem Proving

Tudor Jebelean

June 5, 2018

Contents

Chapter 1

Propositional Logic

1.1 Syntax

Propositional logic is a mathematical model of reasoning with statements composed logically from elementary statements (or *propositions*). The only relevant characteristic of an elementary proposition (like "It rains.") is that it can be true or false. Therefore such a an elementary statement is denoted by a single symbol (*propositional variable*) about which we only know that it can be true or false.

Examples:

$$\underbrace{\text{"It rains."}}_{\text{A}} \qquad \underbrace{\text{"It is sunny."}}_{\text{B}}$$

Using the propositional variables as basic building blocks, we can compose arbitrary complex *formulae* by using the *logical connectives* ($\neg \wedge \vee \Rightarrow \Leftrightarrow$ and possibly other).

Examples:

$$\underbrace{\text{"If it rains then it is not sunny.}}_{A \Rightarrow \neg B}$$

The syntax of propositional logic consists in the definition of the set of all propositional logic formulae, or the language of propositional logic formulae, which will contain formulae like:

$$\mathscr{L} \text{ :Language with "words" like } \begin{cases} A \wedge B \\ A \wedge \neg B \\ (\neg A \wedge B) \Leftrightarrow (A \Rightarrow B) \\ A \wedge \neg A \end{cases}$$

The language $\mathscr{L}$ is defined over a certain set Σ of symbols: the parantheses, the logical connectives, the logical constants, and an infinite set Θ of propositional variables.

$$\text{Set of "symbols": "alphabet"} \quad \Sigma = \{\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow\} \cup \{\mathbb{T}, \mathbb{F}\} \cup \Theta$$

Note: Θ is the set of propositional variables. for instance this could be $\{A, B, C, P, Q, \ldots, A_1, A_2, \ldots\}$. This set Θ is infinite, but enumerable.

Inductive Definition

(0) $\mathbb{T}, \mathbb{F} \in \mathscr{L}$ or one can also write $\{\mathbb{T}, \mathbb{F}\} \subset \mathscr{L}$

(1) if $\underbrace{\vartheta \in \Theta}_{\text{"variable"}}$, then $\underbrace{\vartheta \in \mathscr{L}}_{\text{"word", logical formula}}$ (a propositional variable ϑ "is" also a logical formula)

(2) if $\varphi, \psi \in \mathscr{L}$, then $\underbrace{\underbrace{\neg\varphi}, \underbrace{\varphi\wedge\psi}, \underbrace{\varphi\vee\psi}, \underbrace{\varphi\Rightarrow\psi}, \underbrace{\varphi\Leftrightarrow\psi}, \underbrace{(\varphi)} \in \mathscr{L}}_{\text{"are also words in the language"}}$

(3) These are all the formulae.

Note that Σ^* also has the properties **(0), (1)** and **(2)**. However, **[3]** means that $\mathscr{L}$ is the smallest set having these properties. This allows us to use the structural induction principle in order to prove properties of formulae, and also to define functions over formulae.

Induction principle for proving: In order to prove the property $P[\varphi]$ for all formulae φ:

1. prove $P[\mathbb{T}], P[\mathbb{F}]$ and for all propositional variables $\vartheta : P[\vartheta]$;
2. prove for all propositional formulae $\varphi, \psi : P[\neg\varphi], P[\varphi\wedge\psi], P[\varphi\vee\psi], P[\varphi\Rightarrow\psi], P[\varphi\Leftrightarrow\psi], P[(\varphi)]$ by assuming that $P[\varphi], P[\psi]$ hold.

Example. Prove that every propositional formula is finite (has a finite number of symbols).

1. Holds for the base case, since there is only one symbol.
2. If φ, ψ are finite, then any formula constructed as in the definition will also be finite.

Induction principle for function definitions: In order to define a function $f[\varphi]$ for all formulae φ

1. define $f[\mathbb{T}], f[\mathbb{F}]$ and for all propositional variables $\vartheta : f[\vartheta]$;
2. define for all propositional formulae $\varphi, \psi : f[\neg\varphi], f[\varphi\wedge\psi], f[\varphi\vee\psi], f[\varphi\Rightarrow\psi], f[\varphi\Leftrightarrow\psi], f[(\varphi)]$ by using $f[\varphi], f[\psi]$.

Example. Define the function which constructs the reversed Polish representation of a propositional formula[1].

1. $f[\varphi] = \varphi$ if φ is $\mathbb{T}, \mathbb{F}$ or a propositional variable.
2. For arbitrary formulae φ, ψ: $f[\neg\varphi] = f[\varphi]\neg$, $f[\varphi\wedge\psi] = f[\varphi]f[\psi]\wedge$, $f[\varphi\vee\psi] = f[\varphi]f[\psi]\vee$, $f[\varphi\Rightarrow\psi] = f[\varphi]f[\psi]\Rightarrow$, $f[\varphi\Leftrightarrow\psi] = f[\varphi]f[\psi]\Leftrightarrow$, $f[(\varphi)] = [\varphi]$

Atoms and literals. Formulae consisting in a single propositional variable are called *"atoms"*. Formulae consisting in an atom or a negated atom are called *"literals"*.

Notation: Sometimes we will denote $\neg\varphi$ by $\overline{\varphi}$. Also, if L is a literal, we will denote by $\overline{L}$ the opposite of L (that is $\overline{A}$ if L is A, and A if L is $\overline{A}$).

[1] The reversed Polish representation does not use parantheses and it is easy to parse and evaluate.

Exercise: Formulate the grammar for the language of propositional logic.

$$P = \begin{cases} W \to \mathbb{T} \mid \mathbb{F} \mid A \mid B \mid C \\ W \to (\neg W) \mid (W \wedge W) \mid \dots \end{cases}$$

$$\text{Grammar } G = (\underbrace{\Sigma}_{\text{"alphabet"}}, \underbrace{\Sigma_N}_{\substack{\text{"nonterminal} \\ \text{symbols"}}}, \underbrace{S}_{\substack{\text{"nonterminal} \\ \text{start symbol"}}}, \underbrace{P}_{\substack{\text{set of} \\ \text{productions}}})$$

$$\begin{aligned} \Sigma &= \{\mathbb{T}, \mathbb{F}\} \cup \Theta \cup \{(,), \neg, \vee, \wedge, \Rightarrow, \Leftrightarrow\} \\ \Sigma_N &= \{W\} \\ S &= \Sigma_N \end{aligned}$$

1.2 Semantics

Example: Intuitively, the meaning of "$A \wedge B$" is that "this is only true if both A and B are true".

$f_{A\wedge B}$	$\mathbb{T}$	$\mathbb{F}$
$\mathbb{T}$	$\mathbb{T}$	$\mathbb{F}$
$\mathbb{F}$	$\mathbb{F}$	$\mathbb{F}$

Table 1.1: Semantic value of $A \wedge B$.

The semantic value (or the meaning) of the formula $A \wedge B$ is the function $f_{A\wedge B} : \mathscr{I}_{\{A,B\}} \to \{\mathbb{T}, \mathbb{F}\}$, where $\mathscr{I}_{\{A,B\}} = \{I : \{A, B\} \to \{\mathbb{T}, \mathbb{F}\}\}$ is the set of all assignments of truth values to the variables A, B.

I is called an *"interpretation"* for the formula $A \wedge B$. $\mathscr{I}_{\{A,B\}}$ is the *"set of interpretations"* for the formula $A \wedge B$.

As syntax is defined as the set $\mathcal{L}$ of all correct formulae, the semantics is defined as the set $\mathcal{S}$ of all possible semantic values:

$$\mathcal{S} = \{\mathscr{I}_V | V \subseteq \Theta\}.$$

The "semantic evaluation function" associates each formula φ from $\mathcal{L}$ to its semantic value f_φ from $\mathcal{S}$. If we denote by $\mathrm{Var}(\varphi)$ the set of propositional variables occuring in φ, then:

$$f_\varphi : \mathcal{I}_{\mathrm{Var}(\varphi)} \to \{\mathbb{T}, F\}, \quad f_\varphi(I) = \langle\varphi\rangle_I,$$

where $\langle\varphi\rangle_I$ is the *"truth value of φ under the interpretation I"*.

$\langle\varphi\rangle_I$ (the truth evaluation of a formula φ under the interpretation I) is defined inductively on the structure of formulae:

$$\begin{aligned} \langle\mathbb{F}\rangle_I &= \mathbb{F} \\ \langle\mathbb{T}\rangle_I &= \mathbb{T} \\ \langle v\rangle_I &= I(\vartheta), \quad \text{if } v \in \Theta \end{aligned}$$

$$\begin{aligned} \langle\neg\varphi\rangle_I &= \mathscr{B}_\neg(\langle\varphi\rangle_I) \\ \langle\varphi \vee \psi\rangle_I &= \mathscr{B}_\vee(\langle\varphi\rangle_I, \langle\psi\rangle_I) \\ \langle\varphi\wedge\psi\rangle_I &= \mathscr{B}_\wedge(\langle\varphi\rangle_I, \langle\psi\rangle_I) \\ &\dots \end{aligned}$$

The functions $\mathscr{B}_\neg, \mathscr{B}_\vee, \mathscr{B}_\wedge, \ldots$ (boolean evaluation functions) are defined explicitely by truth tables for each logical connective, and they can be seen as the semantic values of the logical connectives.

	$\mathscr{B}_\neg$
$\mathbb{T}$	$\mathbb{F}$
$\mathbb{F}$	$\mathbb{T}$

$\mathscr{B}_\wedge$	$\mathbb{T}$	$\mathbb{F}$
$\mathbb{T}$	$\mathbb{T}$	$\mathbb{F}$
$\mathbb{F}$	$\mathbb{F}$	$\mathbb{F}$

$\mathscr{B}_\vee$	$\mathbb{T}$	$\mathbb{F}$
$\mathbb{T}$	$\mathbb{T}$	$\mathbb{T}$
$\mathbb{F}$	$\mathbb{T}$	$\mathbb{F}$

$\mathscr{B}_\Rightarrow$	$\mathbb{T}$	$\mathbb{F}$
$\mathbb{T}$	$\mathbb{T}$	$\mathbb{F}$
$\mathbb{F}$	$\mathbb{T}$	$\mathbb{T}$

$\mathscr{B}_\Leftrightarrow$	$\mathbb{T}$	$\mathbb{F}$
$\mathbb{T}$	$\mathbb{T}$	$\mathbb{F}$
$\mathbb{F}$	$\mathbb{F}$	$\mathbb{T}$

Table 1.2: The semantics of logical connectives

Example

$$
\begin{aligned}
\langle (A \wedge (A \Rightarrow B)) \Rightarrow B \rangle_I &= \mathscr{B}_\Rightarrow \left(\langle (A \wedge (A \Rightarrow B)) \rangle_I , \langle B \rangle_I \right) \\
&= \mathscr{B}_\Rightarrow \left(\mathscr{B}_\wedge \left(\langle A \rangle_I , \langle (A \Rightarrow B) \rangle_I \right) , \langle B \rangle_I \right) \\
&= \mathscr{B}_\Rightarrow \left(\mathscr{B}_\wedge \left(\langle A \rangle_I , \mathscr{B}_\Rightarrow \left(\langle A \rangle_I , \langle B \rangle_I \right) \right) , \langle B \rangle_I \right) \\
&= \mathscr{B}_\Rightarrow \left(\underbrace{\mathscr{B}_\wedge \left(\mathbb{T}, \underbrace{\mathscr{B}_\Rightarrow (\mathbb{T}, \mathbb{F})}_{\mathbb{F}} \right)}_{\mathbb{F}} , \mathbb{F} \right) \\
&= \mathbb{T}
\end{aligned}
$$

Model, validity, satisfiability. If $\langle \varphi \rangle_I = \mathbb{T}$, then we say "$I$ satisfies φ" or "I is a model of φ".

If $\Big($for any I, $f_\varphi(I) = \mathbb{T}\Big)$, then we say *"$\varphi$ is valid"* (otherwise it is *"invalid"*)

If $\Big($for any I, $f_\varphi(I) = \mathbb{F}\Big)$, then we say *"$\varphi$ is unsatisfiable"* (otherwise it is *"satisfiable"*)

Example. The formula $A \wedge \neg A$ is ~~"always false"~~: The correct characterization for this is *"unsatisfiable"*: for all interpretations I, $f_{A \wedge \neg A}(I) = \mathbb{F}$.

The formula $(A \Rightarrow (A \Rightarrow B)) \Rightarrow B$ is ~~"always true"~~. Correct is to say that it is *"valid"*: for all $I, \langle \ldots \rangle_I = \mathbb{T}$.

The following formula is *"invalid", but "satisfiable"*: $A \wedge B$ (its truth value depends on I).

Logical consequence. We say *"the formula ψ is a logical consequence of the formulae $\varphi_1, \ldots, \varphi_n$"* (also denoted as $\varphi_1, \ldots, \varphi_n \models \psi$), if and only if:
for all I, whenever $\langle \varphi_1 \rangle_I = \ldots = \langle \varphi_n \rangle_I = \mathbb{T}$, then also $\langle \psi \rangle_I = \mathbb{T}$.

Two Basic Theorems. When working in mathematics, the typical shape of the theorems we want to prove is $\varphi_1, \ldots, \varphi_n \models \psi$. The following two theorems show how to reduce such a problem to establishing the validity, respectively the insatisfiability, of a certain formula. The former is called *"proof by refutation"* and corresponds in fact to the known proof technique of "proof by contradiction".

Theorem 1.
$\varphi_1, \ldots, \varphi_n \models \psi$ if and only if $(\varphi_1 \wedge \ldots \wedge \varphi_n) \Rightarrow \psi$ is valid.

Theorem 2.
$\varphi_1, \ldots, \varphi_n \models \psi$ if and only if $\varphi_1 \wedge \ldots \wedge \varphi_n \wedge \neg\psi$ is unsatisfiable.

Logical equivalence. We say *"φ is equivalent to ψ"*, (also denoted as $\varphi \equiv \psi$) if and only if $\varphi \models \psi$ and $\psi \models \varphi$.

1.3 Interesting equivalences

- Commutativity: $\begin{cases} \varphi \wedge \psi \equiv \psi \wedge \varphi \\ \text{(same with } \vee\text{)} \end{cases}$
- Associativity: $\begin{cases} \varphi_1 \wedge (\varphi_2 \wedge \varphi_3) \equiv (\varphi_1 \wedge \varphi_2) \wedge \varphi_3 \\ \text{(same with } \vee\text{)} \end{cases}$
- Idempotence: $\begin{cases} \varphi \wedge \varphi \equiv \varphi \\ \text{(same with } \vee\text{)} \end{cases}$

This can be extended for more complex expressions, for example:
$(A \vee ((A \vee B) \vee (C \vee B))) \vee C \equiv (A \vee B \vee C)$:

$$\begin{aligned} (((\vartheta_1 \vee \vartheta_2) \vee \vartheta_3) \vee \vartheta_4) \vee \vartheta_5 &\equiv \vartheta_1 \vee \vartheta_2 \vee \vartheta_3 \vee \vartheta_4 \vee \vartheta_5 \\ \text{this can also be denoted} &: \bigvee \{\vartheta_1, \vartheta_2, \vartheta_3, \vartheta_4, \vartheta_5\} \end{aligned}$$

- Properties of negation

$$\begin{aligned} \neg\neg\varphi &\equiv \varphi \\ \neg\varphi \vee \varphi &\equiv \mathbb{T} \\ \neg\varphi \wedge \varphi &\equiv \mathbb{F} \\ \neg(\varphi \vee \psi) &\equiv (\neg\varphi) \wedge (\neg\psi) \\ \neg(\varphi \wedge \psi) &\equiv (\neg\varphi) \vee (\neg\psi) \end{aligned}$$

- Distributivity

$$\begin{aligned} \varphi \wedge (\psi_1 \vee \psi_2) &\equiv (\varphi \wedge \psi_1) \vee (\varphi \wedge \psi_2) \\ \varphi \vee (\psi_1 \wedge \psi_2) &\equiv (\varphi \vee \psi_1) \wedge (\varphi \vee \psi_2) \end{aligned}$$

- Elimination of $\Rightarrow$ and $\Leftrightarrow$

$$\begin{aligned} \varphi \Rightarrow \psi &\equiv (\neg\varphi) \vee \psi \\ \varphi \Leftrightarrow \psi &\equiv (\varphi \Rightarrow \psi) \wedge (\psi \Rightarrow \varphi) \\ &\equiv (\neg\varphi \vee \psi) \wedge (\neg\psi \vee \varphi) \\ &\equiv (\varphi \wedge \psi) \vee (\neg\varphi \wedge \neg\psi) \end{aligned}$$

Example usage of rules:

$$\begin{aligned} A \vee B \vee \neg A &\equiv \\ (A \vee \neg A) \vee B &\equiv \\ \mathbb{T} \vee B &\equiv \mathbb{T} \end{aligned}$$

- Properties of truth constants

$$\mathbb{T} \vee \varphi \equiv \mathbb{T} \qquad \mathbb{F} \vee \varphi \equiv \varphi$$
$$\mathbb{T} \wedge \varphi \equiv \varphi \qquad \mathbb{F} \wedge \varphi \equiv \mathbb{F}$$
$$\neg\mathbb{T} \equiv \mathbb{F} \qquad \neg\mathbb{F} \equiv \mathbb{T}$$

$$\mathbb{T} \Rightarrow \varphi \equiv \varphi \qquad \mathbb{F} \Rightarrow \varphi \equiv \mathbb{T} \text{ (false implies anything)}$$
$$\text{(because } \mathbb{T} \Rightarrow \varphi \equiv (\neg\mathbb{T}) \vee \varphi \equiv \mathbb{F} \vee \varphi \equiv \varphi) \qquad \text{(because } \mathbb{F} \Rightarrow \varphi \equiv (\neg\mathbb{F}) \vee \varphi \equiv \mathbb{T} \vee \varphi \equiv \mathbb{T})$$

$$\varphi \Rightarrow \mathbb{T} \equiv \mathbb{T} \qquad \varphi \Rightarrow \mathbb{F} \equiv \neg\varphi$$

Exercise: Write the rules for the elimination of the truth constants for $\Leftrightarrow$ (in analogy to $\varphi \Rightarrow \mathbb{T} \equiv \mathbb{T}$ and $\varphi \Rightarrow \mathbb{F} \equiv \neg\varphi$)

Proofs of the equivalences For example, prove $\mathbb{T} \vee \varphi \equiv \mathbb{T}$

By definition:

$$\begin{array}{rcl} \text{For any } I: & & \text{(note that again, this only refers to} \\ & & \text{the relevant interpretations as explained above)} \\ f_{\mathbb{T}\vee\varphi}(I) & =^{?} & f_{\mathbb{T}}(I) \\ \langle \mathbb{T} \vee \varphi \rangle_I & =^{?} & \langle \mathbb{T} \rangle_I \\ \mathscr{B}_{\vee}(\langle \mathbb{T} \rangle_I, \langle \varphi \rangle_I) & =^{?} & \langle \mathbb{T} \rangle_I \\ & =_{\text{Yeah!}} & \mathbb{T} \end{array}$$

$\mathscr{B}_{\vee}$	$\mathbb{T}$	$\mathbb{F}$
$\mathbb{T}$	Either this case:$\mathbb{T}$	or this case:$\mathbb{T}$
$\mathbb{F}$	$\mathbb{T}$	$\mathbb{F}$

Proof for $\neg\neg\varphi \equiv \varphi$

$$\begin{array}{rcl} \langle \neg\neg\varphi \rangle_I & =^{?} & \langle \varphi \rangle_I \\ \langle \neg\neg\varphi \rangle_I & \equiv & \mathscr{B}_{\neg}(\langle \neg\varphi \rangle_I) \\ & \equiv & \mathscr{B}_{\neg}(\mathscr{B}_{\neg}(\langle \varphi \rangle_I)) \equiv^{!} \langle \varphi \rangle_I \end{array}$$

$$\mathscr{B}_{\neg}(\mathscr{B}_{\neg}(c)) \equiv^{!} c \quad \text{case} \quad \begin{cases} c = \mathbb{T} & \mathscr{B}_{\neg}(\mathscr{B}_{\neg}(\mathbb{T})) \equiv_{\text{Yeah!}} \mathbb{T} \\ c = \mathbb{F} & \mathscr{B}_{\neg}(\mathscr{B}_{\neg}(\mathbb{F})) \equiv_{\text{Yeah!}} \mathbb{F} \end{cases}$$

Another way to prove, is shown here with the example of proving $\varphi \wedge (\psi_1 \vee \psi_2) \equiv (\varphi \wedge \psi_1) \vee (\varphi \wedge \psi_2)$

$\langle\varphi\rangle_I$	$\langle\psi_1\rangle_I$	$\langle\psi_2\rangle_I$	$\psi_1 \vee \psi_2$	LHS	$\varphi \wedge \psi_1$	$\varphi \wedge \psi_2$	RHS
$\mathbb{T}$	$\mathbb{T}$	$\mathbb{T}$	$\mathbb{T}$	$\mathbb{T}$	$\mathbb{T}$	$\mathbb{T}$	$\mathbb{T}$
$\mathbb{T}$	$\mathbb{T}$	$\mathbb{F}$	$\mathbb{T}$	$\mathbb{T}$	$\mathbb{T}$	$\mathbb{F}$	$\mathbb{T}$
$\mathbb{T}$	$\mathbb{F}$	$\mathbb{T}$	...	...	...	...	...
$\mathbb{T}$	$\mathbb{F}$	$\mathbb{F}$	...	...	...	...	...
$\mathbb{F}$	$\mathbb{T}$	$\mathbb{T}$	...	...	...	...	...
$\mathbb{F}$	$\mathbb{T}$	$\mathbb{F}$	...	...	...	...	...
$\mathbb{F}$	$\mathbb{F}$	$\mathbb{T}$	...	...	...	...	...
$\mathbb{F}$	$\mathbb{F}$	$\mathbb{F}$	...	...	...	...	...

Note: LHS means Left Hand Side, RHS means Right Hand Side

Exercise: Complete this table.

1.4 Transformation rules

Equivalences can be used as transformation rules.

$$\text{Eliminate } \neg \begin{cases} \neg\neg\varphi \equiv \varphi \\ \neg\varphi \vee \varphi \equiv \mathbb{T} \\ \neg\varphi \wedge \varphi \equiv \mathbb{F} \end{cases}$$

$$\text{Push negation: } \begin{cases} \neg(\varphi \vee \psi) & \equiv & (\neg\varphi) \wedge (\neg\psi) \\ \neg(\varphi \wedge \psi) & \equiv & (\neg\varphi) \vee (\neg\psi) \end{cases}$$

By repeated application of this rule, any formula can be transformed such that the negation sign occurs only before atoms. Atoms and negated atoms are called **"literals"**. Atoms are called *positive literals* and negated atoms are called *negative literals*.

$$\text{Distributivity } \left\{ \varphi \vee (\psi_1 \wedge \psi_2) \underset{\rightarrow}{\equiv} (\varphi \vee \psi_1) \wedge (\varphi \vee \psi_2) \right.$$

Note: Every formula can be transformed to a *conjunction of disjunctions of literals* **!**

Conjunctive Normal Form (CNF): $\underline{(\ldots \vee \ldots \vee \ldots)} \wedge \ldots \wedge \underline{(\ldots \vee \ldots \vee \ldots)}$

Note: Each of these underlined *"disjunction of literals"* is called *"clause"*, so a formula in CNF is a *conjunction of clauses*.

$$\begin{aligned}
(A \wedge (A \Rightarrow B)) \Rightarrow B &\equiv \text{(Replace implications...)} \\
\neg(A \wedge (\neg A \vee B)) \vee B &\equiv \text{(Push Negation/"De Morgan"...)} \\
(\neg A \vee \neg(\neg A \vee B)) \vee B &\equiv \text{(Push Negation ...)} \\
(\neg A \vee (\neg\neg A \wedge \neg B)) \vee B &\equiv \text{(Distributivity} \rightarrow \text{Second solution below)} \\
(\underbrace{(\neg A \vee A)}_{\mathbb{T}} \wedge (\neg A \vee \neg B)) \vee B &\equiv \\
(\neg A \vee \underbrace{(\neg B \vee B)}_{\mathbb{T}}) &\equiv \mathbb{T}
\end{aligned}$$

Second solution:

$$\begin{aligned}
(\neg A \vee (\neg\neg A \wedge \neg B)) \vee B &\equiv \text{(Distributivity, other possiblity)} \\
(\neg A \vee ((A \vee B) \wedge \underbrace{(\neg B \vee B)}_{\mathbb{T}})) &\equiv \\
\underbrace{\underbrace{(\neg A \vee A)}_{\mathbb{T}} \vee B}_{\mathbb{T}} &\equiv \mathbb{T}
\end{aligned}$$

Since every formula can be transformed into a CNF formula, it is sufficient to find proof methods for the formulae in CNF.

1.5 The Resolution Principle

"Resolution inference rule":

$$\left.\begin{array}{l} C_1 : L \vee C_1' \\ C_2 : \overline{L} \vee C_2' \end{array}\right\} \quad \mapsto \quad \underbrace{C_1' \vee C_2'}_{\substack{\text{"resolvent of the} \\ \text{clauses } C_1, C_2\text{"}}}$$

For example:

$$\left.\begin{array}{l} A \vee B \vee C \\ \overline{A} \vee B \vee P \vee Q \end{array}\right\} \quad \mapsto \quad B \vee C \vee P \vee Q$$

Correctness of the resolution rule

$$L \vee C_1', L \vee C_2' \quad \models \quad C_1' \vee C_2'$$

Remark: When C_1' is missing, then the resolvent is C_2',
when C_2' is missing, then the resolvent is C_1',
when both C_1' and C_2' are missing, then the resolvent is $\mathbb{F}$.

Proof:

For an arbitrary interpretation I

$\langle L \vee C_1' \rangle_I = \mathbb{T} = \left\langle \overline{L} \vee C_2' \right\rangle_I$

By cases $\left\{ \begin{array}{ll} \langle L \rangle_I = \mathbb{T} : \left(\text{otherwise } \left\langle \overline{L} \vee C_2' \right\rangle_I = \mathbb{F}\right) & \left\langle C_2' \right\rangle_I = \mathbb{T} \\ \left\langle \overline{L} \right\rangle_I = \mathbb{T} : \left(\text{otherwise } \left\langle L \vee C_1' \right\rangle_I = \mathbb{F}\right) & \left\langle C_1' \right\rangle_I = \mathbb{T} \end{array} \right.$, thus in either case $\left\langle C_1' \vee C_2' \right\rangle_I = \mathbb{T}$

Remark.

$$\begin{array}{lcl} (L \vee C_1) \wedge (L \vee C_2) & \not\equiv & C_1' \vee C_2' \,! \\ (L \vee C_1) \wedge (L \vee C_2) & \equiv & (L \vee C_1) \wedge (L \vee C_2) \wedge (C_1' \vee C_2') \end{array}$$

"resolution principle" : resolution inference rule and resolution method

"resolution method" : apply the resolution inference rule until you reach the empty clause.

Theorem (Completeness of the resolution method).
If a set of clauses is unsatisfiable, then there exists a proof by resolution (that is: the empty clause can be obtained by repeated applications of resolution inference rule to the original clauses and to the new ones).

We illustrate the idea of the proof through and example.

Consider the set of clauses:

1. $\overline{A} \vee P$
2. $\overline{B} \vee P$
3. $A \vee B$
4. $\overline{P}$

We represent all possible interpretations using a so called *"semantic tree"*, in which every path corresponds to an interpretation:

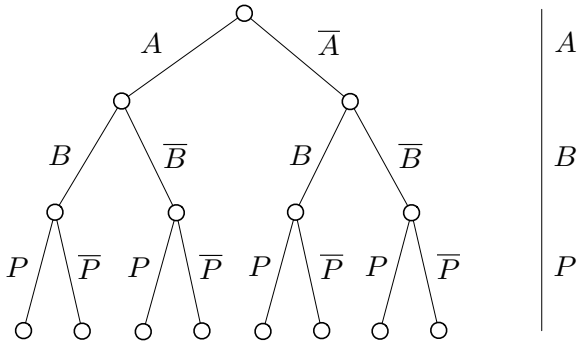


Figure 1.1: Semantic tree

The path $\langle A, \overline{B}, \overline{P} \rangle$ represents the interpretation $\begin{cases} A \leftarrow \mathbb{T} \\ B \leftarrow \mathbb{F} \\ P \leftarrow \mathbb{F} \end{cases}$

If the path leading to a node corresponds to an interpretation which falsifies a clause, then we say that *"the clause closes the node"*. For instance, the clause (3) closes the node with the path $\langle \overline{A}, \overline{B} \rangle$.

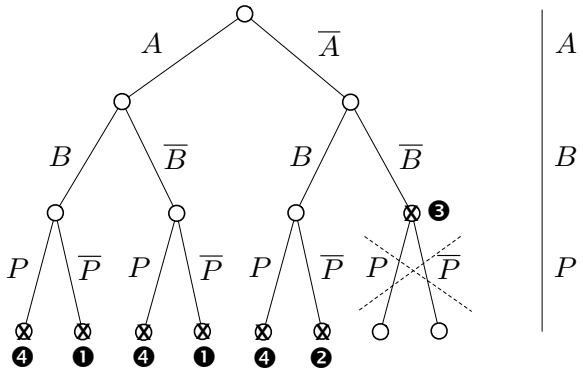


Figure 1.2: Semantic tree with closed nodes (1)

If every possible path has a closed node, than we say that *"the tree is closed"*. Note that a semantic tree corresponding to an unsatisfiable formula is allways closed, because a path whose nodes are all open defines an interpretation which satisfies the formula.

We convene to close every path as early as possible.

In a closed tree, there must be at least one node whose both sons are closed. If allways at least one of the sons is open, then one can construct an open path:

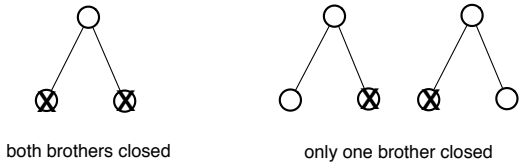


Figure 1.3: Both brothers closed versus one brother closed

When both brothers are closed, let us consider the clauses C_1 and C_2 which close them, and the atom L which corresponds to this branching. Since we close each path as early as possible, the atom L must occur in C_2 and the literal $\overline{L}$ must occur in C_1. Moreover, all the other literals present in C_1 and C_2 must be falsified on the path leading to the father:

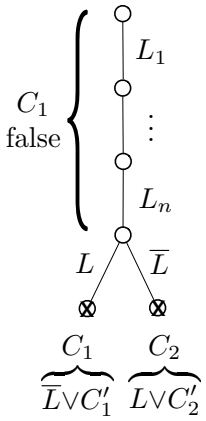


Figure 1.4: Consequence if both brothers are closed

Therefore, the clauses C_1 and C_2 are resolvable over L, and the resulting clause $C_1 \vee C_2$ falsifies the father or a node above it. By adding this resolvent to the set of clauses, we obtain a formula whose closed semantic tree is *smaller* then before.

Thus, inductively, the closed tree will be reduced until the root is closed, which means that the empty clause was produced.

In our example, one may apply this procedure as follows:

(1) $\overline{A} \vee P$

(2) $\overline{B} \vee P$

(3) $A \vee B$

(4) $\overline{P}$

(5) $\overline{A}$ from (1) and (4)

(6) $\overline{B}$ from (2) and (4)

(7) A from (3) and (6)

(8) $\bigvee\{\}$ from (5) and (7)

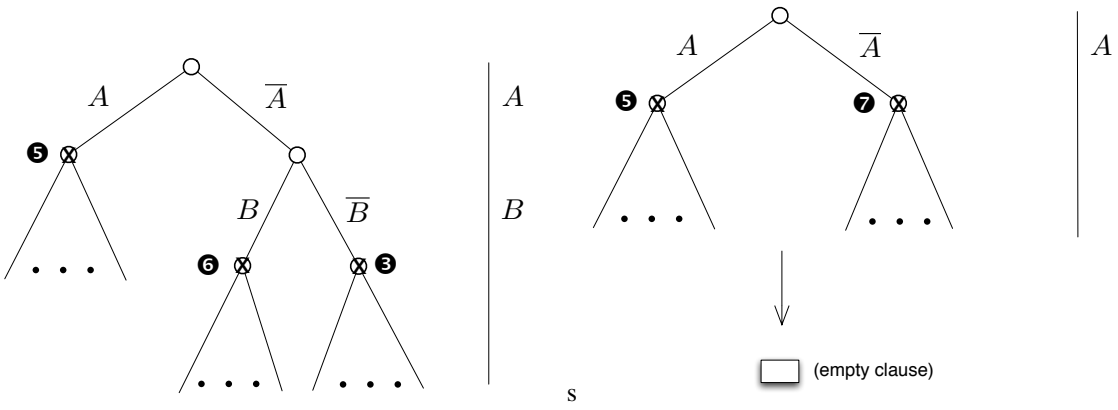


Figure 1.5: Semantic tree with closed nodes.

In conclusion, if we make all possible resolutions on an unsatisfiable set of clauses, then we will allways obtain the empty clause.

1.6 The DPLL Algorithm

The DPLL (Davis–Putnam–Logemann–Loveland) method shows the unsatisfiability (or alternatively finds satisfying interpretations) of a set of propositional clauses by alternating two types of steps:

- unit propagation,
- split.

Unit propagation is applied when the set of clauses contains a *"unit clause"* (that is a clause having only one literal, say L). In this case L is moved into a *"list of assignments"* and:

- all clauses containing L are removed (they cannot contribute better than L at obtaining the empty clause)
- $\overline{L}$ (the opposite of L) is removed from all clauses where it occurs (this is just resolution).

By the former one may produce the empty clause, and then the current branch of the proof is closed. The formula is unsatisfiable if this happens on all branches.

Note that by the former some new unit clauses may be produced. This process is repeated until there are no unit clauses anymore. If the set of clauses becomes empty, then the formula is satisfiable and the interpretation is given by the list of assignments which was constructed on the current branch.

Split is applied when there are no more unit clauses. One chooses one of the variables still present in the clause set, say A, and splits the proof into two branches: one for A and one for $\overline{A}$. These are taken as unit clauses and then one may apply unit propagation.

Additionally one can use the **pure literal** rule: if a propositional variable occurs only positively (or only negatively), then the clauses which contain this variable can be deleted – because they can be satisfied by assigning $\mathbb{T}$ to the respective literal, so the satisfiability of the whole set depends only on the remaining clauses.

Examples:

This is an unsatisfiable set of clauses. By split on P, one obtains the new set $\{Q, \overline{Q}\}$ on each branch, and then the empty clause.

$$\text{initial set} \quad \begin{cases} P \vee Q \\ \overline{P} \vee Q \\ P \vee \overline{Q} \\ \overline{P} \vee \overline{Q} \end{cases} \quad \begin{matrix} P \\ \nearrow \\ \searrow \\ \overline{P} \end{matrix} \quad \begin{cases} Q \\ \overline{Q} \end{cases} \to \square \quad \begin{cases} Q \\ \overline{Q} \end{cases} \to \square$$

This is a satisfiable set of clauses. First we apply unit propagation on R (just move R into the list of assignments). The by split on P one obtains the new set $\{Q\}$ on each branch, and then by unit propagation (just moving Q into the assignment list), one obtains the empty set.

$$\begin{cases} P \vee Q \\ \overline{P} \vee Q \\ R \end{cases} \quad \begin{matrix} \nearrow \\ \\ \\ \searrow \end{matrix} \quad \begin{matrix} P : Q \quad \nearrow \\ \qquad\quad \searrow \\ \\ \overline{P} : Q \quad \nearrow \\ \qquad\quad \searrow \end{matrix} \quad \begin{matrix} \emptyset \\ \emptyset \\ \emptyset \\ \emptyset \end{matrix}$$

$$\begin{array}{ccc} & R \leftarrow \mathbb{T} & \\ \swarrow & & \searrow \\ \mathbb{T} \to P & & \overline{P} \leftarrow \mathbb{T} \\ \downarrow & & \downarrow \\ \mathbb{T} \to Q & & Q \leftarrow \mathbb{T} \end{array}$$

Chapter 2

First-order Predicate Logic

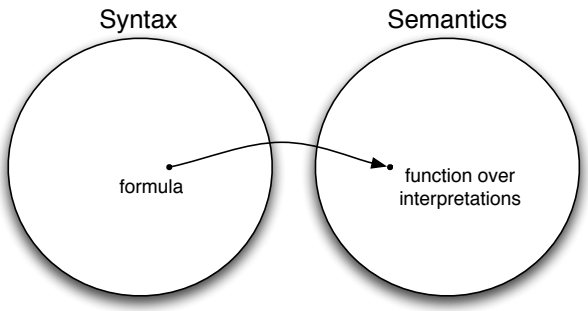


Figure 2.1: Syntax and Semantics

2.1 Definition of syntax

Formulae:

- terms
  - variables, constants
  - function symbols
- quantifiers
- logical connectives

$$\text{2 Languages} \quad \begin{cases} \text{Language of terms } \mathscr{L}_T \\ \text{Language of formulae } \mathscr{L}_F \end{cases}$$

Language of terms

$$\mathscr{L}_T \quad \begin{cases} \vartheta \in \Theta, c \in \mathscr{C} : \text{ are terms} \\ f \in \mathscr{F}, \quad t_1, \ldots, t_n : \text{ terms, then} \\ \qquad\qquad f(t_1, \ldots, t_n) \text{ is term} \\ \text{(these are all !)} \end{cases}$$

variables ↓ , constants ↓

- Θ , $\mathscr{C}$ are (infinite) sets of symbols
- $\mathscr{F}$: set of "functional symbols"
  $\mathscr{F} = \bigcup_{n\in\mathbb{N}} \mathscr{F}_n$
  - each $\mathscr{F}_n$ is infinite
  - $\mathscr{F}_n \cap \mathscr{F}_m = \emptyset$ (disjoint)
  - $f \in \mathscr{F}_n$: "f has arity n"
    - if $n = 0$, we don't write $f()$, but f, which is a constant
    - so $\mathscr{C} = \mathscr{F}_0$

Language of formulae

$$\mathscr{L}_F \left\{ \begin{array}{ll} p \in \mathscr{P}, \quad t_1, \ldots, t_m : \text{ terms, then} & \\ \qquad\qquad P(t_1, \ldots, t_m) \text{ is formula ("atom")} & \\ \varphi, \psi \text{ formulae}, \vartheta \in \Theta : & \forall_\vartheta \varphi, \exists_\vartheta \varphi, \\ & \neg\varphi, \varphi \wedge \psi, \varphi \vee \psi, \\ & \varphi \Rightarrow \psi, \varphi \Leftrightarrow \psi \text{ are formulae} \\ \text{(these are all !)} & \end{array} \right.$$

- $\mathscr{P}$ "predicate symbols"
  - $\mathscr{P} = \bigcup_{m\in\mathbb{N}} \mathscr{P}_m$, each $\mathscr{P}_m$ infinite,
  - $\mathscr{P}_n \cap \mathscr{P}_m = \emptyset$ (disjoint)
  - $p \in \mathscr{P}_m$: "p has arity m"
    - if $m = 0$, we don't write $P()$, but P
    - $P \rightsquigarrow$ propositional variable, so propositional logic is a subset of first-order predicate logic

Example

For every two points, there is one and only one line through the two points

$$\forall_x \forall_y P(x) \wedge P(y) \quad \Rightarrow \quad \exists_z ! \, (L(z) \wedge T(z, x, y))$$

$P(x)$: "x is point"
$L(z)$: "z is line"
$T(z, x, y)$: "z passes through x and y"

Schemata for "there exists exactly one":

$$\exists_x ! P(x) \quad : \quad \exists_x P(x) \wedge \left(\forall_y P(y) \Rightarrow \underbrace{(x = y)}_{\text{more precise: } Equal(x,y)} \right)$$

Written without $\exists!$-Usage, and demonstrating terms "scope" and "bound":

$$\forall_x \forall_y \left(P(x) \wedge P(y) \Rightarrow \exists_z \left(\underbrace{(L(z) \wedge T(z,x,y)) \wedge \forall_z \left(\underbrace{(L(z) \wedge T(z,x,y)) \Rightarrow (t = z)}_{\text{scope of the } \forall_z \text{ quantifier}} \right)}_{\substack{\text{scope of the } \exists \text{ quantifier,} \\ \text{variable } z \text{ is bound in this scope}}} \right) \right)$$

In the previous formula all variables are bound by quantifiers.

In the next formula there is an unbound (free) variable:

$$\exists_x \quad \underset{\substack{\uparrow \\ \text{bound}}}{x} \quad \leq \quad \underset{\substack{\uparrow \\ \text{free} \\ \uparrow \\ \text{which } y \text{ ?} \\ \uparrow \\ \text{no meaning is assigned to } y}}{y}$$

One cannot assign a semantics to this formula. Only the closed formulae (i. e. having no free variables) have semantics.

2.2 Semantics

$$f_{\forall_x \exists_y x \leq y} \quad : \quad \underset{\substack{\downarrow \\ \text{interpretations}}}{\mathscr{I}} \quad \rightarrow \{\mathbb{T}, \mathbb{F}\}$$

An interpretation for a formula contains all the elements which are necessary in order to evaluate the truth value of the formula: a domain for the variables, a concrete function for each function symbol (constants will have elements of the domain), and a concrete predicate for each predicate symbol.

$$I \quad : \quad \begin{cases} \text{"domain"} \ldots & D \neq \emptyset \\ \text{constant symbol} \ldots & c_I \in D \\ \text{functional symbol} \ldots \ (\text{arity } n) & f_I : D^n \rightarrow D \\ \text{predicate symbol} \ \ldots \ (\text{arity } m) & p_I : D^m \rightarrow \{\mathbb{T}, \mathbb{F}\} \end{cases}$$

Example:

$$\forall_x \exists_y x \leq y$$

In order to evaluate quantified formulae, since the particular elements of the domain cannot occur in formulae, one uses the notion of *"truth value under the interpretation and a certain assignment to the free variables"*. Not only that the formulae have truth values, but the terms also have values (under the interpretation and a certain assignment).

$$I : \begin{cases} D = \{0,1\} \\ \leq_I : \begin{array}{|c||c|c|}\hline \text{x\y} & 0 & 1 \\ \hline\hline 0 & \mathbb{T} & \mathbb{T} \\ \hline 1 & \mathbb{F} & \mathbb{T} \\ \hline \end{array} \end{cases}$$

$\left\langle \forall_x \exists_y x \leq y \right\rangle_I = \mathbb{T}$ iff foreach $d \in D$:

$\left\langle \exists_y x \leq y \right\rangle^I_{\{x \leftarrow 0\}}$ $\quad$ ~~$\left\langle \exists_y 0 \leq y \right\rangle_I$~~ $\quad$ $\left\langle \exists_y x \leq y \right\rangle^I_{\{x \leftarrow d\}}$

$\langle x \leq y \rangle^I_{\{x \leftarrow 0, y \leftarrow 0\}}$ $\quad$ ~~$\left\langle \exists_y 1 \leq y \right\rangle_I$~~

$$\begin{aligned} &\leq_I \left(\langle x \rangle^I_{\{x \leftarrow 0, y \leftarrow 0\}}, \langle y \rangle^I_{\{x \leftarrow 0, y \leftarrow 0\}} \right) \\ &\leq_I (0,0) = \mathbb{T} \end{aligned}$$

Example: predicate logic formula, interpretation and truth evaluation

$$\forall_x \left(P(x) \Rightarrow Q\left(f(x), a\right) \right)$$

$$\text{for instance } I : \begin{cases} D = \{1,2\} \\ a_I = 1 \quad (\in D) \\ f_I : D \to D \quad \begin{cases} f_I(1) = 1 \\ f_I(2) = 1 \end{cases} \\ P_I : D \to \{\mathbb{T}, \mathbb{F}\} \quad \begin{cases} P_I(1) = \mathbb{T} \\ P_I(2) = \mathbb{F} \end{cases} \\ Q_I : D^2 \to \{\mathbb{T}, \mathbb{F}\} \quad \begin{array}{|c||c|c|}\hline Q_I & 1 & 2 \\ \hline\hline 1 & \mathbb{T} & \mathbb{F} \\ \hline 2 & \mathbb{F} & \mathbb{T} \\ \hline \end{array} \end{cases}$$

$$\begin{aligned} \langle \forall_x \left(P(x) \Rightarrow Q\left(f(x), a\right) \right) \rangle_I &= \mathbb{T} \\ &\text{iff} \\ \langle P(x) \Rightarrow Q\left(f(x), a\right) \rangle^I_{\{x \leftarrow d\}} &= \mathbb{T} \qquad \text{(for each } d \in D\text{)} \end{aligned}$$

So for each element of the domain there is a case:

- Case $d = 1$:

$$\begin{aligned} \langle P(x) \Rightarrow Q\left(f(x), a\right) \rangle^I_{\{x \leftarrow 1\}} &= \mathscr{B}_{\Rightarrow} \left(\langle P(x) \rangle^I_{\{x \leftarrow 1\}}, \langle Q\left(f(x), a\right) \rangle^I_{\{x \leftarrow 1\}} \right) \\ &= \mathscr{B}_{\Rightarrow} \left(P_I \left(\langle x \rangle^I_{\{x \leftarrow 1\}} \right), Q_I \left(\langle f(x) \rangle^I_{\{x \leftarrow 1\}}, \langle a \rangle^I_{\{x \leftarrow 1\}} \right) \right) \\ &= \mathscr{B}_{\Rightarrow} \left(P_I \left(\langle x \rangle^I_{\{x \leftarrow 1\}} \right), Q_I \left(f_I \left(\langle x \rangle^I_{\{x \leftarrow 1\}} \right), \langle a \rangle^I_{\{x \leftarrow 1\}} \right) \right) \\ &= \mathscr{B}_{\Rightarrow} \left(P_I(1), Q_I \left(f_I \left(\langle x \rangle^I_{\{x \leftarrow 1\}} \right), a_I \right) \right) \\ &= \mathscr{B}_{\Rightarrow} \left(\mathbb{T}, Q_I\left(f_I(1), 1\right) \right) = \mathscr{B}_{\Rightarrow} \left(\mathbb{T}, Q_I(1,1) \right) \\ &= \mathscr{B}_{\Rightarrow} (\mathbb{T}, \mathbb{T}) = \mathbb{T} \end{aligned}$$

- Case $d = 2$:

$$\langle P(x) \Rightarrow Q\left(f(x), a\right) \rangle^I_{\{x \leftarrow 2\}} = \ldots = \mathbb{T}$$

2.3 Truth evaluation $\langle\varphi\rangle_I$

2.3.1 Formula

$\mathbb{T}, \mathbb{F}$	
$P(t_1, \ldots, t_n)$	$\langle P(t_1, \ldots, t_n)\rangle^I_\alpha = P_I\left(\langle t_1\rangle^I_\alpha, \ldots, \langle t_n\rangle^I_\alpha\right)$
$\neg\varphi$, $\quad \varphi\Omega\psi$ $\uparrow$ $\in \{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$	$\langle\varphi\Omega\psi\rangle^I_\alpha = \mathscr{B}_\Omega\left(\langle\varphi\rangle^I_\alpha, \langle\psi\rangle^I_\alpha\right)$ $\langle\neg\varphi\rangle^I_\alpha = \mathscr{B}_\neg\left(\langle\varphi\rangle^I_\alpha\right)$
$\forall_\vartheta\varphi$ $\quad (\vartheta:$ variable symbol)	$\langle\forall_\vartheta\varphi\rangle^I_\alpha = \mathbb{T}$ iff (for each $d \in D, \langle\varphi\rangle^I_{\alpha\cup\{\vartheta\leftarrow d\}} = \mathbb{T}$) (of the interpretation I)
$\exists_\vartheta\varphi$ $\quad (\vartheta:$ variable symbol)	$\langle\exists_\vartheta\varphi\rangle^I_\alpha = \mathbb{T}$ iff (for some $d \in D, \langle\varphi\rangle^I_{\alpha\cup\{\vartheta\leftarrow d\}} = \mathbb{T}$) (of the interpretation I)

$\forall_x P(x) \Rightarrow \exists_y Q(x, y)$
$\langle\ldots\rangle^I_{\{\}} \ \ldots\ \langle\ldots\rangle^I_{\{x\leftarrow d\}} \ \ldots\ \langle\ldots\rangle^I_{\{x\leftarrow d, y\leftarrow d\}}$

But what if x is already bound:

$\forall_x P(x) \Rightarrow \exists_x Q(x, x)$
$\langle\ldots\rangle^I_{\{\}} \ \ldots\ \langle\ldots\rangle^I_{\{x\leftarrow d\}} \ \ldots\ \langle\ldots\rangle^I \quad \{x \leftarrow d, x \leftarrow d\}$
$\uparrow$
which x ?

"$\check{\bigcup}$" : modify the assignment !
(replace any other $\{\vartheta \leftarrow \ldots\}$!

All variables must be bound.

All free variables are present in the assignment α

2.3.2 Term

ϑ $\quad (\in$ variable symbol set)	$\langle\vartheta\rangle^I_\alpha = \langle\vartheta\rangle^I_{\{\ldots,\vartheta\leftarrow d,\ldots\}} = d$ (assume that there is such an assignment $\vartheta \leftarrow d$)
c $\quad (\in$ constant symbol set)	$\langle c\rangle^I_\alpha = c_I$
f $\quad (\in$ functional symbol set)	$\langle f(t_1, \ldots, t_n)\rangle^I_\alpha = f_I\left(\langle t_1\rangle^I_\alpha, \ldots, \langle t_n\rangle^I_\alpha\right)$

2.3.3 Equivalence

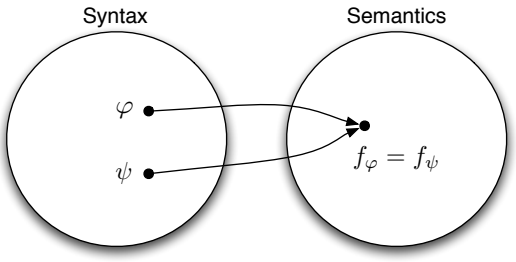


Figure 2.2: Syntax, Semantics and semantical equivalence

Then we can define equivalence of two formulae φ and ψ as that they've got the same semantical function.

$\Phi \models \Psi$ semantical logical consequence

$$\begin{array}{lrcl} \diamond: & \neg(\forall_x \varphi) & \equiv & \exists_x(\neg\varphi) \\ & \neg(\exists_x \varphi) & \equiv & \forall_x(\neg\varphi) \end{array}$$

$$\begin{array}{lrcl} \clubsuit: & \forall_x(\varphi \wedge \psi) & \equiv & (\forall_x\varphi) \wedge (\forall_x\psi) \\ & \exists_x(\varphi \vee \psi) & \equiv & (\exists_x\varphi) \vee (\exists_x\psi) \end{array}$$

$$\left.\begin{array}{l} \forall_x(\varphi \vee \psi) \equiv (\forall_x\varphi) \vee \psi \\ \exists_x(\varphi \wedge \psi) \equiv (\exists_x\varphi) \wedge \psi \end{array}\right\} \quad \begin{array}{c} \text{if } x \mathring{\notin} \psi \\ \text{"does not occur"} \end{array}$$

Substitution:

$$\begin{array}{rcl} \forall_x\varphi & \equiv & \forall_y\varphi_{x\leftarrow y} \\ \multicolumn{3}{c}{\text{if } y \mathring{\notin} \varphi} \\ \exists_x\varphi & \equiv & \exists_y\varphi_{x\leftarrow y} \end{array}$$

We try to prove $\diamond$:

$$\begin{array}{rcll} \text{For arbitary interpretation } I: & & & \\ \langle\neg\forall_x\varphi\rangle^I = \langle\neg\forall_x\varphi\rangle^I_{\{\}} & = & \mathscr{B}_\neg\left(\langle\forall_x\varphi\rangle^I_{\{\}}\right) & \\ & = \mathbb{F} & \text{iff} & \left(\langle\forall_x\varphi\rangle^I_{\{\}} = \mathbb{T}\right) \text{ iff} \\ & & & \text{for each } d \in D: \langle\varphi\rangle^I_{\{x\leftarrow d\}} = \mathbb{T} \\ & = \mathbb{T} & \text{iff} & \ldots \end{array}$$

$$\left(\langle\neg\exists_x\varphi\rangle^I_{\{\}} = \mathbb{F}\right) \quad \text{iff} \quad \text{for some } d \in D: \langle\varphi\rangle^I_{\{x\leftarrow d\}} = \mathbb{T}$$

A formula in "prenex normal form" is of the form:

$$\underbrace{\left(\forall_x \exists_y \exists_z \forall_t \dots\right)}_{\text{all quantifiers}} \quad \underbrace{\varphi}_{\text{quantifier-free formula}}$$

The quantifier-free formula part can be transformed into CNF

$$\left.\begin{array}{r}\text{"prenex formula"}\\ \text{(with the quantifier free formula part in) CNF}\end{array}\right\} \quad \text{"prenex normal form"}$$

So, a formula in *prenex normal form* somehow looks like:

$$\left(\forall_x \exists_y \exists_z \forall_t \dots\right)((\cdots \vee \cdots \vee \dots) \wedge \cdots \wedge (\cdots \vee \cdots \vee \dots))$$

2.4 Skolem transformation

A "skolem transformation" is $\exists_x P(x) \rightsquigarrow P(a)$

if this is satisfiable $\rightarrow$ then $\rightarrow$ this is sat.

this is sat. $\leftarrow$ then $\leftarrow$ if this is sat.

Assume that we have an interpretation I that satisfies $\exists_x P(x)$

$$I: \left\{\begin{array}{l} D \\ P_I : D \to \{\mathbb{T}, \mathbb{F}\} \\ \text{for some } d \in D : P_I(d) = \mathbb{T}\end{array}\right. \quad \mapsto \quad I': \left\{\begin{array}{ll} D & \\ P_{I'} = P_I & \text{so } P_{I'}(a_I) = \mathbb{T} \\ a_{I'} = d & \end{array}\right.$$

$$\begin{array}{rcl}(\exists_x \varphi \text{ satisfiable}) & \text{iff} & (\varphi_{x \leftarrow a} \text{ satisfiable}) \\ \text{and more interesting:} & & \\ (\exists_x \varphi \text{ unsatisfiable}) & \text{iff} & (\varphi_{x \leftarrow a} \text{ unsatisfiable})\end{array}$$

$$\underline{\forall_x \exists_y P(x,y)} \quad \leftarrow\!\shortmid \quad \underline{\forall_x P(x, f(x))}$$

$I' \dots (D, P_I)$
for each $d \in D$
(there is $d' \in D$,
$P_I(d, d') = \mathbb{T}$)
take $d' = f_I(d)$

$I \dots (D, P_I, f_I)$
for each $d \in D$
$P_I(d, f_I(d)) = \mathbb{T}$

Note: Homework 5.5 was to show the other direction $\mapsto$. Hint: Take an interpretation which is sat on the LHS and show that it is sat RHS.

$$\begin{array}{rcl}\left(\forall_x \exists_y \varphi \text{ unsatisfiable}\right) & \text{iff} & \left(\forall_x \varphi_{y \leftarrow f(x)} \text{ unsatisfiable}\right) \\ & & \text{where } f \mathring{\notin} \varphi \\ \text{and more generally:} & & \\ \left(\forall_{x_1} \dots \forall_{x_n} \exists_y \varphi\right) & \rightsquigarrow & \left(\forall_{x_1} \dots \forall_{x_n} \varphi_{x \leftarrow f(x_1, \dots, x_n)}\right)\end{array}$$

So, by *skolem transformation*, one can eliminate all the $\exists$-quantifiers.

$$\begin{array}{c}\left(\forall_x\exists_y\forall_z\forall_t\exists_w\right)\ \underbrace{((\cdots\vee\cdots\vee\ldots)\wedge\cdots\wedge(\cdots\vee\cdots\vee\ldots))}_{CNF}\\ \downarrow\text{ "skolem transform"}\\ (\forall_{x_1}\ldots\forall_{x_n})\ \underbrace{((\cdots\vee\cdots\vee\ldots)\wedge\cdots\wedge(\cdots\vee\cdots\vee\ldots))}_{CNF}\end{array}$$

One does not need to write the quantifiers $(\forall_{x_1}\ldots\forall_{x_n})$ any more ! You simply assume that all are universally quantified.

2.5 Resolution

"Resolution principle": $\left.\begin{array}{c}L\vee C_1\\ \overline{L}\vee C_2\end{array}\right\}\ C_1\vee C_2$

For example:

$$\left.\begin{array}{c}\forall_x P(x)\Rightarrow Q(x)\\ P(a)\end{array}\right\}\ Q(a)\qquad\qquad\left.\begin{array}{c}\overline{P}(x)\vee Q(a)\\ P(a)\end{array}\right\}\ Q(a)$$

If $(L_1\sigma = L_2\sigma)$, $\left.\begin{array}{c}L_1\vee C_1\\ L_2\vee C_2\end{array}\right\}\ C_1\sigma\vee C_2\sigma$

For example:

$$\left.\begin{array}{ccc}\overline{P}(x,a)\vee Q(x) & \overset{\text{instantiate}}{\underset{\{y\leftarrow a\}}{\overset{\{x\leftarrow b\}}{\longrightarrow}}} & \overline{P}(b,a)\vee Q(b)\\ P(b,y)\vee R(y) & & P(b,a)\vee R(a)\end{array}\right\}\ Q(b)\vee R(a)$$

Substitution $\sigma = \{x\leftarrow b, y\leftarrow a\}$

$$\nearrow\quad\text{Correctness: }\begin{array}{c}L_1\vee C_1, \overline{L_2}\vee C_2\models C_1\sigma\vee C_2\sigma\\ (\text{where } L_1\sigma = L_2\sigma)\end{array}\quad\left\{\begin{array}{l}\forall_\vartheta\varphi\models\quad\forall_{\vartheta_1}\ldots\forall_{\vartheta_n}\varphi_{\vartheta\leftarrow t}\\ \qquad\{\vartheta_1,\ldots,\vartheta_n\} = FreeVars(t)\\ \forall_{x_1}\ldots\forall_{x_n}\left((L_1\vee C_1)\wedge\left(\overline{L_2}\vee C_2\right)\right)\Rightarrow(C_1\vee C_2)\\ \updownarrow\\ \quad(\forall_{x_1}\ldots\forall_{x_n}L_1\vee C_1)\\ \quad\left(\forall_{x_1}\ldots\forall_{x_n}\overline{L_2}\vee C_2\right)\end{array}\right.\quad\models\quad(\forall_{x_1}\ldots\forall_{x_n}C_1\vee C_2)$$

$\searrow$ Completeness: If φ unsatisfiable, then $\varphi\vdash_{Res}\square$ (the empty clause)

The *resolution inference rule*:

$\left.\begin{array}{c}L_1\vee C_1\\ L_2\vee C_2\end{array}\right\}\ L_1\sigma = L_2\sigma\vdash C_1\sigma\vee C_2\sigma$

$$\left.\begin{array}{c}\left(\begin{array}{c}\text{formulae which are not ground are}\\ \text{implicitly universally quantified}\end{array}\ \overset{[\forall_x]}{\longrightarrow}\right)\quad P(x,f(a))\vee Q(x)\\ \overline{P}(b,y)\vee R(y)\end{array}\right\}\quad\underbrace{Q(b)\vee R(f(x))}_{\text{"they are ground"}}$$

Substitution $\sigma = \{x\leftarrow b, y\leftarrow f(x)\}$

The *resolution proof method*: "Apply the resolution inference rule until the empty clause is obtained".

- Correct
- Complete

2.5.1 Correctness

$$\frac{L_1\sigma \vee C_1\sigma \quad \overline{L_2}\sigma \vee C_2\sigma}{}$$

Forall $d \in D : \langle P(x)\rangle^I_{\{x\leftarrow d\}} = \mathbb{T}$

$$\begin{array}{ccc} \forall_x P(x) & \models & \forall_y P(f(y)) \\ & x \leftarrow f(y) & \\ P(x) & \vdash & P(f(y)) \end{array}$$

$$\begin{array}{l} \forall_x P(x) \vee Q(x) \\ \forall_x \overline{P}(x) \vee R(x) \end{array} \quad \models \quad \forall_x Q(x) \vee R(x)$$

$$\text{For any } d \in D : \langle Q(x) \vee R(x)\rangle^I_{\{x\leftarrow d\}} = \mathbb{T}$$

$$\begin{array}{c} \mathscr{B}_\vee\left(\left\langle \overline{P}(x)\right\rangle^I_{\{x\leftarrow d\}}, \langle Q(x)\rangle^I_{\{x\leftarrow d\}}\right) \\ = \mathscr{B}_\vee\left(\mathscr{B}_\vee\left(\langle P(x)\rangle^I_{\{x\leftarrow d\}}\right), \langle Q(x)\rangle^I_{\{x\leftarrow d\}}\right) \\ \cdots \end{array}$$

2.5.2 Completeness

For doing this, we will introduce a special domain ("Herbrand universe")

$$\begin{array}{rcl} H & = & \{a, f(x), f(f(x)), \dots\} \text{ (set of ground terms, which is } \infty_{\text{enumerable}}) \\ H_0 & = & \{ \text{ constants } \} \\ H_{k+1} & = & H_k \cup \{ \text{ terms obtained by applying all functional symbols to all elements of } H_k\} \\ H & = & \bigcup_{k\in\mathbb{N}} H_k \text{ (can be constructed in a systematic way)} \\ I : & & \left\{ \begin{array}{l} H \\ f_I : H \to H \qquad f_I(t) = "f(" \smile t \smile ")" \quad \text{(where } \smile \text{ denotes a concatenation of strings)} \\ P_I : H \to \{\mathbb{T}, \mathbb{F}\} \end{array} \right. \\ M & = & \{P(a), P(f(a)), P(f(f(a))), \dots\} \text{ ("atom set": set of ground atoms, which is } \infty_{\text{enumerable}}) \end{array}$$

An interpretation I is a list of ground literals (some are positive, some are not).

One can write all possible interpretations as a tree.

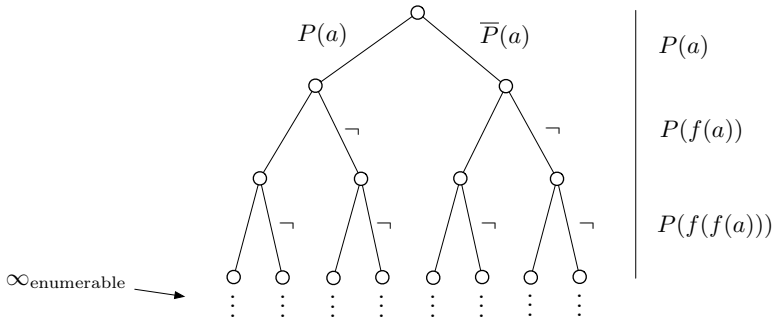


Figure 2.3: "Semantic tree"

$$\begin{array}{rcl} H\text{-satisfiable} & \text{iff} & \text{satisfiable} \\ I_H & \Leftarrow & (\text{over some } I/D) \end{array}$$

Take M and evaluating ground terms $M = \left\{ \begin{array}{ccc} & & \\ \text{"}P(a)\text{"} & , \quad \text{"}P(f(a))\text{"} & , \ldots \\ \uparrow & \uparrow & \\ P_I(a_I) = \mathbb{T} & P_I(f_I(a_I)) = \mathbb{T} & \end{array} \right\}$

So practically it is sufficient so speak about terms "$P(a)$" instead of $P_I(a_I) = \mathbb{T}$

$$\left\{ \begin{array}{ll} P(a) & (1) \\ \overline{P}(x) \vee P(f(x)) & \\ \overline{P}(f(f(a))) & \end{array} \right.$$

So we can close the nodes

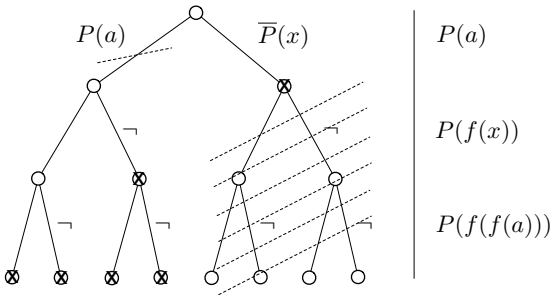
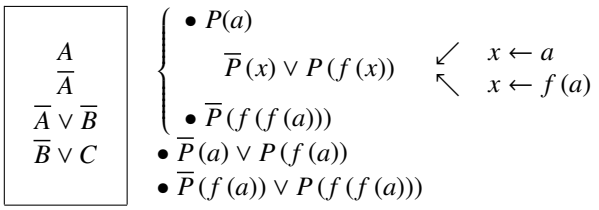


Figure 2.4: Semantic tree with closed nodes

$\Rightarrow$ The semantic tree is closed

Now if it would not be closed, one would have an infinite path somewhere where all clauses evaluate to true.

Conclusion: If φ is unsat, then the semantic tree is closed



Herbrand Theorem: If a formula is unsatisfiable, then there exists an unsatisfiable set of ground instances of the clauses in the formula.

1. This theorem shows the remarkable fact that for proving in first order logic it is enough to reason about ground instances.
2. The theorem also suggests a procedure for proving: we enumerate all possible ground instances of clauses, and for each new instance we test (by propositional logic!) whether the set is already unsatisfiable.
3. Note that this is not a decision procedure: if the original set of clauses in satisfiable, then this process will never terminate (thus, this is a semi-decision procedure).
4. However, this is the best we can hope for first order predicate logic: one can prove that there is no general decision procedure.
5. This proving procedure is not efficient: resolution is more efficient because it finds (by unification) the instances which are more likely to lead to contradiction.

The Herbrand theorem is important because it is a first step towards proving the completeness of the resolution method. Namely, since there exists an unsatisfiable set of ground instances of clauses, it follows (using the construction of the semantic tree), that there exists a deduction by resolution of the empty clause, over the ground instances of clauses.

The next step in proving the incompleteness is to show that there is a deduction by resolution of the empty clause over the original set of clauses. This is done by "lifting" the ground deduction to a non-ground deduction, using the *Lifting Lemma*. The Lifting Lemma states that every resolvent of two ground instances of clause is itself an instance of a resolvent of two original instances. Thus, we can replace each grounde resolution step by a resolution step over the non-ground clauses, and this is the "lifting" of the deduction.

The completeness of resolution is in fact equivalent to *Gödel's Completeness Theorem*, which is of great importance for the philosophy of logic and also of computer science.

Since resolution calculus is essentially equivalent to natural deduction calculus (used in Gödel's theorem), one has the following:

If a formula is unsatisfiable, then one can obtain a contradiction by natural deduction, thus False *is a logical consequence of it.*

(A formula having the latter property is called *inconsistent*, and a formulae from which False does not follow are called *consistent.*)

By contraposition (reversed implication between the negations) one obtains:

If a formula is consistent, than it has a model (that is: an interpretation for which the formula evaluates to True).

This is the Gödel's completeness theorem, which states the remarkable fact that for every noncontradictory theory in first order predicate logic there exists a domain on which this theory holds.

Moreover, the proof is constructive and (similarly to the proof of the Herbrand theorem) it actually exhibits a concrete domain on which the theory holds: this is the Herbrand universe (that is set of ground terms),

together with the functions and the predicates which occur in the theory, thus its construction only needs the "syntactic material" which is already present in the respective theory.

When we want to solve a problem with the help of the computer, we start from an abstract model of the problem (that is a logical theory) and we need to create a concrete domain on which the functions and the predicates of the theory can be implemented. The completeness theorem shows that this is allways possible, and also gives us a general method to realize the implementation.