Lecture Notes

Logic in Computer Science

taught in Summer term 2017

by

Sven Kosub

May 18, 2017

Version v1.2

Contents

Prologue

A very short (and incomplete) history of mathematics-based logic in a perspective of computer science:

$\sim$ 1700	Gottfried Wilhelm Leibniz	*characteristica universalis*; binary numbers
1847	George Boole	boolean algebra and logic; *The Laws of Thought*
1874	Georg Cantor	first formal statements about sets (*Inbegriffe*)
1879	Gottlob Frege	*Begriffschrift*; first use of quantifiers
1895	Georg Cantor	naïve set theory
1902	Bertrand Russell	Russell's paradox
1903	Bertrand Russell, Alfred North Whitehead	*Principia mathematica*
1920	David Hilbert	Hilbert's program (e.g., solvability of diophantic equations, axiomatizability of mathematics)
1928	Kurt Gödel	completeness of first-order logic
1931	Kurt Gödel	incompleteness of arithmetics
1936	Alonzo Church Alan Turing Emil Post	formalization of the notion of an algorithm

Around that time, as computer science began to exist as a scientific discipline, mathematical logic became an integral part of computer science at all levels and differentiated into its full entirety. The following list contains exemplary computer science applications of logics beyond rigorous mathematical proofs:

- logic programming (e.g., PROLOG)
- knowledge representation
- rule-based machine learning
- automated theorem proving
- model checking
- circuits
- ...

Today, formal and mathematical logic is mainly used in a computer scientist's everyday life for specifying computer systems etc., i.e., for the definability of certain objects in appropriate logical systems. In the following, we want to exemplify this usage.

Assume we want to describe words $w = w_1 \dots w_n$ over the alphabet $\Sigma = \{a, b\}$ by letters at certain positions. That is, we use two predicates P_a, P_b so that $P_a(i), P_b(i)$ are true if and only if $w_i = a, w_i = b$. If position i does not exist in a particular word then both $P_a(i)$

and $P_b(i)$ are false. For instance, the expression $P_a(1) \wedge P_b(2)$ is true for each word having letter a at its first and letter b at its second position. That is, the expression defines the regular language $ab\{a,b\}^*$.

The other way round, we might ask which formula defines a certain language or a certain pattern we want to have in all words w of that language, e.g.:

1. "letter a occurs," i.e., $w \in \{a,b\}^*a\{a,b\}^*$: $(\exists x)[P_a(x)]$
2. "last letter is b," i.e., $w \in \{a,b\}^*b$: $(\exists x)(\forall y)[x \geq y \wedge P_b(x)]$

   Note that the expression $P_b(n)$ is not correct as it depends on n which is part of the "input." We have to find a formula forcing that a certain position is the last position—here, the maximum. Defining $(x = \max) =_{\text{def}} (\forall y)[x \geq y]$, we can rewrite the expression equivalently as:

   $$(\exists x)[(x = \max) \wedge P_b(x)]$$

   Obviously, $(x = \min) =_{\text{def}} (\forall y)[x \leq y]$.
3. "a is immediately followed by b," i.e., $w \in b^*(abb^*)^*$: $(\forall x)[P_a(x) \rightarrow P_b(x+1)]$

   But, what if the increment $x + 1$ (as a function) is not allowed in our logic. Then, we have to define what is the next position given a position x. Together with the formula above, this can be clearly expressed by the following formula:

   $$(\forall x)[P_a(x) \rightarrow (\exists y)[P_b(y) \wedge x < y \wedge \neg(\exists z)[x < z \wedge z < y]]]$$

   So, defining $(y = x + 1) =_{\text{def}} x < y \wedge (\forall z)[x \geq z \vee z \geq y]$, we can rewrite the expression equivalently as:

   $$(\forall x)[P_a(x) \rightarrow (\exists y)[(y = x + 1) \wedge P_b(y)]]$$

It is clear by these examples that it is important which predicates or functions are allowed to express statements on words formally. We want to discuss two more, more complex examples.

How to describe the regular language $ab(ab)^*$? This can be done by the following formula:

$$(\exists x)[(x = \min) \wedge P_a(x)] \ \wedge \ (\forall x)[P_a(x) \leftrightarrow (\exists y)[(y = x + 1) \wedge P_b(y)]]$$

Suppose the formula is true for a (finite) word w. The first subformula forces w to start with an a. The second subformula is true if at a certain position there is letter a then at the next position there is letter b. For instance, ab satisfies the formula (note that $P_a(2)$ is false and there exists no y such that $y = x + 1$). So do all words $ab(ab)^*$. In contrast, aba does not satisfy the formula (note that $P_a(3)$ is true but there is no next position).

How to describe the regular language $aa(aa)^*$? Though seemingly simpler than the language above, the situation is in fact, more complicated as we have to check for words of even length explicitly. We use the idea to check if it is possible to partition the set of

positions into two disjoint sets of the same size. For two finite sets to have the same size we check if there is a bijective mapping between the sets. The following formula realizes this idea:

$$
\begin{aligned}
(\exists A)(\exists B) \quad [\quad & (\forall x)[x \in A \leftrightarrow x \notin B] \\
& \wedge\ (\exists x)[(x = \min) \wedge x \in A] \\
& \wedge\ (\exists x)[(x = \max) \wedge x \in B] \\
& \wedge\ (\forall x)(\forall y)[x \in A \wedge (y = x + 1) \rightarrow y \in B] \\
& \wedge\ (\forall x)(\forall y)[x \in B \wedge (y = x + 1) \rightarrow y \in A] \quad]
\end{aligned}
$$

It is important to note that there is a fundamental difference between both formulas. In the first formula, quantifiers only range over positions. We will say that the formula is first order (FO). In the second formula, there are existential quantifiers $(\exists A)$ and $(\exists B)$ ranging over sets of positions. We will say that the formula is monadic second order (MSO). So, $ab(ab)^*$ is FO-definable and $aa(aa)^*$ is MSO-definable. Later, in the chapter on finite model theory, we will learn that it is not possible to define $aa(aa)^*$ by an FO formula.

Propositional logic 1

In a sense, propositional logic (PL) is the coarsest logic: PL is domain-independent. Statements are only distinguished with respect to their truth values, e.g., there is no difference between the sentences "$2 + 3 = 5$" and "Konstanz is situated on Lake Constance" as both statements are true. So, statements are represented by propositional variables X which can take on exactly one of the values 0 (or "true") or 1 (or "false").

The use of propositional variables eliminates non-sense sentences, e.g., "2 and 3 form an acute angle," and paradoxical sentences, e.g., "This sentence is false" from logic. In both cases, it is not possible to assign exactly one truth value. Note that there are scenarios and domains where a formal treatment of paradoxical situations is necessary. In these cases, specific logical approaches exist, e.g., paraconsistent logic or variants of many-valued logic. However, the soundness of these logical systems is based on PL.

1.1 Syntax and semantics of propositional logic

We use the following notations and symbols for defining propositional formulas:

- $\tau = \{X_0, X_1, X_2, \ldots\}$ is a countable set of propositional variables
- $0, 1$ are constant symbols
- $\neg, \wedge, \vee, \rightarrow$ are (logical) connectives
- $(,)$ are parenthesis symbols

Definition 1.1 *The set* PL *of* propositional formulas *is inductively defined as follows:*

1. base case:

$\tau \subseteq \mathsf{PL}$	*(i.e., variables are formulas)*
$0, 1 \in \mathsf{PL}$	*(i.e., constants are formulas)*

2. inductive step: *if* $\varphi, \psi \in \mathsf{PL}$ *then*

$\neg\varphi \in \mathsf{PL}$	*(*negation*; stands for: "* not φ*")*
$(\varphi \wedge \psi) \in \mathsf{PL}$	*(*conjunction*; stands for: "* φ and ψ*")*
$(\varphi \vee \psi) \in \mathsf{PL}$	*(*disjunction*; stands for: "* φ or ψ*")*
$(\varphi \rightarrow \psi) \in \mathsf{PL}$	*(*implication*; stands for: "* if φ then ψ*")*

The formulas defined in the base case are called atomic.

Further connectives are common (though not considered in the forthcoming):

$$(\varphi \leftrightarrow \psi) \;=_{\text{def}}\; ((\varphi \to \psi) \wedge (\psi \to \varphi)) \qquad \text{(equivalence)}$$
$$(\varphi \oplus \psi) \;=_{\text{def}}\; \neg(\varphi \leftrightarrow \psi) \qquad \text{(exclusive disjunction)}$$
$$(\varphi \mid \psi) \;=_{\text{def}}\; \neg(\varphi \wedge \psi) \qquad \text{(Sheffer stroke)}$$

Notice that $\mathsf{PL} \subseteq (\tau \cup \{0, 1, \neg, \wedge, \vee, \to, (,)\})^*$ and, when using an encoding of X_i by $X\underbrace{||\ldots|}_{i \text{ times}}$, even $\mathsf{PL} \subseteq \{X, 0, 1, \neg, \wedge, \vee, \to, (,)\}^*$. So, propositional formulas can be represented as words over a finite alphabet, e.g., when formulas are inputs to some algorithm.

We mention some further remarks:

1. The equality symbol $=$ stands for equality of symbol sequences.
2. As in the case of arithmetical formulas, parentheses are often omitted according to the following binding rules:
   - $\neg$ binds stronger than $\wedge, \vee, \to$
   - $\wedge, \vee$ bind stronger than $\to$
   - implicit left parentheses in iterated conjunctions and disjunctions

   For instance, $\varphi \wedge \neg\psi \to \eta = ((\varphi \wedge \neg\psi) \to \eta)$ and $\varphi \wedge \psi \wedge \eta = ((\varphi \wedge \psi) \wedge \eta)$.
3. We also use the following abbreviations for iterated conjunctions and disjunctions:

$$\bigwedge_{i=1}^{n} \varphi_i \;=_{\text{def}}\; \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \cdots \wedge \varphi_n$$
$$\bigvee_{i=1}^{n} \varphi_i \;=_{\text{def}}\; \varphi_1 \vee \varphi_2 \vee \varphi_3 \vee \cdots \vee \varphi_n$$

Definition 1.2 *An* assignment *is a mapping $I : \tau \to \{0, 1\}$. Given an assignment I, the* interpretation $[\![\varphi]\!]^I \in \{0, 1\}$ *of φ is inductively defined as follows:*

1. base case:

$$[\![X]\!]^I \;=_{\text{def}}\; I(X) \quad \textit{for all } X \in \tau$$
$$[\![0]\!]^I \;=_{\text{def}}\; 0, \quad [\![1]\!]^I \;=_{\text{def}}\; 1$$

2. inductive step:

$$[\![\neg\varphi]\!]^I \;=_{\text{def}}\; 1 - [\![\varphi]\!]^I$$
$$[\![\varphi \wedge \psi]\!]^I \;=_{\text{def}}\; \min\{[\![\varphi]\!]^I, [\![\psi]\!]^I\}$$
$$[\![\varphi \vee \psi]\!]^I \;=_{\text{def}}\; \max\{[\![\varphi]\!]^I, [\![\psi]\!]^I\}$$
$$[\![\varphi \to \psi]\!]^I \;=_{\text{def}}\; [\![\neg\varphi \vee \psi]\!]^I$$

Let $\tau(\varphi)$ denote the set of variables occurring in $\varphi \in \mathsf{PL}$.

Lemma 1.3 *Let $\varphi \in \mathsf{PL}$ be a propositional formula and let I, I' be assignments. If $I(X) = I'(X)$ for all $X \in \tau(\varphi)$ then $[\![\varphi]\!]^I = [\![\varphi]\!]^{I'}$.*

It follows from this coincidence lemma that only assignments to variables in $\tau(\varphi)$ are needed to be considered when determinig the interpretation of a formula.

As long as possible, we meet the following conventions:

- small greek letters $\varphi, \psi, \eta, \ldots$ stand for formulas,
- capital greek letters $\Phi, \Psi, \Gamma, \ldots$ stand for sets of formulas,
- $\varphi(X_1, \ldots, X_n)$ indicates that $\tau(\varphi) \subseteq \{X_1, \ldots, X_n\}$, and
- for $I(X_1) = w_1, \ldots, I(X_n) = w_n$, we use $[\![\varphi(w_1, \ldots, w_n)]\!]$, $[\![\varphi(w)]\!]$ to denote $[\![\varphi]\!]^I$.

Examples:

- Consider the following propositional formula

$$\varphi =_{\text{def}} ((\neg(X_1 \to (X_2 \vee \neg X_3)) \to (X_2 \to (\neg X_2 \wedge X_3))).$$

We decompose φ to the following subformulas:

$$\begin{aligned}
\varphi_1 &=_{\text{def}} X_2 \vee \neg X_3 \\
\varphi_2 &=_{\text{def}} \neg(X_1 \to \varphi_1) \\
\varphi_3 &=_{\text{def}} \neg X_2 \wedge X_3 \\
\varphi_4 &=_{\text{def}} X_2 \to \varphi_3
\end{aligned}$$

Thus, $\varphi = \varphi_2 \to \varphi_4$. We obtain the following truth table (consisting of interpretations of subformulas):

X_1	X_2	X_3	φ_1	φ_2	φ_3	φ_4	φ
0	0	0	1	0	0	1	1
0	0	1	0	0	1	1	1
0	1	0	1	0	0	0	1
0	1	1	1	0	0	0	1
1	0	0	1	0	0	1	1
1	0	1	0	1	1	1	1
1	1	0	1	0	0	0	1
1	1	1	1	0	0	0	1

So, φ is true for all assignments.

- Consider the following propositional formula (for the colloquial "and/or"):

$$\varphi =_{\text{def}} ((X_1 \wedge X_2) \vee (X_1 \vee X_2)).$$

Again, we decompose φ to subformulas:

$$\varphi_1 =_{\text{def}} X_1 \wedge X_2, \qquad \varphi_2 =_{\text{def}} X_1 \vee X_2$$

Thus, $\varphi = \varphi_1 \vee \varphi_2$. We obtain the following truth table:

X_1	X_2	φ_1	φ_2	φ
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	1

So, φ and φ_2 have the same interpretations (i.e., we can omit φ_1).

Definition 1.4 *Let $\varphi, \psi \in \mathsf{PL}$ be propositional formulas.*

1. *A formula φ is said to be a* tautology *if and only if $[\![\varphi]\!]^I = 1$ for all assignments I.*
2. *A formula φ is said to be* satisfiable *if and only if $[\![\varphi]\!]^I = 1$ for some assignment I.*
3. *The formulas φ and ψ are said to be* (logically, semantically) equivalent*, in symbols, $\varphi \equiv \psi$, if and only if $[\![\varphi]\!]^I = [\![\psi]\!]^I$ for all assignments I.*

Example: In the examples above, the first formula is a tautology. The second formula is not a tautology, but satisfiable; it holds that $\varphi \equiv X_1 \vee X_2$.

Note that $\equiv$ is an equivalence relation on PL (i.e., $\equiv$ is reflexive, transitiv, symmetric).

Proposition 1.5 *Let $\varphi, \psi \in \mathsf{PL}$ be propositional formulas.*

1. *φ is a tautology $\iff$ $\neg\varphi$ is not satisfiable*
2. *φ is satisfiable $\iff$ $\neg\varphi$ is not a tautology*
3. *φ, ψ are tautologies $\implies$ $\varphi \equiv \psi$*
4. *$\varphi \equiv \psi \iff (\varphi \leftrightarrow \psi)$ is a tautology*

Proof: Exercise. ∎

A list of important equivalences:

$\varphi \equiv (\varphi \wedge \varphi)$ $\varphi \equiv (\varphi \vee \varphi)$	idempotent laws
$(\varphi \wedge \psi) \equiv (\psi \wedge \varphi)$ $(\varphi \vee \psi) \equiv (\psi \vee \varphi)$	commutative laws
$((\varphi \wedge \psi) \wedge \eta) \equiv (\varphi \wedge (\psi \wedge \eta))$ $((\varphi \vee \psi) \vee \eta) \equiv (\varphi \vee (\psi \vee \eta))$	associative laws
$((\varphi \wedge \psi) \vee \varphi) \equiv \varphi$ $((\varphi \vee \psi) \wedge \varphi) \equiv \varphi$	absorption laws
$((\varphi \wedge \psi) \vee \eta) \equiv ((\varphi \vee \eta) \wedge (\psi \vee \eta))$ $((\varphi \vee \psi) \wedge \eta) \equiv ((\varphi \wedge \eta) \vee (\psi \wedge \eta))$	distributive laws
$\neg(\varphi \wedge \psi) \equiv (\neg\varphi \vee \neg\psi)$ $\neg(\varphi \vee \psi) \equiv (\neg\varphi \wedge \psi)$	De Morgan's laws
$\neg\neg\varphi \equiv \varphi$	double negation law
$(\varphi \wedge 0) \equiv 0$ $(\varphi \vee 1) \equiv 1$	domination laws
$(\varphi \wedge 1) \equiv \varphi$ $(\varphi \vee 0) \equiv \varphi$	identity laws
$(\varphi \wedge \neg\varphi) \equiv 0$ $(\varphi \vee \neg\varphi) \equiv 1$	*tertium non datur*
$(\varphi \to \psi) \equiv (\neg\psi \to \neg\varphi)$	contraposition

Example: We want to show that the first formula in the examples above is a tautology without using truth tables. Using the list above and the obvious equivalence $(\varphi \to \psi) \equiv (\neg\varphi \vee \psi)$, we obtain the following:

$$\begin{aligned}
\neg(X_1 \to (X_2 \vee \neg X_3)) \to (X_2 \to (\neg(X_2 \wedge X_3)) & \\
&\equiv \neg(\neg(\neg X_1 \vee (X_2 \vee \neg X_3))) \vee (\neg X_2 \vee (\neg X_2 \wedge X_3)) \\
&\equiv (\neg X_1 \vee (X_2 \vee \neg X_3)) \vee \neg X_2 \\
&\equiv (\neg X_1 \vee \neg X_3) \vee (X_2 \vee \neg X_2) \\
&\equiv (\neg X_1 \vee \neg X_3) \vee 1 \\
&\equiv 1
\end{aligned}$$

1.2 Boolean functions and normal forms

Any formula $\varphi(X_1, \ldots, X_n) \in \mathsf{PL}$ is associated with a boolean function $f : \{0,1\}^n \to \{0,1\}$ via its interpretations. In this section, we want to study propositional formulas in terms of boolean functions. The set of all boolean functions is denoted by BF:

$$\mathsf{BF} =_{\text{def}} \{\ f \mid f : \{0,1\}^n \to \{0,1\}, n \in \mathbb{N}_+\ \}$$

There are 2^{2^n} boolean functions of arity n. The 4 unary boolean functions f^1_j are (in lexicographical order):

x_1	f^1_0	f^1_1	f^1_2	f^1_3
0	0	0	1	1
1	0	1	0	1
	C^1_0	id	non	C^1_1

Here we use the following function names: C^1_0, C^1_1 are unary *constant* 0, 1, id is the *identity*, and non is the *negation* (also denoted by NOT).

The 16 binary boolean functions f^2_j are (in lexicographical order):

x_1	x_2	f^2_0	f^2_1	f^2_2	f^2_3	f^2_4	f^2_5	f^2_6	f^2_7	f^2_8	f^2_9	f^2_{10}	f^2_{11}	f^2_{12}	f^2_{13}	f^2_{14}	f^2_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
		C^2_0	et		id^2_1		id^2_2	aut	vel	nd	eq	non^2_2		non^2_1	seq	sh	C^2_1

The names commonly used in logic for the most relevant functions are: $\mathrm{C}^2_0, \mathrm{C}^2_1$ are the binary *constants* 0, 1, $\mathrm{id}^2_0, \mathrm{id}^2_1$ are *identities* in the first and second argument (i.e., projections), $\mathrm{non}^2_0, \mathrm{non}^2_1$ are *negations* in the first and second argument, et is the *conjunction* (also denoted by AND), vel is the *disjunction* (also denoted by OR), seq is the *implication*, eq is the *equivalence* (also denoted by XNOR), aut is the *exclusive disjunction* (also denoted by XOR), nd is *Nicod's function* (also denoted by NOR), and sh is *Sheffer's function* (also denoted by NAND).

In an obvious way, we can define a homomorphism $(\mathsf{PL}; \neg, \wedge, \vee, \to) \to (\mathsf{BF}; \mathrm{non}, \mathrm{et}, \mathrm{vel}, \mathrm{seq})$. To see this, let $\varphi = \varphi(X_1, \ldots, X_n) \in \mathsf{PL}$ and define a function f_φ depending on φ as

$$f_\varphi(w_1, \ldots, w_n) =_{\text{def}} [\![\varphi(w_1, \ldots, w_n)]\!].$$

Then, $f_\varphi([\![X_1]\!]^I, \ldots, [\![X_n]\!]^I) = [\![\varphi(X_1, \ldots, X_n)]\!]^I$ for all assignments I. Thus, the mapping $\varphi \mapsto f_\varphi$ is homomorphism. Intuitively, this means that we can replace logic by algebra (and computation).

Which boolean functions can be defined by propositional formulas?

We answer to this question by considering disjunctive normalforms (DNF) and conjunctive normal forms (CNF). First, we define for a propositional variable X:

$$X^0 =_{\text{def}} \neg X, \qquad X^1 =_{\text{def}} X$$

Propositional formulas X and $\neg X$ are called *literals*; X is a positive literal, $\neg X$ is a negative literal. A propositional formula $\varphi \in \mathsf{PL}$ is in *disjunctive normal form* or, in short, is a DNF, if and only if it has the form

$$\bigvee_{i=1}^{n} \bigwedge_{j=1}^{n_i} L_{ij},$$

where L_{ij} is a literal. A propositional formula $\varphi \in \mathsf{PL}$ is in *conjunctive normal form* or, in short, is a CNF, if and only if it has the form

$$\bigwedge_{i=1}^{n} \bigvee_{j=1}^{n_i} L_{ij},$$

where L_{ij} is a literal.

Proposition 1.6 *For each assignment I, the following statements hold:*

1. $[\![X^w]\!]^I = 1 \iff [\![X]\!]^I = w$
2. $[\![\bigwedge_{i=1}^n \varphi_i]\!]^I = 1 \iff [\![\varphi_i]\!]^I = 1$ *for all* $i \in \{1, \ldots, n\}$
3. $[\![\bigvee_{i=1}^n \varphi_i]\!]^I = 1 \iff [\![\varphi_i]\!]^I = 1$ *for some* $i \in \{1, \ldots, n\}$

Theorem 1.7 *For each satisfiable formula $\varphi = \varphi(X_1, \ldots, X_n) \in \mathsf{PL}$,*

$$\varphi \equiv \bigvee_{f_\varphi(w_1,\ldots,w_n)=1} \bigwedge_{i=1}^{n} X_i^{w_i}.$$

The constructed formula on the right-hand side is called *canonical DNF*. Notice that the canonical DNF for a satisfiable formula φ needs not be the shortest DNF equivalent to φ. In case that φ is not satisfiable, $X \wedge \neg X$ is an equivalent DNF.

Proof: Let $I : \tau \to \{0, 1\}$ be an assignment. We conclude:

$$\begin{aligned}
[\![\varphi]\!]^I = 1 &\iff f_\varphi([\![X_1]\!]^I, \ldots, [\![X_n]\!]^I) = 1 \\
&\iff \text{there exists } (w_1, \ldots, w_n) \in \{0,1\}^n \text{ such that } f_\varphi(w_1, \ldots, w_n) = 1 \\
&\qquad\qquad \text{and } [\![X_i]\!]^I = w_i \text{ for all } i \in \{1, \ldots, n\} \\
&\iff \text{there exists } (w_1, \ldots, w_n) \in \{0,1\}^n \text{ such that } f_\varphi(w_1, \ldots, w_n) = 1
\end{aligned}$$

$$\begin{aligned} & \text{and } [\![X_i^{w_i}]\!]^I = 1 \text{ for all } i \in \{1,\ldots,n\} \\ \iff & \text{there exists } (w_1,\ldots,w_n) \in \{0,1\}^n \text{ such that } f_\varphi(w_1,\ldots,w_n) = 1 \\ & \text{and } [\![\textstyle\bigwedge_{i=1}^n X_i^{w_i}]\!]^I = 1 \\ \iff & [\![\textstyle\bigvee_{f_\varphi(w_1,\ldots,w_n)=1} \bigwedge_{i=1}^n X_i^{w_i}]\!]^I = 1 \end{aligned}$$

This proves the theorem. ■

By duality, we have a similar theorem for canonical CNF's.

Theorem 1.8 *For each non-tautology $\varphi = \varphi(X_1,\ldots,X_n) \in \mathsf{PL}$,*

$$\varphi \equiv \bigwedge_{f_\varphi(w_1,\ldots,w_n)=0} \bigvee_{i=1}^n X_i^{1-w_i}.$$

Proof: Swap $\wedge$ with $\vee$ and 0 with 1 in the proof of Theorem 1.7. ■

1.3 Models and proofs in propositional logic

Definition 1.9 *Let $\Phi \subseteq \mathsf{PL}$ be a set of propositional formulas.*

1. *An assignment $I : \tau \to \{0,1\}$ is said to be a* model *of Φ, in symbols, $[\![\Phi]\!]^I = 1$, if and only if $[\![\varphi]\!]^I = 1$ for all $\varphi \in \Phi$.*
2. *Φ is said to* entail *a propositional formula $\varphi \in \mathsf{PL}$, in symbols, $\Phi \models \varphi$, if and only if $[\![\Phi]\!]^I \leq [\![\varphi]\!]^I$ for all assignments I.*
3. *$\Phi^{\models} =_{\text{def}} \{\ \varphi \in \mathsf{PL} \mid \Phi \models \varphi\ \}$.*

Note that to check if $\Phi \models \varphi$ we have to verify that $[\![\varphi]\!]^I = 1$ whenever $[\![\Phi]\!]^I = 1$. In other words, $\Phi \models \varphi$ means that each model of Φ is also a model of φ.

Example:

- $\{\psi, \varphi\} \models \psi \wedge \varphi$.
- $\{\psi, \psi \to \varphi\} \models \varphi$.
- If $\Phi \cup \{\psi\} \models \varphi$ and $\Phi \cup \{\neg\psi\} \models \varphi$ then $\Phi \models \varphi$.

Proposition 1.10 $\models$ *is a closure operator on* $\mathcal{P}(\mathsf{PL})$*, i.e.,* $\models$ *satisfies the following conditions for all set* $\Phi, \Psi \subseteq \mathsf{PL}$*:*

1. $\Phi \subseteq \Phi^{\models}$ *(*$\models$ *is extensive)*
2. *If* $\Phi \subseteq \Psi$ *then* $\Phi^{\models} \subseteq \Psi^{\models}$ *(*$\models$ *is increasing)*
3. $\left(\Phi^{\models}\right)^{\models} = \Phi^{\models}$ *(*$\models$ *is idempotent)*

Proof: We prove all properties of a closure operator individually:

1. We have to show: $\varphi \in \Phi \Longrightarrow \Phi \models \varphi$. This is easily seen as if $[\![\Phi]\!]^I = 1$ then $[\![\varphi]\!]^I = 1$ since $\varphi \in \Phi$. Hence, $\Phi \models \varphi$.

2. We have to show: $\Phi \models \varphi \Longrightarrow \Psi \models \varphi$. This is easily seen as if $[\![\Psi]\!]^I = 1$ then $[\![\Phi]\!]^I = 1$, since $\Phi \subseteq \Psi$, and thus, $[\![\varphi]\!]^I = 1$, since $\Phi \models \varphi$. Hence, $\Psi \models \varphi$.

3. it is enough to show: $\Phi^{\models} \models \varphi \Longrightarrow \Phi \models \varphi$. This is easily seen as if $[\![\Phi]\!]^I = 1$ then $[\![\psi]\!]^I = 1$ for all $\psi \in \Phi^{\models}$ (by definition), i.e., $[\![\Phi^{\models}]\!]^I = 1$, and thus, $[\![\varphi]\!]^I = 1$, since $\Phi^{\models} \models \varphi$. Hence, $\Phi \models \varphi$.

This proves the proposition. ∎

Proposition 1.11 *Let* $\varphi, \psi\varphi_1, \ldots, \varphi_n \in \mathsf{PL}$ *be propositional formulas.*

1. φ *is a tautology* $\iff \emptyset \models \varphi$
2. $\emptyset \models (\varphi \to \psi) \iff \{\varphi\} \models \psi$
3. $\varphi \equiv \psi \iff \{\varphi\} \models \psi$ *and* $\{\psi\} \models \varphi$
4. $\varphi \equiv \psi \iff \Phi \models \varphi$ *if and only if* $\Phi \models \psi$ *for all sets* $\Phi \subseteq \mathsf{PL}$
5. $\{\varphi_1, \ldots, \varphi_n\}^{\models} = \{\bigwedge_{i=1}^{n} \varphi_i\}^{\models}$

Proof: Exercise. ∎

Definition 1.12 *Let $\Phi \subseteq \mathsf{PL}$ be a set of propositional formulas. The set $\Phi^\vdash$ (i.e., the set of formulas derived from Φ) is inductively defined as follows:*

1. Base case: $\Phi \subseteq \Phi^\vdash$.
2. Inductive step: *Let $\varphi, \psi \in \mathsf{PL}$ be propositional formulas.*

$$\varphi \in (\Phi \cup \{\psi\})^\vdash,\ \varphi \in (\Phi \cup \{\neg\psi\})^\vdash \implies \varphi \in \Phi^\vdash \quad \textit{(case distinction; (CD))}$$
$$\psi, \neg\psi \in (\Phi \cup \{\neg\varphi\})^\vdash \implies \varphi \in \Phi^\vdash \quad \textit{(indirect proof; (IP))}$$
$$\neg\psi, (\psi \vee \varphi) \in \Phi^\vdash \implies \varphi \in \Phi^\vdash \quad \textit{(modus ponens; (MP))}$$
$$\varphi \in \Phi^\vdash \implies (\varphi \vee \psi), (\psi \vee \varphi) \in \Phi^\vdash \quad \textit{(disjunction introduction; (DI))}$$
$$\varphi, \psi \in \Phi^\vdash \implies (\varphi \wedge \psi) \in \Phi^\vdash \quad \textit{(conjunction introduction; (CI))}$$
$$(\varphi \wedge \psi) \in \Phi^\vdash \implies \varphi, \psi \in \Phi^\vdash \quad \textit{(conjunction elimination; (CE))}$$
$$(\neg\varphi \vee \psi) \in \Phi^\vdash \implies (\varphi \to \psi) \in \Phi^\vdash \quad \textit{(implication; (IMP))}$$

Finally, $\Phi \vdash \varphi$ denotes $\varphi \in \Phi^\vdash$.

Examples:

- $\emptyset \vdash (\varphi \vee \neg\varphi)$:

  By base case, $\{\varphi\} \vdash \varphi$. Thus, $\emptyset \cup \{\varphi\} = \{\varphi\} \vdash (\varphi \vee \neg\varphi)$ by disjunction introduction (DI). Again, by base case, $\{\neg\varphi\} \vdash \neg\varphi$. Hence, $\emptyset \cup \{\neg\varphi\} \vdash (\varphi \vee \neg\varphi)$ by disjunction introduction (DI). So, $\emptyset \vdash (\varphi \vee \neg\varphi)$ by case distinction (CD).

- $\{\varphi, \neg\varphi\}^\vdash = \mathsf{PL}$:

  Let $\psi \in \mathsf{PL}$ be an arbitrary formula. By base case, $\neg\varphi, \varphi \in \{\varphi, \neg\varphi, \neg\psi\}^\vdash$. Hence, $\psi \in \{\varphi, \neg\varphi\}^\vdash$ by indirect proof (IP).

- $\{\varphi\} \vdash \neg\neg\varphi$:

  By base case, $\{\varphi, \neg\neg\varphi\} \vdash \neg\neg\varphi$, i.e., $\neg\neg\varphi \in \{\varphi, \neg\neg\varphi\}^\vdash$. Furthermore, $\neg\neg\varphi \in \{\varphi, \neg\varphi\}^\vdash = \mathsf{PL}$. Hence, $\neg\neg\varphi \in \{\varphi\}^\vdash$ by case distinction (CD).

Proposition 1.13 *$^\vdash$ is a closure operator on $\mathcal{P}(\mathsf{PL})$, i.e., $^\vdash$ is extensive, increasing, and idempotent.*

The proofs of the following theorems are postponed to Ch. **??** where we prove the same results in the more general setting of first-order logic. The theorems mentioned here appear as special cases.

Theorem 1.14 (Correctness theorem) *$\Phi^\vdash \subseteq \Phi^\models$ for all sets $\Phi \subseteq \mathsf{PL}$.*

In other words, Theorem 1.14 states that everything that can be proven is also true.

Theorem 1.15 (Completeness theorem) $\Phi^{\models} \subseteq \Phi^{\vdash}$ *for all sets* $\Phi \subseteq \mathsf{PL}$.

In other words, Theorem 1.15 states that everything that is true can also be proven. We should note that this theorem cannot be proven anymore if one of the rules in Definition 1.12 is omitted.

Both together Theorem 1.14 and 1.15 show that the entailment operator $\models$ and the derivation operator $\vdash$ are interchangeable.

Theorem 1.16 $\Phi^{\models} = \Phi^{\vdash}$ *for all sets* $\Phi \subseteq \mathsf{PL}$.

1.4 The compactness theorem for propositional logic

Lemma 1.17 *Let* $\Phi \subseteq \mathsf{PL}$ *be a set of propositional formulas, let* $\varphi \in \Phi$ *be a propositional formula. If* $\Phi \vdash \varphi$ *then* $\Phi_0 \vdash \varphi$ *for some finite subset* $\Phi_0 \subseteq \Phi$.

Proof: The proof is by induction on the structure of the set $\Phi^{\vdash}$. Suppose $\varphi \in \Phi^{\vdash}$.

1. *Base case:* $\varphi \in \Phi^{\vdash}$ because $\varphi \in \Phi$. Then, $\varphi \in \{\varphi\}^{\vdash}$ and $\{\varphi\} \subseteq \Phi$ is finite.
2. *Inductive step:* We make case distinctions according to the derivation rules:
   - (CD): $\varphi \in \Phi^{\vdash}$ because $\varphi \in (\Phi \cup \{\psi\})^{\vdash}$ and $\varphi \in (\Phi \cup \{\neg\psi\})^{\vdash}$. By inductive assumption, there are finite sets $\Psi_1 \subseteq \Phi \cup \{\psi\}$ and $\Psi_2 \subseteq \Phi \cup \{\neg\psi\}$ such that $\varphi \in \Psi_1^{\vdash}$ and $\varphi \in \Psi_2^{\vdash}$. Define
   $$\Phi_0 =_{\text{def}} (\Psi_1 \cup \Psi_2) \cap \Phi.$$
   So, $\Phi_0 \subseteq \Phi$ is finite and it holds that
   $$\varphi \in \Psi_1^{\vdash} \subseteq ((\Psi_1 \cap \Phi) \cup \{\psi\})^{\vdash} \subseteq (\Phi_0 \cup \{\psi\})^{\vdash}$$
   $$\varphi \in \Psi_2^{\vdash} \subseteq ((\Psi_2 \cap \Phi) \cup \{\psi\})^{\vdash} \subseteq (\Phi_0 \cup \{\neg\psi\})^{\vdash}$$
   Hence, $\varphi \in \Phi_0^{\vdash}$ by applying the derivation rule (CD).
   - (IP): $\varphi \in \Phi^{\vdash}$ because $\psi, \neg\psi \in (\Phi \cup \{\neg\varphi\})^{\vdash}$. By inductive assumption, there are finite sets $\Psi_1, \Psi_2 \subseteq \Phi \cup \{\neg\varphi\}$ such that $\psi \in \Psi_1^{\vdash}$ and $\neg\psi \in \Psi_2^{\vdash}$. Again, define
   $$\Phi_0 =_{\text{def}} (\Psi_1 \cup \Psi_2) \cap \Phi.$$
   So, $\Phi_0 \subseteq \Phi$ is finite and it holds that
   $$\psi \in \Psi_1^{\vdash} \subseteq ((\Psi_1 \cap \Phi) \cup \{\neg\varphi\})^{\vdash} \subseteq (\Phi_0 \cup \{\neg\varphi\})^{\vdash}$$
   $$\neg\psi \in \Psi_2^{\vdash} \subseteq ((\Psi_2 \cap \Phi) \cup \{\neg\varphi\})^{\vdash} \subseteq (\Phi_0 \cup \{\neg\varphi\})^{\vdash}$$
   Hence, $\varphi \in \Phi_0^{\vdash}$ by applying the derivation rule (IP).

- (MP), (DI), (CI), (CE), (IMP): Analogously.

This proves the lemma. ■

Theorem 1.18 (Compactness theorem for propositional logic) *Let $\Phi \subseteq \mathsf{PL}$ be set of propositional formulas. Then, Φ is satisfiable if and only if each finite subset $\Phi_0 \subseteq \Phi$ is satisfiable.*

Proof: The direction ($\Rightarrow$) holds trivially true. The direction ($\Leftarrow$) is proven by contraposition. So, suppose Φ is not satisfiable. Then, $\Phi \models 0$ (or any other contradiction). By Theorem 1.16, $\Phi \vdash 0$. By Lemma 1.17, there exists a finite set $\Phi_0 \subseteq \Phi$ such that $\Phi_0 \vdash 0$. Again, by Theoremn 1.16, $\Phi_0 \models 0$. Hence, there exists a finite set $\Phi_0 \subseteq \Phi$ that is not satisfiable. This proves the theorem. ■

We discuss some consequences of the compactness theorem for locally finite graphs. A graph $G = (V, E)$ over a finite or countably infinite set of vertices is called *locally finite* if each vertex $v \in V$ is incident with a finite number of edges in E.

Lemma 1.19 (Kőnig's lemma) *Let $T = (V, E)$ be a locally finite tree rooted at vertex $w \in V$. If T contains paths of arbitrary length then T contains an infinite path (starting at w).*

Proof: Let $T = (V, E)$ be a locally finite tree rooted at $w \in V$. Without loss of generality, we consider the tree oriented away from root w (i.e., there is no edge pointing to w and for all vertices $v \in V \setminus \{w\}$ there is exactly one directed edge $(u, v) \in E$). Define for $n \in \mathbb{N}$

$$S_n =_{\text{def}} \{\ v \in V \mid d_T(w, v) = n\ \},$$

where $d_T(w, v)$ denotes the length of a shortest (w, v)-path (in fact, the only (w, v)-path) on T. Since T is locally finite, S_n is finite for all $n \in \mathbb{N}$. Furthermore, $S_0 = \{w\}$ and $S_n \neq \emptyset$ since T contains arbitrarily long paths.

An infinite path in T corresponds to some subset $W \subseteq V$ satisfying

- $\|W \cap S_n\| = 1$ for all $n \in \mathbb{N}$ and
- if $v \in W$ and $(u, v) \in E$ then $u \in W$.

We describe such a set using an infinite set of propositional formulas. We assign a propositional variable X_v to each vertex $v \in V$, define formulas

$$\alpha_n =_{\text{def}} \bigvee_{v \in S_n} X_v \qquad \text{(to check for } \|W \cap S_n\| \geq 1\text{)}$$

$$\beta_n =_{\text{def}} \bigwedge_{\substack{u,v \in S_n \\ u \neq v}} \neg(X_u \wedge X_v) \qquad \text{(to check for } \|W \cap S_n\| \leq 1\text{)}$$

and, finally, define the set Φ of formulas as follows:

$$\Phi =_{\text{def}} \{ \alpha_n \mid n \in \mathbb{N} \} \cup \{ \beta_n \mid n \in \mathbb{N} \} \cup \{ (X_v \to X_u) \mid (u,v) \in E \}$$

It is easily seen that each finite subset $\Phi_0 \subseteq \Phi$ is satisfiable: Let $n \in \mathbb{N}$ be maximum subject to $a_n \in \Phi_0$ or $\beta_n \in \Phi_0$. Choose some $z \in S_n$. Then, there exists a (w,z)-path in T. Consider the assignment I fulfilling $I(X_v) = 1 \iff v$ lies on the (w,z)-path. Clearly, I is model of Φ_0. By Theorem 1.18, there exists a model I of Φ. Define $W \subseteq V$ to be the following vertex set:

$$W =_{\text{def}} \{ v \in V \mid I(X_v) = 1 \}$$

We conclude that W corresponds to an infinite path in T: Since $\alpha_n, \beta_n \in \Phi$, there is exactly one vertex $v \in W \cap S_n$ for all $n \in \mathbb{N}$. Moreover, we obtain for $v \in W$ and $(u,v) \in W$

$$[\![X_v \to X_u]\!]^I = 1, \quad [\![X_v]\!]^I = 1; \qquad \text{thus,} \quad [\![X_u]\!]^I = 1.$$

Hence, $u \in W$. This completes the proof of the lemma. ∎

Kőnig's lemma is helpful in extending results from finite graphs to locally finite graphs. As an example, consider Hall's marriage theorem. A *matching* M of an undirected, finite or infinite graph $G = (V,E)$ is a subset $M \subseteq E$ of edges such that $e \cap f = \emptyset$ for all distinct edges $e, f \in M$. Given a finite or infinite, bipartite graph $G = (A \uplus B, E)$, a matching $M \subseteq E$ is *perfect* for A if and only if each vertex $v \in A$ is incident with an edge in M and each edge $e \in M$ is incident with a vertex in A, i.e., M covers exactly A.

Theorem 1.20 *Let $G = (A \uplus B, E)$ be a locally finite, bipartite graph. There exists a matching in G which is perfect for A if and only if $\|S\| \leq \|N_G(S)\|$ for all subsets $S \subseteq A$.*

Proof: Obviously, we only have to show the direction ($\Leftarrow$). Let $G = (A \uplus B, E)$ be a locally finite, bipartite graph. If A is finite then the statement reduces to the usual Hall's marriage theorem. So, let A be countably infinite; without loss of generality, $A = \{1, 2, \ldots\} = \mathbb{N}_+$. In order to apply Lemma 1.19, we define an appropriate tree $T = (V_T, E_T)$ as follows:

$$V_T =_{\text{def}} \{ M \subseteq E \mid M \text{ is matching of size } n \text{ covering } \{1, \ldots, n\} \text{ for some } n \in \mathbb{N} \}$$
$$E_T =_{\text{def}} \{ (M, M') \mid \|M'\| = \|M\| + 1 \text{ and } M \subseteq M' \}$$

That is, there is an edge between two matchings M and M' if M' extends M via a new edge for vertex $n+1$. Clearly, T is a tree rooted at $\emptyset$. Since G is locally finite, T is locally finite. Moreover, T has paths of arbitrary length. To see this, consider hypothetical paths of length n starting at the root. All such paths would lead to vertices representing matchings of size n. The induced graph $G[\{1, \ldots, n\} \uplus B]$ is finite and thus, the existence of a matching perfect for $\{1, \ldots, n\}$ follows from the usual Hall's marriage theorem. Hence, there is a vertex reachable via a path of length n. Now, Lemma 1.19 implies the existence of a matching perfect for A. This proves the theorem. ∎

Note that the theorem still holds for "one-sided" locally finite, bipartite graphs; it is, however, not true for arbitrary countably infinite graphs.

1.5 Resolution

Resolution is a principle applicable to test if a propositional formula in CNF is not satisfiable. As the satisfiability problem for CNF is NP-complete, the resolution principle cannot lead to a polynomial algorithm for testing non-satisfiability (unless P = NP). Nevertheless, in many cases it shows an efficient behavior.

For sake of convenience, formulas are identified with sets:

- A *clause* is a finite set of literals.
- The empty clause is denoted by $\Box$.
- A set $K(\varphi)$ of clauses is assigned to a given CNF $\varphi = \bigwedge_{i=1}^{n} \bigvee_{j=1}^{n_i} L_{ij}$:

$$K(\varphi) =_{\text{def}} \{ \{ L_{ij} \mid j \in \{1, \ldots, n_i\} \} \mid i \in \{1, \ldots, n\} \}$$

  Note that multiplicities and the ordering of literals and clauses are neglected.

We extend the semantics from formulas to set of clauses. Let $I : \tau \to \{0, 1\}$ be an assignment. For a clause C and a set K of clauses, we define:

$$\begin{aligned} [\![C]\!]^I = 1 &\iff_{\text{def}} [\![L]\!]^I = 1 \text{ for some literal } L \in C \\ [\![K]\!]^I = 1 &\iff_{\text{def}} [\![C]\!]^I = 1 \text{ for all clauses } C \in K \end{aligned}$$

In particular, $[\![\varphi]\!]^I = [\![K(\varphi)]\!]^I$ for all assignments I. Related notions, such as equivalence ($\equiv$) or satisfiability, translate from formulas to sets. Note that the empty set of clauses is satisfiable and the empty clause $\Box$ is not satisfiable.

Definition 1.21 *Let C, C_1 and C_2 be a clauses. Then, C is said to be a* resolvent *of C_1 and C_2 if and only if there is a variable X such that $X \in C_1$ and $\neg X \in C_2$ and it holds that*

$$C = C_1 \setminus \{X\} \cup C_2 \setminus \{\neg X\}.$$

Example:

- $\{X_1, \neg X_3, X_4\}$ is the resolvent of $\{X_1, \neg X_2, \neg X_3\}$ and $\{X_2, X_4\}$.
- $\{X_1, X_3, \neg X_3, X_4\}$ and $\{X_1, \neg X_2, X_2, X_4\}$ are resolvents of $\{X_1, \neg X_2, X_3\}$ and $\{X_2, \neg X_3 X_4\}$.
- The clause $\Box$ is the resolvent of $\{X_1\}$ and $\{\neg X_1\}$.

Lemma 1.22 *Let K be any set of clauses and let $C_1, C_2 \in K$ be clauses. Suppose C is a resolvent of C_1 and C_2. Then, $K \equiv K \cup \{C\}$.*

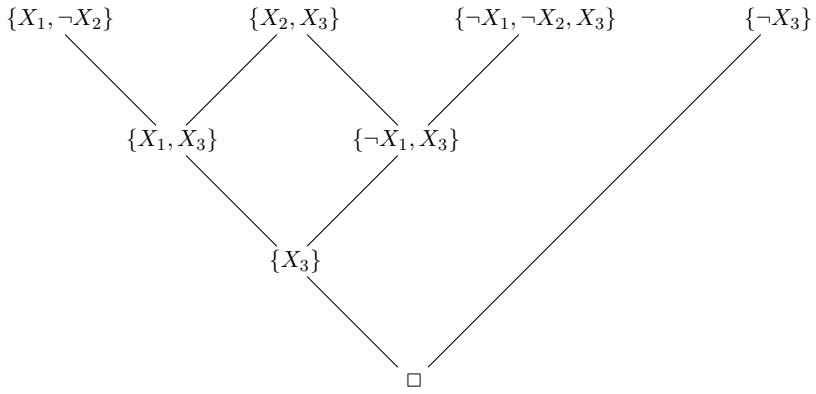
Proof: Let $I : \tau \to \{0,1\}$ be an assignment. Clearly, $[\![K \cup \{C\}]\!]^I = 1$ implies $[\![K]\!]^I = 1$. Now, suppose $[\![K]\!]^I = 1$ and $C = C_1 \setminus \{X\} \cup C_2 \setminus \{\neg X\}$. We have two cases:

1. If $[\![X]\!]^I = 1$ then $[\![C_2 \setminus \{\neg X\}]\!]^I = 1$ (otherwise $[\![C_2]\!]^I = 0$). Thus, $[\![C]\!]^I = 1$.
2. If $[\![X]\!]^I = 0$ then $[\![C_1 \setminus \{X\}]\!]^I = 1$ (otherwise $[\![C_1]\!]^I = 0$). Thus, $[\![C]\!]^I = 1$.

Hence, $[\![K \cup \{C\}]\!]^I = 1$. This proves the lemma. ■

In the light of the last example above, namely that $\square$ is the resolvent of $\{X_1\}$ and $\{\neg X_1\}$, Lemma 1.22 can be iteratively applied to derive the empty clause as a resolvent.

Example: Consider $K = \{\ \{X_1, \neg X_2\}, \{\neg X_3\}, \{\neg X_1, \neg X_2, X_3\}, \{X_2, X_3\}\ \}$. A proof that $\square$ is a resolvent can be represented graphically as follows:



Hence, K is not satisfiable.

This gives rise to an algorithm for non-satisfiablitiy based on iteratively determining all resolvents of a set of clause. Let K be any set of clauses. Then,

$$\mathrm{Res}(K) =_{\mathrm{def}} K \cup \{\ C \mid C \text{ is a resolvent of some clauses } C_1, C_2 \in K\ \}.$$

After n iterations the set of clauses is given by

$$\mathrm{Res}^n(K) =_{\mathrm{def}} \mathrm{Res}\left(\mathrm{Res}^{n-1}(K)\right) \text{ for } n > 0; \qquad \mathrm{Res}^0(K) =_{\mathrm{def}} K.$$

In the end, we compute: $\mathrm{Res}^*(K) =_{\mathrm{def}} \bigcup_{n \in \mathbb{N}} \mathrm{Res}^n(K)$

Proposition 1.23 *Let K be a set of clauses. Then, $K \equiv \mathrm{Res}^*(K)$.*

Proof: Inductively apply Lemma 1.22. ■

Theorem 1.24 (Resolution theorem) *Let K be any set of clauses. K is not satisfiable if and only if $\Box \in \mathrm{Res}^*(K)$.*

Proof: We prove both directions individually:

- ($\Leftarrow$): $\Box$ is not satisfiable. Thus, if $\Box \in \mathrm{Res}^*(K)$ then $\mathrm{Res}^*(K) \equiv K$ is not satisfiable.
- ($\Rightarrow$): Suppose K is not satisfiable. The proof is by induction on the number n of variables occurring as literals in the clauses of K.
  1. *Base case* $n = 0$: Thus, either $K = \emptyset$ which is satisfiable or $K = \{\Box\}$ which is not satisfiable. Hence, $\Box \in K \subseteq \mathrm{Res}^*(K)$.
  2. *Inductive step* $n > 0$: Choose a variable X occurring in K. Define the following two sets of clauses:

     $$K_+ =_{\text{def}} \{\, C \backslash \{\neg X\} \mid X \notin C, C \in K \,\}, \quad K_- =_{\text{def}} \{\, C \backslash \{X\} \mid \neg X \notin C, C \in K \,\}$$

     Then, K_+ and K_- contain only $n-1$ variables and both are not satisfiable. By inductive assumption, $\Box \in \mathrm{Res}^*(K_+)$ and $\Box \in \mathrm{Res}^*(K_-)$. That is, there exist clauses $C_1, C_2, \ldots, C_m$ such that $C_m = \Box$ and for all $i \in \{1, \ldots, m-1\}$, $C_i \in K_+$ or C_i is a resolvent of C_j and C_k, where $j, k \in \{1, \ldots, i-1\}$. Two cases are possible:
     (a) Some clauses C_i are obtained by removing $\neg X$: When re-inserting $\neg X$ appropriately, there exists thus a sequence of clauses $C'_1, \ldots, C'_m$ such that $C'_m = \{\neg X\}$ and for all $i \in \{1, \ldots, m-1\}$, $C'_i \in K$ and C'_i is a resolvent of C'_j, C'_k for $j, k \in \{1, \ldots, i-1\}$.
     (b) All clauses C_i are obtained without removing $\neg X$: That is, $C_i \in \mathrm{Res}^*(K)$ for all $i \in \{1, \ldots, m\}$.

     Hence, either $\{\neg X\} \in \mathrm{Res}^*(K)$ or $\Box \in \mathrm{Res}^*(K)$. Analogously, we obtain that either $\{X\} \in \mathrm{Res}^*(K)$ or $\Box \in \mathrm{Res}^*(K)$. Since $\Box$ is the resolvent of $\{X\}$ and $\{\neg X\}$, we conclude $\Box \in \mathrm{Res}^*(K)$.

This proves the theorem. ■

Remark: Using the compactness theorem, Theorem 1.24 can also be proven for countably infinite sets K of clauses.

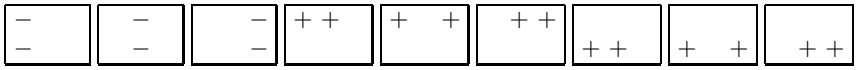
Complexity: The complexity of resolution depends on the number of resolvents. Given n variables, there are at most 2^{2n} clauses as candidates for resolvents. Thus, resolution is an exponential algorithm (but not worse).

A set of formulas that actually forces an exponential number of resolvents is known as the *pigeonhole formulas* PF_n. It is best described by variables organized in matrices filled with $-$, $+$, and blank symbols:

- $-$ at position (i, j) means that $\neg X_{i,j}$ is in the clause,
- $+$ at position (i, j) means that $X_{i,j}$ is in the clause, and
- blank at position (i, j) means that $X_{i,j}$ is neither positively nor negatively contained in the clause.

PF_n corresponds to the set of all $n \times (n+1)$-matrices that have either exactly one column consisting only of $-$'s or exactly one row with exactly two $+$'s.

Example: PF_2 consists of the following matrices:



That is,

$$\begin{aligned}\mathsf{PF}_2 =_{\text{def}} \quad \{ \quad & \{\neg X_{1,1}, \neg X_{2,1}\},\ \{\neg X_{1,2}, \neg X_{2,2}\},\ \{\neg X_{1,3}, \neg X_{2,3}\},\\ & \{X_{1,1}, X_{1,2}\},\ \{X_{1,1}, X_{1,3}\},\ \{X_{1,2}, X_{1,3}\},\\ & \{X_{2,1}, X_{2,2}\},\ \{X_{2,1}, X_{2,3}\},\ \{X_{2,2}, X_{2,3}\} \ \}\end{aligned}$$

Note that PF_n is not satisfiable: Assume I is a satisfying assignment. So, in each row i at most one positive literal $X_{i,j}$ is allowed to be $I(X_{i,j}) = 0$. Thus, there are at most n such variables but there are $n + 1$ columns. By the pigeonhole principle, there is a column, i.e., a clause, consisting only of negative literals $\neg X_{1,j}, \ldots, \neg X_{n,j}$ such that $I(X_{1,j}) = \cdots = I(X_{n,j}) = 1$. Hence, I cannot be a satisfying assinment for PF_n.

Though PF_n consists of $\Theta(n^3)$ clauses, it can be shown that there are c^n, $c > 1$, resolvents of PF_n.

Bibliography

[Ben12] Mordechai Ben-Ari. *Mathematical Logic for Computer Science.* 3rd edition, Springer, London, 2012.

[EF05] Heinz-Dieter Ebbinghaus, Jörg Flum. *Finite Model Theory.* 2nd edition, Springer-Verlag, Berlin, 2005.

[EFT94] Heinz-Dieter Ebbinghaus, Jörg Flum, Wolfgang Thomas. *Mathematical Logic.* 2nd edition, Springer-Verlag, New York, NY, 1994.

German edition: Einführung in die Mathematische Logik. 5. Auflage, Spektrum Akademischer Verlag, 2007.

[HR04] Michael Huth, Mark Ryan. *Logic in Computer Science. Modelling and Reasoning about Systems.* 2nd edition, Cambridge University Press, Cambridge, 2004.

[Rau05] Wolfgang Rautenberg. *Einführung in die Mathematische Logik. Ein Lehrbuch.* 3., überarbeitete Auflage, Vieweg + Teubner, Wiesbaden, 2008.