Fundamentals of theory of computation 2

1st lecture

lecturer: Tichler Krisztián
ktichler@inf.elte.hu

Brief syllabus

- Two models of mathematical logic. Propositional logic and first order logic.
- Asympthotical properties of functions.
- Turing machines (TM) as a model of algorithms
- Multitape TM, nondeterministic TM, counting TM
- Cardinality of infinite sets.
- Algorithmic language classes RE,R and their properties
- Undecidable problems, reduction of a language (problem) to another language
- Algorithmic vs. Chomsky language classes.
- Basic concepts of complexity theory, time complexity classes, NP-completeness, polynomial time reduction
- NP-complete problems (SAT, graph problems, ...)
- Offline TM, space complexity.

Two models of mathematical logic

Basic concept: atomic **proposition** (or **statement**): either **true** or **false** independent of context. **true** and **false** are called **truth values**. Compound propositions consist of atomic propositions and linguistic connectors corresponding to Boolean operators.

Two models of mathematical logic

Basic concept: atomic **proposition** (or **statement**): either **true** or **false** independent of context. **true** and **false** are called **truth values**. Compound propositions consist of atomic propositions and linguistic connectors corresponding to Boolean operators.

I. Propositional logic (propositional calculus, zeroth-order logic)

Two models of mathematical logic

Basic concept: atomic **proposition** (or **statement**): either **true** or **false** independent of context. **true** and **false** are called **truth values**. Compound propositions consist of atomic propositions and linguistic connectors corresponding to Boolean operators.

I. Propositional logic (propositional calculus, zeroth-order logic)

The formulas are built from atomic propositions (variables).
Internal structure of atomic propositions are not considered.
Formulas can be combined using Boolean operators (and, or, etc.).
Atomic propositions can be either true or false.

Two models of mathematical logic

Basic concept: atomic **proposition** (or **statement**): either **true** or **false** independent of context. **true** and **false** are called **truth values**. Compound propositions consist of atomic propositions and linguistic connectors corresponding to Boolean operators.

I. Propositional logic (propositional calculus, zeroth-order logic)

The formulas are built from atomic propositions (variables).
Internal structure of atomic propositions are not considered.
Formulas can be combined using Boolean operators (and, or, etc.).
Atomic propositions can be either true or false.

II. First-order logic (predicate logic, predicate calculus)

Two models of mathematical logic

Basic concept: atomic **proposition** (or **statement**): either **true** or **false** independent of context. **true** and **false** are called **truth values**. Compound propositions consist of atomic propositions and linguistic connectors corresponding to Boolean operators.

I. Propositional logic (propositional calculus, zeroth-order logic)

The formulas are built from atomic propositions (variables). *Internal structure* of atomic propositions are not considered. Formulas can be combined using Boolean operators (and, or, etc.). Atomic propositions can be either true or false.

II. First-order logic (predicate logic, predicate calculus)

Statements have inner structure. Allows the use of sentences that contain variables and symbols that can be interpreted as functions and relations. Variables range over a domain. Formulas can be combined using Boolean operators (and, or, etc.) and quantifiers.

Two models of mathematical logic

Basic concept: atomic **proposition** (or **statement**): either **true** or **false** independent of context. **true** and **false** are called **truth values**. Compound propositions consist of atomic propositions and linguistic connectors corresponding to Boolean operators.

I. Propositional logic (propositional calculus, zeroth-order logic)

The formulas are built from atomic propositions (variables). *Internal structure* of atomic propositions are not considered. Formulas can be combined using Boolean operators (and, or, etc.). Atomic propositions can be either true or false.

II. First-order logic (predicate logic, predicate calculus)

Statements have inner structure. Allows the use of sentences that contain variables and symbols that can be interpreted as functions and relations. Variables range over a domain. Formulas can be combined using Boolean operators (and, or, etc.) and quantifiers.

First order logic has more expressive power but a less simple model.

Propositional logic

Formal syntax

Definition

Let $\mathcal{P} = \{p, q, r, \ldots\}$ be a (countably) infinite set, its elements are called **atoms** (also called: atomic propositions, variables, atomic variables).

Propositional logic

Formal syntax

Definition

Let $\mathcal{P} = \{p, q, r, \ldots\}$ be a (countably) infinite set, its elements are called **atoms** (also called: atomic propositions, variables, atomic variables).

We define an **alphabet** (the set of terminals) as follows:

Propositional logic

Formal syntax

Definition

Let $\mathcal{P} = \{p, q, r, \ldots\}$ be a (countably) infinite set, its elements are called **atoms** (also called: atomic propositions, variables, atomic variables).

We define an **alphabet** (the set of terminals) as follows:

- elements of $\mathcal{P}$,

Propositional logic

Formal syntax

Definition

Let $\mathcal{P} = \{p, q, r, \ldots\}$ be a (countably) infinite set, its elements are called **atoms** (also called: atomic propositions, variables, atomic variables).

We define an **alphabet** (the set of terminals) as follows:

- elements of $\mathcal{P}$,
- Boolean operators:

Propositional logic

Formal syntax

Definition

Let $\mathcal{P} = \{p, q, r, \ldots\}$ be a (countably) infinite set, its elements are called **atoms** (also called: atomic propositions, variables, atomic variables).

We define an **alphabet** (the set of terminals) as follows:

- elements of $\mathcal{P}$,
- Boolean operators:
  negation $\neg$

Propositional logic

Formal syntax

Definition

Let $\mathcal{P} = \{p, q, r, \ldots\}$ be a (countably) infinite set, its elements are called **atoms** (also called: atomic propositions, variables, atomic variables).

We define an **alphabet** (the set of terminals) as follows:

- elements of $\mathcal{P}$,
- Boolean operators:

  negation $\neg$ conjunction $\wedge$

Propositional logic

Formal syntax

Definition

Let $\mathcal{P} = \{p, q, r, \dots\}$ be a (countably) infinite set, its elements are called **atoms** (also called: atomic propositions, variables, atomic variables).

We define an **alphabet** (the set of terminals) as follows:

- elements of $\mathcal{P}$,
- Boolean operators:

negation	$\neg$	conjunction	$\wedge$
disjunction	$\vee$		

Propositional logic

Formal syntax

Definition

Let $\mathcal{P} = \{p, q, r, \ldots\}$ be a (countably) infinite set, its elements are called **atoms** (also called: atomic propositions, variables, atomic variables).

We define an **alphabet** (the set of terminals) as follows:

- elements of $\mathcal{P}$,
- Boolean operators:

negation	$\neg$	conjunction	$\wedge$
disjunction	$\vee$	implication	$\rightarrow$

Propositional logic

Formal syntax

Definition

Let $\mathcal{P} = \{p, q, r, \ldots\}$ be a (countably) infinite set, its elements are called **atoms** (also called: atomic propositions, variables, atomic variables).

We define an **alphabet** (the set of terminals) as follows:

- elements of $\mathcal{P}$,
- Boolean operators:

negation	$\neg$	conjunction	$\wedge$
disjunction	$\vee$	implication	$\rightarrow$

- (,) (in the case of a string representation).

Propositional logic

Formal syntax

Definition

Let $\mathcal{P} = \{p, q, r, \ldots\}$ be a (countably) infinite set, its elements are called **atoms** (also called: atomic propositions, variables, atomic variables).

We define an **alphabet** (the set of terminals) as follows:

- elements of $\mathcal{P}$,
- Boolean operators:

negation	$\neg$	conjunction	$\wedge$
disjunction	$\vee$	implication	$\rightarrow$

- (,) (in the case of a string representation).

Note: sometimes further Boolean operators are used as well, such as equivalence ($\leftrightarrow$), exclusive or ($\oplus$), nor ($\downarrow$), nand ($\uparrow$).

Syntax of propositional logic

The tree representation of formulas

Definition

A **formula** is one of the following rooted node-labelled binary trees

Syntax of propositional logic

The tree representation of formulas

Definition

A **formula** is one of the following rooted node-labelled binary trees

- a single node (root) labelled by an atom

Syntax of propositional logic

The tree representation of formulas

Definition

A **formula** is one of the following rooted node-labelled binary trees

- a single node (root) labelled by an atom
- a node (root) labelled by $\neg$ with a single child that is a formula

Syntax of propositional logic

The tree representation of formulas

Definition

A **formula** is one of the following rooted node-labelled binary trees

- a single node (root) labelled by an atom
- a node (root) labelled by $\neg$ with a single child that is a formula
- a node (root) labelled by one of the binary operations with two children both of which are formulas

Syntax of propositional logic

The tree representation of formulas

Definition

A **formula** is one of the following rooted node-labelled binary trees

- a single node (root) labelled by an atom
- a node (root) labelled by $\neg$ with a single child that is a formula
- a node (root) labelled by one of the binary operations with two children both of which are formulas

Definition

A **(proper) subformula** is a (proper) subtree.

Syntax of propositional logic

The tree representation of formulas

Definition

A **formula** is one of the following rooted node-labelled binary trees

- a single node (root) labelled by an atom
- a node (root) labelled by $\neg$ with a single child that is a formula
- a node (root) labelled by one of the binary operations with two children both of which are formulas

Definition

A **(proper) subformula** is a (proper) subtree.

Definition

Principal operator of a formula is the operator at the root of the tree.

Syntax of propositional logic

The tree representation of formulas

Definition

A **formula** is one of the following rooted node-labelled binary trees

- a single node (root) labelled by an atom
- a node (root) labelled by $\neg$ with a single child that is a formula
- a node (root) labelled by one of the binary operations with two children both of which are formulas

Definition

A **(proper) subformula** is a (proper) subtree.

Definition

Principal operator of a formula is the operator at the root of the tree.

(Formulas, that are atoms have no principal operator).

Syntax of propositional logic

The string representation of formulas

Input: a formula F given by its tree representation

Syntax of propositional logic

The string representation of formulas

Input: a formula F given by its tree representation
Output: string representation of F

Syntax of propositional logic

The string representation of formulas

Input: a formula F given by its tree representation
Output: string representation of F

Algorithm $\textsc{Inorder}(F)$

---

if F is a leaf **then**
    write its label
    return
let F_1 and F_2 be the left and right subtrees of F
write a left parenthesis '('
if the label of the root is not $\neg$ **then**
    $\textsc{Inorder}(F_1)$
write the label of the root of F
$\textsc{Inorder}(F_2)$
write a right parenthesis ')'

Syntax of propositional logic

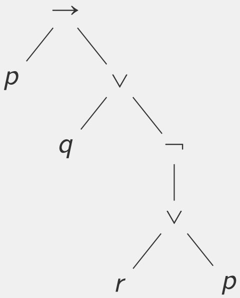
The need of parenthesis

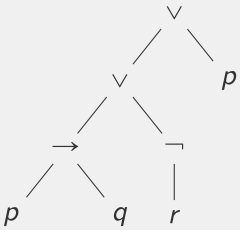
Example:

Syntax of propositional logic

The need of parenthesis

Example:

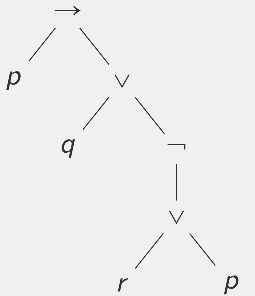




Syntax of propositional logic

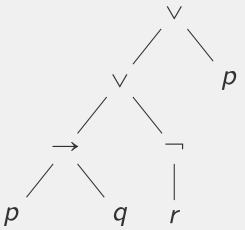
The need of parenthesis

Example:



String representation:

$(p \rightarrow (q \vee (\neg(r \vee p))))$



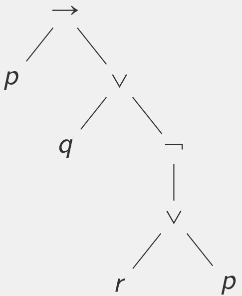
String representation:

$(((p \rightarrow q) \vee (\neg r)) \vee p)$

Syntax of propositional logic

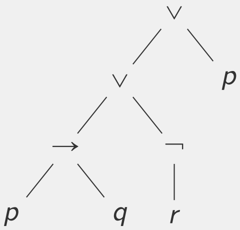
The need of parenthesis

Example:



String representation:

$(p \rightarrow (q \vee (\neg(r \vee p))))$



String representation:

$(((p \rightarrow q) \vee (\neg r)) \vee p)$

Without parenthesis there would be an ambiguity as both string representation were $p \rightarrow q \vee \neg r \vee p$.

Syntax of propositional logic

Leaving parenthesis

Another way to resolve ambiguous formulas is to define precedence and associativity conventions.

Syntax of propositional logic

Leaving parenthesis

Another way to resolve ambiguous formulas is to define precedence and associativity conventions.

Analogy: arithmetic expression, e.g., $3 + 8 \cdot 7$.

Order of precedence from high to low: $\neg$, $\wedge$, $\vee$, $\rightarrow$.

Syntax of propositional logic

Leaving parenthesis

Another way to resolve ambiguous formulas is to define precedence and associativity conventions.

Analogy: arithmetic expression, e.g., $3 + 8 \cdot 7$.

Order of precedence from high to low: $\neg$, $\wedge$, $\vee$, $\rightarrow$.

Operators are assumed to associate to the right, that is, $A \vee B \vee C$ means $(A \vee (B \vee C))$.

Syntax of propositional logic

Leaving parenthesis

Another way to resolve ambiguous formulas is to define precedence and associativity conventions.

Analogy: arithmetic expression, e.g., $3 + 8 \cdot 7$.

Order of precedence from high to low: $\neg$, $\wedge$, $\vee$, $\rightarrow$.

Operators are assumed to associate to the right, that is, $A \vee B \vee C$ means $(A \vee (B \vee C))$.

Parentheses are used only if needed to indicate an order different from that imposed by the rules for precedence and associativity.

Syntax of propositional logic

Leaving parenthesis

Another way to resolve ambiguous formulas is to define precedence and associativity conventions.

Analogy: arithmetic expression, e.g., $3 + 8 \cdot 7$.

Order of precedence from high to low: $\neg$, $\wedge$, $\vee$, $\rightarrow$.

Operators are assumed to associate to the right, that is, $A \vee B \vee C$ means $(A \vee (B \vee C))$.

Parentheses are used only if needed to indicate an order different from that imposed by the rules for precedence and associativity.

Examples, minimum number of parenthesis:

Syntax of propositional logic

Leaving parenthesis

Another way to resolve ambiguous formulas is to define precedence and associativity conventions.

Analogy: arithmetic expression, e.g., $3 + 8 \cdot 7$.

Order of precedence from high to low: $\neg$, $\wedge$, $\vee$, $\rightarrow$.

Operators are assumed to associate to the right, that is, $A \vee B \vee C$ means $(A \vee (B \vee C))$.

Parentheses are used only if needed to indicate an order different from that imposed by the rules for precedence and associativity.

Examples, minimum number of parenthesis:

$(p \rightarrow (q \vee (\neg(r \vee p))))$

Syntax of propositional logic

Leaving parenthesis

Another way to resolve ambiguous formulas is to define precedence and associativity conventions.

Analogy: arithmetic expression, e.g., $3 + 8 \cdot 7$.

Order of precedence from high to low: $\neg,\ \wedge,\ \vee,\ \rightarrow$.

Operators are assumed to associate to the right, that is, $A \vee B \vee C$ means $(A \vee (B \vee C))$.

Parentheses are used only if needed to indicate an order different from that imposed by the rules for precedence and associativity.

Examples, minimum number of parenthesis:

$$(p \rightarrow (q \vee (\neg(r \vee p)))) \qquad \Rightarrow \qquad p \rightarrow q \vee \neg(r \vee p)$$

Syntax of propositional logic

Leaving parenthesis

Another way to resolve ambiguous formulas is to define precedence and associativity conventions.

Analogy: arithmetic expression, e.g., $3 + 8 \cdot 7$.

Order of precedence from high to low: $\neg$, $\wedge$, $\vee$, $\rightarrow$.

Operators are assumed to associate to the right, that is, $A \vee B \vee C$ means $(A \vee (B \vee C))$.

Parentheses are used only if needed to indicate an order different from that imposed by the rules for precedence and associativity.

Examples, minimum number of parenthesis:

$(p \rightarrow (q \vee (\neg(r \vee p)))) \qquad \Rightarrow \qquad p \rightarrow q \vee \neg(r \vee p)$

$(((p \rightarrow q) \vee (\neg r)) \vee p)$

Syntax of propositional logic

Leaving parenthesis

Another way to resolve ambiguous formulas is to define precedence and associativity conventions.

Analogy: arithmetic expression, e.g., $3 + 8 \cdot 7$.

Order of precedence from high to low: $\neg$, $\wedge$, $\vee$, $\rightarrow$.

Operators are assumed to associate to the right, that is, $A \vee B \vee C$ means $(A \vee (B \vee C))$.

Parentheses are used only if needed to indicate an order different from that imposed by the rules for precedence and associativity.

Examples, minimum number of parenthesis:

$$(p \rightarrow (q \vee (\neg(r \vee p)))) \quad \Rightarrow \quad p \rightarrow q \vee \neg(r \vee p)$$
$$(((p \rightarrow q) \vee (\neg r)) \vee p) \quad \Rightarrow \quad ((p \rightarrow q) \vee \neg r) \vee p$$

Semantics of propositional logic

Interpretation

Analogy: arithmetic expressions. E.g., $y = a + 2 \cdot b$.

Semantics of propositional logic

Interpretation

Analogy: arithmetic expressions. E.g., $y = a + 2 \cdot b$. Assign values to a and b then evaluate y.

Semantics of propositional logic

Interpretation

Analogy: arithmetic expressions. E.g., $y = a + 2 \cdot b$. Assign values to a and b then evaluate y.

Definition

Let A be a formula and let P_A be the set of atoms appearing in A. An **interpretation** for A is a total function $I_A : P_A \rightarrow \{T, F\}$ that assigns one of the truth values T or F to every atom in P_A.

Semantics of propositional logic

Interpretation

Analogy: arithmetic expressions. E.g., $y = a + 2 \cdot b$. Assign values to a and b then evaluate y.

Definition

Let A be a formula and let P_A be the set of atoms appearing in A. An **interpretation** for A is a total function $I_A : P_A \rightarrow \{T, F\}$ that assigns one of the truth values T or F to every atom in P_A.

We will use the short notation I instead of I_A whenever the formula is clear from the context.

Semantics of propositional logic

Interpretation

Analogy: arithmetic expressions. E.g., $y = a + 2 \cdot b$. Assign values to a and b then evaluate y.

Definition

Let A be a formula and let P_A be the set of atoms appearing in A. An **interpretation** for A is a total function $I_A : P_A \rightarrow \{T, F\}$ that assigns one of the truth values T or F to every atom in P_A.

We will use the short notation I instead of I_A whenever the formula is clear from the context. Example:

$A = p \rightarrow q \vee \neg(r \vee p)$.

Semantics of propositional logic

Interpretation

Analogy: arithmetic expressions. E.g., $y = a + 2 \cdot b$. Assign values to a and b then evaluate y.

Definition

Let A be a formula and let P_A be the set of atoms appearing in A. An **interpretation** for A is a total function $I_A : P_A \rightarrow \{T, F\}$ that assigns one of the truth values T or F to every atom in P_A.

We will use the short notation I instead of I_A whenever the formula is clear from the context. Example:

$A = p \rightarrow q \vee \neg(r \vee p)$. $P_A = \{p, q, r\}$.

Semantics of propositional logic

Interpretation

Analogy: arithmetic expressions. E.g., $y = a + 2 \cdot b$. Assign values to a and b then evaluate y.

Definition

Let A be a formula and let P_A be the set of atoms appearing in A. An **interpretation** for A is a total function $I_A : P_A \rightarrow \{T, F\}$ that assigns one of the truth values T or F to every atom in P_A.

We will use the short notation I instead of I_A whenever the formula is clear from the context. Example:

$A = p \rightarrow q \vee \neg(r \vee p)$. $P_A = \{p, q, r\}$. One possibility is the following: $I(p) = T,\ I(q) = F,\ I(r) = F$.

Semantics of propositional logic

Interpretation

Analogy: arithmetic expressions. E.g., $y = a + 2 \cdot b$. Assign values to a and b then evaluate y.

Definition

Let A be a formula and let P_A be the set of atoms appearing in A. An **interpretation** for A is a total function $I_A : P_A \to \{T, F\}$ that assigns one of the truth values T or F to every atom in P_A.

We will use the short notation I instead of I_A whenever the formula is clear from the context. Example:

$A = p \to q \vee \neg(r \vee p)$. $P_A = \{p, q, r\}$. One possibility is the following: $I(p) = T,\ I(q) = F,\ I(r) = F$.
There are 8 possibilities for I in this case.

Semantics of propositional logic

Evaluation of a formula

Definition

Let I be an interpretation for a formula A. $v_I(A)$, the **truth value of A under I** is defined recursively as follows:

Semantics of propositional logic

Evaluation of a formula

Definition

Let I be an interpretation for a formula A. $v_I(A)$, the **truth value of A under I** is defined recursively as follows:

$v_I(A) = I(A)$	if A is an atom

Semantics of propositional logic

Evaluation of a formula

Definition

Let I be an interpretation for a formula A. $v_I(A)$, the **truth value of A under I** is defined recursively as follows:

$v_I(A) = I(A)$	if A is an atom
$v_I(\neg A) = T$	if $v_I(A) = F$
$v_I(\neg A) = F$	if $v_I(A) = T$

Semantics of propositional logic

Evaluation of a formula

Definition

Let I be an interpretation for a formula A. $v_I(A)$, the **truth value of A under I** is defined recursively as follows:

$v_I(A) = I(A)$	if A is an atom
$v_I(\neg A) = T$	if $v_I(A) = F$
$v_I(\neg A) = F$	if $v_I(A) = T$
$v_I(A_1 \wedge A_2) = T$	if $v_I(A_1) = T$ and $v_I(A_2) = T$
$v_I(A_1 \wedge A_2) = F$	in the other three cases

Semantics of propositional logic

Evaluation of a formula

Definition

Let I be an interpretation for a formula A. $v_I(A)$, the **truth value of A under I** is defined recursively as follows:

$v_I(A) = I(A)$	if A is an atom
$v_I(\neg A) = T$	if $v_I(A) = F$
$v_I(\neg A) = F$	if $v_I(A) = T$
$v_I(A_1 \wedge A_2) = T$	if $v_I(A_1) = T$ and $v_I(A_2) = T$
$v_I(A_1 \wedge A_2) = F$	in the other three cases
$v_I(A_1 \vee A_2) = F$	if $v_I(A_1) = F$ and $v_I(A_2) = F$
$v_I(A_1 \vee A_2) = T$	in the other three cases

Semantics of propositional logic

Evaluation of a formula

Definition

Let I be an interpretation for a formula A. $v_I(A)$, the **truth value of A under I** is defined recursively as follows:

$v_I(A) = I(A)$	if A is an atom
$v_I(\neg A) = T$	if $v_I(A) = F$
$v_I(\neg A) = F$	if $v_I(A) = T$
$v_I(A_1 \wedge A_2) = T$	if $v_I(A_1) = T$ and $v_I(A_2) = T$
$v_I(A_1 \wedge A_2) = F$	in the other three cases
$v_I(A_1 \vee A_2) = F$	if $v_I(A_1) = F$ and $v_I(A_2) = F$
$v_I(A_1 \vee A_2) = T$	in the other three cases
$v_I(A_1 \rightarrow A_2) = F$	if $v_I(A_1) = T$ and $v_I(A_2) = F$
$v_I(A_1 \rightarrow A_2) = T$	in the other three cases

Semantics of propositional logic

Example for an evaluation

We will use the short notation v instead of v_I or v_{I_A} whenever the formula A and the interpretation I is clear from the context.

Semantics of propositional logic

Example for an evaluation

We will use the short notation v instead of v_I or v_{I_A} whenever the formula A and the interpretation I is clear from the context.

Example:

Semantics of propositional logic

Example for an evaluation

We will use the short notation v instead of v_I or v_{I_A} whenever the formula A and the interpretation I is clear from the context.

Example:

$A = p \rightarrow q \vee \neg(r \vee p)$.

$P_A = \{p, q, r\}$.

Semantics of propositional logic

Example for an evaluation

We will use the short notation v instead of v_I or v_{I_A} whenever the formula A and the interpretation I is clear from the context.

Example:

$A = p \to q \vee \neg(r \vee p)$.

$P_A = \{p, q, r\}$.

$I(p) = T,\ I(q) = F,\ I(r) = F$.

Semantics of propositional logic

Example for an evaluation

We will use the short notation v instead of v_I or v_{I_A} whenever the formula A and the interpretation I is clear from the context.

Example:

$A = p \to q \vee \neg(r \vee p)$.

$P_A = \{p, q, r\}$.

$I(p) = T,\ I(q) = F,\ I(r) = F$.

$v(r \vee p) = v(r) \vee v(p) = F \vee T = T$.

Semantics of propositional logic

Example for an evaluation

We will use the short notation v instead of v_I or v_{I_A} whenever the formula A and the interpretation I is clear from the context.

Example:

$A = p \to q \vee \neg(r \vee p)$.

$P_A = \{p, q, r\}$.

$I(p) = T,\ I(q) = F,\ I(r) = F$.

$v(r \vee p) = v(r) \vee v(p) = F \vee T = T$.

$v(\neg(r \vee p)) = \neg v(r \vee p) = \neg T = F$.

Semantics of propositional logic

Example for an evaluation

We will use the short notation v instead of v_I or v_{I_A} whenever the formula A and the interpretation I is clear from the context.

Example:

$A = p \to q \vee \neg(r \vee p)$.

$P_A = \{p, q, r\}$.

$I(p) = T,\ I(q) = F,\ I(r) = F$.

$v(r \vee p) = v(r) \vee v(p) = F \vee T = T$.

$v(\neg(r \vee p)) = \neg v(r \vee p) = \neg T = F$.

$v(q) = F$, $v(\neg(r \vee p)) = F$, so $v(q \vee \neg(r \vee p)) = F$

Semantics of propositional logic

Example for an evaluation

We will use the short notation v instead of v_I or v_{I_A} whenever the formula A and the interpretation I is clear from the context.

Example:

$A = p \to q \vee \neg(r \vee p)$.

$P_A = \{p, q, r\}$.

$I(p) = T,\ I(q) = F,\ I(r) = F$.

$v(r \vee p) = v(r) \vee v(p) = F \vee T = T$.

$v(\neg(r \vee p)) = \neg v(r \vee p) = \neg T = F$.

$v(q) = F$, $v(\neg(r \vee p)) = F$, so $v(q \vee \neg(r \vee p)) = F$

$v(p) = T$, $v(q \vee \neg(r \vee p)) = F$, so $v(A) = F$.

Semantics of propositional logic

Truth table

We may be interested in the truth value for every possible interpretation of a formula.

Semantics of propositional logic

Truth table

We may be interested in the truth value for every possible interpretation of a formula.

Let A be a formula and suppose, that $|P_A| = n$.

Semantics of propositional logic

Truth table

We may be interested in the truth value for every possible interpretation of a formula.

Let A be a formula and suppose, that $|P_A| = n$.

Since each of the n atoms can be assigned T or F independently, there are 2^n possible interpretations.

Semantics of propositional logic

Truth table

We may be interested in the truth value for every possible interpretation of a formula.

Let A be a formula and suppose, that $|P_A| = n$.

Since each of the n atoms can be assigned T or F independently, there are 2^n possible interpretations.

Definition

A **truth table** for a formula A is a table with $n+1$ columns and 2^n rows, where $n = |P_A|$. There is a column for each atom in P_A , plus a column for the formula A. The first n columns specify the interpretation I that maps atoms in P_A to $\{T, F\}$. The last column shows $v_I(A)$, the truth value of A for the interpretation I.

Semantics of propositional logic

Example for truth table

Example:

Let us make the truth table for $A = p \to q \vee \neg(r \vee p)$.

Semantics of propositional logic

Example for truth table

Example:

Let us make the truth table for $A = p \to q \vee \neg(r \vee p)$.

p	q	r	$r \vee p$	$\neg(r \vee p)$	$q \vee \neg(r \vee p)$	$p \to q \vee \neg(r \vee p)$
T	T	T	T	F	T	T
T	T	F	T	F	T	T
T	F	T	T	F	F	F
T	F	F	T	F	F	F
F	T	T	T	F	T	T
F	T	F	F	T	T	T
F	F	T	T	F	F	T
F	F	F	F	T	T	T

Semantics of propositional logic

Semantic properties of formulas

Definition

Let A be a formula

- A is satisfiable iff $v_I(A) = T$ for some interpretation I .

Semantics of propositional logic

Semantic properties of formulas

Definition

Let A be a formula

- A is **satisfiable** iff $v_I(A) = T$ for some interpretation I .
- A satisfying interpretation I is a **model** for A, denoted $I \models A$.

Semantics of propositional logic

Semantic properties of formulas

Definition

Let A be a formula

- A is **satisfiable** iff $v_I(A) = T$ for some interpretation I .
- A satisfying interpretation I is a **model** for A, denoted $I \models A$.
- A is **valid**, denoted $\models A$, iff $v_I(A) = T$ for all interpretations I . A valid propositional formula is also called a **tautology**.

Semantics of propositional logic

Semantic properties of formulas

Definition

Let A be a formula

- A is **satisfiable** iff $v_I(A) = T$ for some interpretation I .
- A satisfying interpretation I is a **model** for A, denoted $I \models A$.
- A is **valid**, denoted $\models A$, iff $v_I(A) = T$ for all interpretations I . A valid propositional formula is also called a **tautology**.
- A is **unsatisfiable** iff it is not satisfiable, that is, if $v_I(A) = F$ for all interpretations I.

Definition

Let A be a formula

- A is **satisfiable** iff $v_I(A) = T$ for some interpretation I .
- A satisfying interpretation I is a **model** for A, denoted $I \models A$.
- A is **valid**, denoted $\models A$, iff $v_I(A) = T$ for all interpretations I . A valid propositional formula is also called a **tautology**.
- A is **unsatisfiable** iff it is not satisfiable, that is, if $v_I(A) = F$ for all interpretations I.
- A is **falsifiable**, denoted $\not\models A$, iff it is not valid, that is, if $v_I(A) = F$ for some interpretation I.

Semantics of propositional logic

Semantic properties of formulas, examples

Let A_1, A_2 be formulas.

Definition

If $v_I(A_1) = v_I(A_2)$ for all interpretations I , then A_1 is **logically equivalent** to A_2 , denoted $A_1 \equiv A_2$.

Semantics of propositional logic

Semantic properties of formulas, examples

Let A_1, A_2 be formulas.

Definition

If $v_I(A_1) = v_I(A_2)$ for all interpretations I , then A_1 is **logically equivalent** to A_2 , denoted $A_1 \equiv A_2$.

Examples:

- Let $A = p \rightarrow q \vee \neg(r \vee p)$.

Let A_1, A_2 be formulas.

Definition

If $v_I(A_1) = v_I(A_2)$ for all interpretations I , then A_1 is **logically equivalent** to A_2 , denoted $A_1 \equiv A_2$.

Examples:

- Let $A = p \to q \lor \neg(r \lor p)$.

  Then A is satisfiable, falsifiable, interpretation TTT is a model for A. (See its truth table.)

Semantics of propositional logic

Semantic properties of formulas, examples

Let A_1, A_2 be formulas.

Definition

If $v_I(A_1) = v_I(A_2)$ for all interpretations I, then A_1 is **logically equivalent** to A_2, denoted $A_1 \equiv A_2$.

Examples:

- Let $A = p \rightarrow q \vee \neg(r \vee p)$.

  Then A is satisfiable, falsifiable, interpretation TTT is a model for A. (See its truth table.)

  On the other hand A is not valid and not unsatisfiable.

Semantics of propositional logic

Semantic properties of formulas, examples

Let A_1, A_2 be formulas.

Definition

If $v_I(A_1) = v_I(A_2)$ for all interpretations I , then A_1 is **logically equivalent** to A_2 , denoted $A_1 \equiv A_2$.

Examples:

- Let $A = p \rightarrow q \vee \neg(r \vee p)$.

  Then A is satisfiable, falsifiable, interpretation TTT is a model for A. (See its truth table.)

  On the other hand A is not valid and not unsatisfiable.

- formula $p \vee \neg p$ is valid, on the other hand $p \wedge \neg p$ is unsatisfiable.

Semantics of propositional logic

Semantic properties of formulas, examples

Let A_1, A_2 be formulas.

Definition

If $v_I(A_1) = v_I(A_2)$ for all interpretations I, then A_1 is **logically equivalent** to A_2, denoted $A_1 \equiv A_2$.

Examples:

- Let $A = p \rightarrow q \vee \neg(r \vee p)$.

  Then A is satisfiable, falsifiable, interpretation TTT is a model for A. (See its truth table.)

  On the other hand A is not valid and not unsatisfiable.

- formula $p \vee \neg p$ is valid, on the other hand $p \wedge \neg p$ is unsatisfiable.

- $p \vee q$ and $q \vee p$ are logically equivalent formulas.

Laws of propositional logic

Let us extend the syntax of Boolean formulas to include the two constant atomic propositions $\top$ (true) and $\bot$ (false).

Laws of propositional logic

Let us extend the syntax of Boolean formulas to include the two constant atomic propositions $\top$ (true) and $\bot$ (false). Their semantics are defined as $I(\top) = T$ and $I(\bot) = F$ for any interpretation I.

Laws of propositional logic

Let us extend the syntax of Boolean formulas to include the two constant atomic propositions $\top$ (true) and $\bot$ (false). Their semantics are defined as $I(\top) = T$ and $I(\bot) = F$ for any interpretation I.

- $A \vee \top \equiv \top$ and $A \wedge \bot \equiv \bot$ (domination laws),
- $A \vee \bot \equiv A$ and $A \wedge \top \equiv A$ (identity laws),

Laws of propositional logic

Let us extend the syntax of Boolean formulas to include the two constant atomic propositions $\top$ (true) and $\bot$ (false). Their semantics are defined as $I(\top) = T$ and $I(\bot) = F$ for any interpretation I.

- $A \vee \top \equiv \top$ and $A \wedge \bot \equiv \bot$ (domination laws),
- $A \vee \bot \equiv A$ and $A \wedge \top \equiv A$ (identity laws),
- $A \vee \neg A \equiv \top$ and $A \wedge \neg A \equiv \bot$,
- $\neg\neg A \equiv A$ (double negation law),
- $A \vee A \equiv A$ and $A \wedge A \equiv A$ (idempotent laws),
- $A \rightarrow B \equiv \neg A \vee B$,
- $A \rightarrow B \equiv \neg B \rightarrow \neg A$ (law of transposition),

Laws of propositional logic
(cont'd)

- $A \vee B \equiv B \vee A$ and $A \wedge B \equiv B \wedge A$ (commutative laws),

Laws of propositional logic
(cont'd)

- $A \vee B \equiv B \vee A$ and $A \wedge B \equiv B \wedge A$ (commutative laws),
- $(A \vee B) \vee C \equiv A \vee (B \vee C)$ and
  $(A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$ (associative laws),

Laws of propositional logic
(cont'd)

- $A \vee B \equiv B \vee A$ and $A \wedge B \equiv B \wedge A$ (commutative laws),
- $(A \vee B) \vee C \equiv A \vee (B \vee C)$ and
  $(A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$ (associative laws),
- $(A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C)$ and
  $(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C)$ (distributive laws),

Laws of propositional logic
(cont'd)

- $A \vee B \equiv B \vee A$ and $A \wedge B \equiv B \wedge A$ (commutative laws),
- $(A \vee B) \vee C \equiv A \vee (B \vee C)$ and
  $(A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$ (associative laws),
- $(A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C)$ and
  $(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C)$ (distributive laws),
- $\neg(A \wedge B) \equiv \neg A \vee \neg B$ and $\neg(A \vee B) \equiv \neg A \wedge \neg B$
  (De Morgan laws),

Laws of propositional logic

(cont'd)

- $A \vee B \equiv B \vee A$ and $A \wedge B \equiv B \wedge A$ (commutative laws),
- $(A \vee B) \vee C \equiv A \vee (B \vee C)$ and
  $(A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$ (associative laws),
- $(A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C)$ and
  $(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C)$ (distributive laws),
- $\neg(A \wedge B) \equiv \neg A \vee \neg B$ and $\neg(A \vee B) \equiv \neg A \wedge \neg B$
  (De Morgan laws),
- $(A \vee B) \wedge B \equiv B$ and $(A \wedge B) \vee B \equiv B$ (absorption laws).

Laws of propositional logic

(cont'd)

- $A \vee B \equiv B \vee A$ and $A \wedge B \equiv B \wedge A$ (commutative laws),
- $(A \vee B) \vee C \equiv A \vee (B \vee C)$ and
  $(A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$ (associative laws),
- $(A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C)$ and
  $(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C)$ (distributive laws),
- $\neg(A \wedge B) \equiv \neg A \vee \neg B$ and $\neg(A \vee B) \equiv \neg A \wedge \neg B$
  (De Morgan laws),
- $(A \vee B) \wedge B \equiv B$ and $(A \wedge B) \vee B \equiv B$ (absorption laws).

Definitions of other Boolean ops:

- $A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$,
- $A \oplus B \equiv (A \vee B) \wedge \neg(A \wedge B)$,

Laws of propositional logic
(cont'd)

- $A \vee B \equiv B \vee A$ and $A \wedge B \equiv B \wedge A$ (commutative laws),
- $(A \vee B) \vee C \equiv A \vee (B \vee C)$ and
  $(A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$ (associative laws),
- $(A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C)$ and
  $(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C)$ (distributive laws),
- $\neg(A \wedge B) \equiv \neg A \vee \neg B$ and $\neg(A \vee B) \equiv \neg A \wedge \neg B$
  (De Morgan laws),
- $(A \vee B) \wedge B \equiv B$ and $(A \wedge B) \vee B \equiv B$ (absorption laws).

Definitions of other Boolean ops:

- $A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$,
- $A \oplus B \equiv (A \vee B) \wedge \neg(A \wedge B)$,
- $A \uparrow B \equiv \neg(A \wedge B)$,
- $A \downarrow B \equiv \neg(A \vee B)$.

Substitution

Definition

Let A be a subformula of B and let A' be any formula. $B\{A \leftarrow A'\}$, the **substitution** of A for A' in B, is the formula obtained by replacing all occurrences of the subtree for A in B by A'.

Substitution

Definition

Let A be a subformula of B and let A' be any formula. $B\{A \leftarrow A'\}$, the **substitution** of A for A' in B, is the formula obtained by replacing all occurrences of the subtree for A in B by A'.

Example:

Substitution

Definition

Let A be a subformula of B and let A' be any formula. $B\{A \leftarrow A'\}$, the **substitution** of A for A' in B, is the formula obtained by replacing all occurrences of the subtree for A in B by A'.

Example:

$B = (p \to q) \vee \neg(p \to q) \vee (\neg q \to \neg p)$.

Substitution

Definition

Let A be a subformula of B and let A' be any formula. $B\{A \leftarrow A'\}$, the **substitution** of A for A' in B, is the formula obtained by replacing all occurrences of the subtree for A in B by A'.

Example:

$B = (p \rightarrow q) \vee \neg(p \rightarrow q) \vee (\neg q \rightarrow \neg p)$.

$A = p \rightarrow q, \; A' = \neg p \vee q$

Substitution

Definition

Let A be a subformula of B and let A' be any formula. $B\{A \leftarrow A'\}$, the **substitution** of A for A' in B, is the formula obtained by replacing all occurrences of the subtree for A in B by A'.

Example:

$B = (p \rightarrow q) \vee \neg(p \rightarrow q) \vee (\neg q \rightarrow \neg p)$.

$A = p \rightarrow q, \;\; A' = \neg p \vee q$

$B\{A \leftarrow A'\} = (\neg p \vee q) \vee \neg(\neg p \vee q) \vee (\neg q \rightarrow \neg p)$.

Substitution

Definition

Let A be a subformula of B and let A' be any formula. $B\{A \leftarrow A'\}$, the **substitution** of A for A' in B, is the formula obtained by replacing all occurrences of the subtree for A in B by A'.

Example:

$B = (p \rightarrow q) \vee \neg(p \rightarrow q) \vee (\neg q \rightarrow \neg p)$.

$A = p \rightarrow q, \; A' = \neg p \vee q$

$B\{A \leftarrow A'\} = (\neg p \vee q) \vee \neg(\neg p \vee q) \vee (\neg q \rightarrow \neg p)$.

$B\{p \leftarrow \neg r\} = (\neg r \rightarrow q) \vee \neg(\neg r \rightarrow q) \vee (\neg q \rightarrow \neg\neg r)$,

Substitution

Definition

Let A be a subformula of B and let A' be any formula. $B\{A \leftarrow A'\}$, the **substitution** of A for A' in B, is the formula obtained by replacing all occurrences of the subtree for A in B by A'.

Example:

$B = (p \to q) \vee \neg(p \to q) \vee (\neg q \to \neg p)$.
$A = p \to q,\ A' = \neg p \vee q$
$B\{A \leftarrow A'\} = (\neg p \vee q) \vee \neg(\neg p \vee q) \vee (\neg q \to \neg p)$.
$B\{p \leftarrow \neg r\} = (\neg r \to q) \vee \neg(\neg r \to q) \vee (\neg q \to \neg\neg r)$,

Theorem

Let A be a subformula of B and let A' be a formula such that $A \equiv A'$. Then $B \equiv B\{A \leftarrow A'\}$.

Substitution

Proof:
Let $B' = B\{A \leftarrow A'\}$ and I be an arbitrary interpretation. Since $A \equiv A'$ we have $v(A) = v(A')$. We have to show that $v(B) = v(B')$.

Substitution

Proof:
Let $B' = B\{A \leftarrow A'\}$ and I be an arbitrary interpretation. Since $A \equiv A'$ we have $v(A) = v(A')$. We have to show that $v(B) = v(B')$.

The proof is by induction on the depth d of the highest occurrence of the subtree A in B.

Substitution

Proof:
Let $B' = B\{A \leftarrow A'\}$ and I be an arbitrary interpretation. Since $A \equiv A'$ we have $v(A) = v(A')$. We have to show that $v(B) = v(B')$.

The proof is by induction on the depth d of the highest occurrence of the subtree A in B.

If $d = 0$, there is only one occurrence of A, namely B itself. Obviously, $v(B) = v(A) = v(A') = v(B')$.

Substitution

Proof:

Let $B' = B\{A \leftarrow A'\}$ and I be an arbitrary interpretation. Since $A \equiv A'$ we have $v(A) = v(A')$. We have to show that $v(B) = v(B')$.

The proof is by induction on the depth d of the highest occurrence of the subtree A in B.

If $d = 0$, there is only one occurrence of A, namely B itself. Obviously, $v(B) = v(A) = v(A') = v(B')$.

If $d \neq 0$, then B is either $\neg B_1$ or $B_1\, op\, B_2$ for some formulas B_1, B_2 and binary operator op.

Substitution

Proof:
Let $B' = B\{A \leftarrow A'\}$ and I be an arbitrary interpretation. Since $A \equiv A'$ we have $v(A) = v(A')$. We have to show that $v(B) = v(B')$.

The proof is by induction on the depth d of the highest occurrence of the subtree A in B.

If $d = 0$, there is only one occurrence of A, namely B itself. Obviously, $v(B) = v(A) = v(A') = v(B')$.

If $d \neq 0$, then B is either $\neg B_1$ or $B_1 \, op \, B_2$ for some formulas B_1, B_2 and binary operator op.

In B_1 , the depth of A is less than d. By the inductive hypothesis, $v(B_1) = v(B_1') = v(B_1\{A \leftarrow A'\})$, and similarly $v(B_2) = v(B_2') = v(B_2\{A \leftarrow A'\})$.

Substitution

Proof:
Let $B' = B\{A \leftarrow A'\}$ and I be an arbitrary interpretation. Since $A \equiv A'$ we have $v(A) = v(A')$. We have to show that $v(B) = v(B')$.

The proof is by induction on the depth d of the highest occurrence of the subtree A in B.

If $d = 0$, there is only one occurrence of A, namely B itself. Obviously, $v(B) = v(A) = v(A') = v(B')$.

If $d \neq 0$, then B is either $\neg B_1$ or $B_1 \, op \, B_2$ for some formulas B_1, B_2 and binary operator op.

In B_1 , the depth of A is less than d. By the inductive hypothesis, $v(B_1) = v(B_1') = v(B_1\{A \leftarrow A'\})$, and similarly $v(B_2) = v(B_2') = v(B_2\{A \leftarrow A'\})$.

By the definition of v, $v(B)$ depends only on $v(B_1)$ and $v(B_2)$, so $v(B) = v(B')$ proving the theorem.

Substitution

We can prove validity/unsatisfiability of a formula or logical equivalence of two formulas by applying laws of propositional logic and the substitution theorem.

Substitution

We can prove validity/unsatisfiability of a formula or logical equivalence of two formulas by applying laws of propositional logic and the substitution theorem.

Example: $A \rightarrow (B \rightarrow A)$

Substitution

We can prove validity/unsatisfiability of a formula or logical equivalence of two formulas by applying laws of propositional logic and the substitution theorem.

Example: $A \rightarrow (B \rightarrow A) \ \equiv \neg A \vee (\neg B \vee A)$

Substitution

We can prove validity/unsatisfiability of a formula or logical equivalence of two formulas by applying laws of propositional logic and the substitution theorem.

Example: $A \to (B \to A) \equiv \neg A \vee (\neg B \vee A) \equiv (\neg A \vee \neg B) \vee A$

Substitution

We can prove validity/unsatisfiability of a formula or logical equivalence of two formulas by applying laws of propositional logic and the substitution theorem.

Example: $A \to (B \to A) \ \equiv \neg A \lor (\neg B \lor A) \ \equiv (\neg A \lor \neg B) \lor A$
$\equiv (\neg B \lor \neg A) \lor A$

Substitution

We can prove validity/unsatisfiability of a formula or logical equivalence of two formulas by applying laws of propositional logic and the substitution theorem.

Example: $A \to (B \to A) \ \equiv \neg A \lor (\neg B \lor A) \ \equiv (\neg A \lor \neg B) \lor A$
$\equiv (\neg B \lor \neg A) \lor A \ \equiv \neg B \lor (\neg A \lor A)$

Substitution

We can prove validity/unsatisfiability of a formula or logical equivalence of two formulas by applying laws of propositional logic and the substitution theorem.

Example: $A \to (B \to A) \;\equiv \neg A \vee (\neg B \vee A) \;\equiv (\neg A \vee \neg B) \vee A$
$\equiv (\neg B \vee \neg A) \vee A \;\equiv \neg B \vee (\neg A \vee A) \;\equiv \neg B \vee (A \vee \neg A)$

Substitution

We can prove validity/unsatisfiability of a formula or logical equivalence of two formulas by applying laws of propositional logic and the substitution theorem.

Example: $A \to (B \to A) \equiv \neg A \vee (\neg B \vee A) \equiv (\neg A \vee \neg B) \vee A$
$\equiv (\neg B \vee \neg A) \vee A \equiv \neg B \vee (\neg A \vee A) \equiv \neg B \vee (A \vee \neg A)$
$\equiv \neg B \vee \top$

Substitution

We can prove validity/unsatisfiability of a formula or logical equivalence of two formulas by applying laws of propositional logic and the substitution theorem.

Example: $A \to (B \to A) \ \equiv \neg A \lor (\neg B \lor A) \ \equiv (\neg A \lor \neg B) \lor A$
$\equiv (\neg B \lor \neg A) \lor A \ \equiv \neg B \lor (\neg A \lor A) \ \equiv \neg B \lor (A \lor \neg A)$
$\equiv \neg B \lor \top \ \equiv \top.$

Substitution

We can prove validity/unsatisfiability of a formula or logical equivalence of two formulas by applying laws of propositional logic and the substitution theorem.

Example: $A \to (B \to A) \;\equiv \neg A \vee (\neg B \vee A) \;\equiv (\neg A \vee \neg B) \vee A$
$\equiv (\neg B \vee \neg A) \vee A \;\equiv \neg B \vee (\neg A \vee A) \;\equiv \neg B \vee (A \vee \neg A)$
$\equiv \neg B \vee \top \;\equiv \top$.

So $A \to (B \to A)$ is a valid formula.

Semantic properties of a set of formulas

Satisfiability

Definition

A set of formulas $U = \{A_1, \ldots\}$ is (simultaneously) **satisfiable** iff there exists an interpretation I such that $v_I(A_i) = T$ for all i. The satisfying interpretation is a **model** of U, denoted $I \models U$. U is **unsatisfiable** iff for every interpretation I, there exists an i such that $v_I(A_i) = F$.

Semantic properties of a set of formulas

Satisfiability

Definition

A set of formulas $U = \{A_1, \ldots\}$ is (simultaneously) **satisfiable** iff there exists an interpretation I such that $v_I(A_i) = T$ for all i. The satisfying interpretation is a **model** of U, denoted $I \models U$. U is **unsatisfiable** iff for every interpretation I, there exists an i such that $v_I(A_i) = F$.

Example:

Semantic properties of a set of formulas

Satisfiability

Definition

A set of formulas $U = \{A_1, \ldots\}$ is (simultaneously) **satisfiable** iff there exists an interpretation I such that $v_I(A_i) = T$ for all i. The satisfying interpretation is a **model** of U, denoted $I \models U$. U is **unsatisfiable** iff for every interpretation I, there exists an i such that $v_I(A_i) = F$.

Example:

$U_1 = \{p, \neg p \vee q, q \wedge r\}$,

Semantic properties of a set of formulas

Satisfiability

Definition

A set of formulas $U = \{A_1, \ldots\}$ is (simultaneously) **satisfiable** iff there exists an interpretation I such that $v_I(A_i) = T$ for all i. The satisfying interpretation is a **model** of U, denoted $I \models U$. U is **unsatisfiable** iff for every interpretation I, there exists an i such that $v_I(A_i) = F$.

Example:

$U_1 = \{p, \neg p \vee q, q \wedge r\}$,
$U_2 = \{p, \neg p \vee q, \neg p\}$.

Semantic properties of a set of formulas

Definition

A set of formulas $U = \{A_1, \ldots\}$ is (simultaneously) **satisfiable** iff there exists an interpretation I such that $v_I(A_i) = T$ for all i. The satisfying interpretation is a **model** of U, denoted $I \models U$. U is **unsatisfiable** iff for every interpretation I, there exists an i such that $v_I(A_i) = F$.

Example:

$U_1 = \{p, \neg p \vee q, q \wedge r\}$,
$U_2 = \{p, \neg p \vee q, \neg p\}$.
$U_3 = \{p, \neg p \vee q, \neg q\}$.

Semantic properties of a set of formulas

Satisfiability

Definition

A set of formulas $U = \{A_1, \ldots\}$ is (simultaneously) **satisfiable** iff there exists an interpretation I such that $v_I(A_i) = T$ for all i. The satisfying interpretation is a **model** of U, denoted $I \models U$. U is **unsatisfiable** iff for every interpretation I, there exists an i such that $v_I(A_i) = F$.

Example:

$U_1 = \{p, \neg p \vee q, q \wedge r\}$,
$U_2 = \{p, \neg p \vee q, \neg p\}$.
$U_3 = \{p, \neg p \vee q, \neg q\}$.
Which of the three is satisfiable?

Semantic properties of a set of formulas

Satisfiability

Definition

A set of formulas $U = \{A_1, \ldots\}$ is (simultaneously) **satisfiable** iff there exists an interpretation I such that $v_I(A_i) = T$ for all i. The satisfying interpretation is a **model** of U, denoted $I \models U$. U is **unsatisfiable** iff for every interpretation I, there exists an i such that $v_I(A_i) = F$.

Example:

$U_1 = \{p, \neg p \vee q, q \wedge r\}$,
$U_2 = \{p, \neg p \vee q, \neg p\}$.
$U_3 = \{p, \neg p \vee q, \neg q\}$.
Which of the three is satisfiable?
$I \models U_1$, where $I(p) = I(q) = I(r) = T$.

Semantic properties of a set of formulas

Satisfiability

Definition

A set of formulas $U = \{A_1, \ldots\}$ is (simultaneously) **satisfiable** iff there exists an interpretation I such that $v_I(A_i) = T$ for all i. The satisfying interpretation is a **model** of U, denoted $I \models U$. U is **unsatisfiable** iff for every interpretation I, there exists an i such that $v_I(A_i) = F$.

Example:

$U_1 = \{p, \neg p \vee q, q \wedge r\}$,
$U_2 = \{p, \neg p \vee q, \neg p\}$.
$U_3 = \{p, \neg p \vee q, \neg q\}$.
Which of the three is satisfiable?
$I \models U_1$, where $I(p) = I(q) = I(r) = T$.
p and $\neg p$ can not be T simultaneously, so for all interpretations I:
$I \not\models U_2$.

Semantic properties of a set of formulas

Satisfiability

Definition

A set of formulas $U = \{A_1, \ldots\}$ is (simultaneously) **satisfiable** iff there exists an interpretation I such that $v_I(A_i) = T$ for all i. The satisfying interpretation is a **model** of U, denoted $I \models U$. U is **unsatisfiable** iff for every interpretation I, there exists an i such that $v_I(A_i) = F$.

Example:

$U_1 = \{p, \neg p \vee q, q \wedge r\}$,
$U_2 = \{p, \neg p \vee q, \neg p\}$.
$U_3 = \{p, \neg p \vee q, \neg q\}$.
Which of the three is satisfiable?
$I \models U_1$, where $I(p) = I(q) = I(r) = T$.
p and $\neg p$ can not be T simultaneously, so for all interpretations I: $I \not\models U_2$.
One can check that for all interpretations I: $I \not\models U_3$.

Semantic properties of a set of formulas

Logical consequence

Proposition

If U is satisfiable, then U' is satisfiable, too, for all $U' \subseteq U$.

Semantic properties of a set of formulas

Logical consequence

Proposition

If U is satisfiable, then U' is satisfiable, too, for all $U' \subseteq U$.
If U is unsatisfiable, then U'' is unsatisfiable for all $U \subseteq U''$.

Semantic properties of a set of formulas

Logical consequence

Proposition

If U is satisfiable, then U' is satisfiable, too, for all $U' \subseteq U$.
If U is unsatisfiable, then U'' is unsatisfiable for all $U \subseteq U''$.

Definition

Let U be a set of formulas and A a formula. A is a **logical consequence** of U, denoted $U \models A$, iff every model of U is a model for A.

Semantic properties of a set of formulas

Logical consequence

Proposition

If U is satisfiable, then U' is satisfiable, too, for all $U' \subseteq U$.
If U is unsatisfiable, then U'' is unsatisfiable for all $U \subseteq U''$.

Definition

Let U be a set of formulas and A a formula. A is a **logical consequence** of U, denoted $U \models A$, iff every model of U is a model for A.

Example:

Semantic properties of a set of formulas

Logical consequence

Proposition

If U is satisfiable, then U' is satisfiable, too, for all $U' \subseteq U$.
If U is unsatisfiable, then U'' is unsatisfiable for all $U \subseteq U''$.

Definition

Let U be a set of formulas and A a formula. A is a **logical consequence** of U, denoted $U \models A$, iff every model of U is a model for A.

Example:

Let $A = (p \vee r) \wedge (\neg q \vee \neg r)$.

Semantic properties of a set of formulas

Logical consequence

Proposition

If U is satisfiable, then U' is satisfiable, too, for all $U' \subseteq U$.
If U is unsatisfiable, then U'' is unsatisfiable for all $U \subseteq U''$.

Definition

Let U be a set of formulas and A a formula. A is a **logical consequence** of U, denoted $U \models A$, iff every model of U is a model for A.

Example:
Let $A = (p \vee r) \wedge (\neg q \vee \neg r)$. Then A is a logical consequence of $\{p, \neg q\}$, denoted $\{p, \neg q\} \models A$,

Semantic properties of a set of formulas

Logical consequence

Proposition

If U is satisfiable, then U' is satisfiable, too, for all $U' \subseteq U$.
If U is unsatisfiable, then U'' is unsatisfiable for all $U \subseteq U''$.

Definition

Let U be a set of formulas and A a formula. A is a **logical consequence** of U, denoted $U \models A$, iff every model of U is a model for A.

Example:
Let $A = (p \vee r) \wedge (\neg q \vee \neg r)$. Then A is a logical consequence of $\{p, \neg q\}$, denoted $\{p, \neg q\} \models A$, since A is true in all interpretations I such that $I(p) = T$ and $I(q) = F$.

Semantic properties of a set of formulas

Logical consequence

Proposition

If U is satisfiable, then U' is satisfiable, too, for all $U' \subseteq U$.
If U is unsatisfiable, then U'' is unsatisfiable for all $U \subseteq U''$.

Definition

Let U be a set of formulas and A a formula. A is a **logical consequence** of U, denoted $U \models A$, iff every model of U is a model for A.

Example:
Let $A = (p \vee r) \wedge (\neg q \vee \neg r)$. Then A is a logical consequence of $\{p, \neg q\}$, denoted $\{p, \neg q\} \models A$, since A is true in all interpretations I such that $I(p) = T$ and $I(q) = F$.
Note, that A is not valid, since it is not true in the interpretation I where $I(p) = F, I(q) = T, I(r) = T$.

Conjunctive normal form (CNF)

Definition

A **literal** is either an atom or a negation of an atom.

Conjunctive normal form (CNF)

Definition

A **literal** is either an atom or a negation of an atom.

Definition

A formula is in **conjunctive normal form (CNF)** iff it is a conjunction of disjunctions of literals.

Conjunctive normal form (CNF)

Definition

A **literal** is either an atom or a negation of an atom.

Definition

A formula is in **conjunctive normal form (CNF)** iff it is a conjunction of disjunctions of literals.

Example:

Conjunctive normal form (CNF)

Definition

A **literal** is either an atom or a negation of an atom.

Definition

A formula is in **conjunctive normal form (CNF)** iff it is a conjunction of disjunctions of literals.

Example:

$$(\neg p \vee q \vee r) \wedge (\neg q \vee r) \wedge (\neg r)$$

Conjunctive normal form (CNF)

Definition

A **literal** is either an atom or a negation of an atom.

Definition

A formula is in **conjunctive normal form (CNF)** iff it is a conjunction of disjunctions of literals.

Example:

$(\neg p \vee q \vee r) \wedge (\neg q \vee r) \wedge (\neg r)$ CNF

Conjunctive normal form (CNF)

Definition

A **literal** is either an atom or a negation of an atom.

Definition

A formula is in **conjunctive normal form (CNF)** iff it is a conjunction of disjunctions of literals.

Example:

$(\neg p \vee q \vee r) \wedge (\neg q \vee r) \wedge (\neg r)$ CNF

$(\neg p \vee q \vee r) \wedge ((p \wedge \neg q) \vee r) \wedge (\neg r)$

Conjunctive normal form (CNF)

Definition

A **literal** is either an atom or a negation of an atom.

Definition

A formula is in **conjunctive normal form (CNF)** iff it is a conjunction of disjunctions of literals.

Example:

$(\neg p \lor q \lor r) \land (\neg q \lor r) \land (\neg r)$ CNF

$(\neg p \lor q \lor r) \land ((p \land \neg q) \lor r) \land (\neg r)$ not in CNF

Conjunctive normal form (CNF)

Definition

A **literal** is either an atom or a negation of an atom.

Definition

A formula is in **conjunctive normal form (CNF)** iff it is a conjunction of disjunctions of literals.

Example:

$(\neg p \lor q \lor r) \land (\neg q \lor r) \land (\neg r)$ CNF

$(\neg p \lor q \lor r) \land ((p \land \neg q) \lor r) \land (\neg r)$ not in CNF

$(\neg p \lor q \lor r) \land \neg(\neg q \lor r) \land (\neg r)$

Conjunctive normal form (CNF)

Definition

A **literal** is either an atom or a negation of an atom.

Definition

A formula is in **conjunctive normal form (CNF)** iff it is a conjunction of disjunctions of literals.

Example:

$(\neg p \vee q \vee r) \wedge (\neg q \vee r) \wedge (\neg r)$ CNF

$(\neg p \vee q \vee r) \wedge ((p \wedge \neg q) \vee r) \wedge (\neg r)$ not in CNF

$(\neg p \vee q \vee r) \wedge \neg(\neg q \vee r) \wedge (\neg r)$ not in CNF

Conjunctive normal form (CNF)

Definition

A **literal** is either an atom or a negation of an atom.

Definition

A formula is in **conjunctive normal form (CNF)** iff it is a conjunction of disjunctions of literals.

Example:

$(\neg p \vee q \vee r) \wedge (\neg q \vee r) \wedge (\neg r)$ CNF

$(\neg p \vee q \vee r) \wedge ((p \wedge \neg q) \vee r) \wedge (\neg r)$ not in CNF

$(\neg p \vee q \vee r) \wedge \neg(\neg q \vee r) \wedge (\neg r)$ not in CNF

$\neg p \vee q \vee r$

Conjunctive normal form (CNF)

Definition

A **literal** is either an atom or a negation of an atom.

Definition

A formula is in **conjunctive normal form (CNF)** iff it is a conjunction of disjunctions of literals.

Example:

$(\neg p \lor q \lor r) \land (\neg q \lor r) \land (\neg r)$ CNF

$(\neg p \lor q \lor r) \land ((p \land \neg q) \lor r) \land (\neg r)$ not in CNF

$(\neg p \lor q \lor r) \land \neg(\neg q \lor r) \land (\neg r)$ not in CNF

$\neg p \lor q \lor r$ CNF

Conjunctive normal form (CNF)

Theorem

For every formula A in propositional logic there is a logically equivalent formula in CNF.

Conjunctive normal form (CNF)

Theorem

For every formula A in propositional logic there is a logically equivalent formula in CNF.

Proof Let $P_A = \{p_1, \ldots, p_n\}$ and let $A^F = \{I \mid v_I(A) = F\}$ be the set of those interpretation evaluating A for false.

Conjunctive normal form (CNF)

Theorem

For every formula A in propositional logic there is a logically equivalent formula in CNF.

Proof Let $P_A = \{p_1, \ldots, p_n\}$ and let $A^F = \{I \mid v_I(A) = F\}$ be the set of those interpretation evaluating A for false.

For every $I \in A^F$

$$B_I = \bigvee_{x:I(x)=T} \neg x \vee \bigvee_{x:I(x)=F} x$$

is a disjunction of literals with the property $B_I^F = \{I\}$.

Conjunctive normal form (CNF)

Theorem

For every formula A in propositional logic there is a logically equivalent formula in CNF.

Proof Let $P_A = \{p_1, \ldots, p_n\}$ and let $A^F = \{I \mid v_I(A) = F\}$ be the set of those interpretation evaluating A for false.

For every $I \in A^F$

$$B_I = \bigvee_{x:I(x)=T} \neg x \vee \bigvee_{x:I(x)=F} x$$

is a disjunction of literals with the property $B_I^F = \{I\}$.

$B = \bigwedge_{I \in A^F} B_I$ has the property

$$B^F = \bigcup_{I \in A^F} B_I^F = \bigcup_{I \in A^F} \{I\} = A^F,$$

i.e., $B \equiv A$ and B is in CNF proving the theorem.

Conjunctive normal form (CNF)

Example: Let $A = (p \to q) \to r$. Then the truth table for A is the following

p	q	r	A
T	T	T	T
T	T	F	F
T	F	T	T
T	F	F	T
F	T	T	T
F	T	F	F
F	F	T	T
F	F	F	F

Conjunctive normal form (CNF)

Example: Let $A = (p \to q) \to r$. Then the truth table for A is the following

p	q	r	A
T	T	T	T
T	T	F	F
T	F	T	T
T	F	F	T
F	T	T	T
F	T	F	F
F	F	T	T
F	F	F	F

According the previous proof

$$(\neg p \vee \neg q \vee r) \wedge (p \vee \neg q \vee r) \wedge (p \vee q \vee r)$$

is a CNF equivalent with A.

Conjunctive normal form (CNF)

Heuristic algorithm (can be made formal):
1st step: Eliminate implications by the law $A \to B \equiv \neg A \vee B$

Conjunctive normal form (CNF)

Heuristic algorithm (can be made formal):
1st step: Eliminate implications by the law $A \to B \equiv \neg A \vee B$

2nd step: Use De Morgan's laws and double negation law to tranform the formula into a formula with the property that negations occur only right before the atoms

Conjunctive normal form (CNF)

Heuristic algorithm (can be made formal):
1st step: Eliminate implications by the law $A \to B \equiv \neg A \vee B$

2nd step: Use De Morgan's laws and double negation law to tranform the formula into a formula with the property that negations occur only right before the atoms
$\neg(A \wedge B) \equiv \neg A \vee \neg B, \quad \neg(A \vee B) \equiv \neg A \wedge \neg B$ and $\neg\neg A \equiv A$,

Conjunctive normal form (CNF)

Heuristic algorithm (can be made formal):
1st step: Eliminate implications by the law $A \to B \equiv \neg A \vee B$

2nd step: Use De Morgan's laws and double negation law to tranform the formula into a formula with the property that negations occur only right before the atoms
$\neg(A \wedge B) \equiv \neg A \vee \neg B$, $\neg(A \vee B) \equiv \neg A \wedge \neg B$ and $\neg\neg A \equiv A$,

3rd step: Now, the formula is in the form of literals connected by $\wedge$'s and $\vee$'s. Use distributive laws,
$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$ and
$(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C)$
to eliminate conjunctions within disjunctions or vica versa.

Conjunctive normal form (CNF)

Heuristic algorithm (can be made formal):
1st step: Eliminate implications by the law $A \to B \equiv \neg A \vee B$

2nd step: Use De Morgan's laws and double negation law to tranform the formula into a formula with the property that negations occur only right before the atoms
$\neg(A \wedge B) \equiv \neg A \vee \neg B, \quad \neg(A \vee B) \equiv \neg A \wedge \neg B$ and $\neg\neg A \equiv A$,

3rd step: Now, the formula is in the form of literals connected by $\wedge$'s and $\vee$'s. Use distributive laws,
$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$ and
$(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C)$
to eliminate conjunctions within disjunctions or vica versa.

Example:

Conjunctive normal form (CNF)

Heuristic algorithm (can be made formal):
1st step: Eliminate implications by the law $A \to B \equiv \neg A \vee B$

2nd step: Use De Morgan's laws and double negation law to tranform the formula into a formula with the property that negations occur only right before the atoms
$\neg(A \wedge B) \equiv \neg A \vee \neg B, \quad \neg(A \vee B) \equiv \neg A \wedge \neg B$ and $\neg\neg A \equiv A$,

3rd step: Now, the formula is in the form of literals connected by $\wedge$'s and $\vee$'s. Use distributive laws,
$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$ and
$(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C)$
to eliminate conjunctions within disjunctions or vica versa.

Example:

$(\neg p \to \neg q) \to (p \to q)$

Conjunctive normal form (CNF)

Heuristic algorithm (can be made formal):
1st step: Eliminate implications by the law $A \to B \equiv \neg A \vee B$

2nd step: Use De Morgan's laws and double negation law to tranform the formula into a formula with the property that negations occur only right before the atoms
$\neg(A \wedge B) \equiv \neg A \vee \neg B, \quad \neg(A \vee B) \equiv \neg A \wedge \neg B$ and $\neg\neg A \equiv A$,

3rd step: Now, the formula is in the form of literals connected by $\wedge$'s and $\vee$'s. Use distributive laws,
$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$ and
$(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C)$
to eliminate conjunctions within disjunctions or vica versa.

Example:

$$(\neg p \to \neg q) \to (p \to q) \quad \equiv \neg(\neg\neg p \vee \neg q) \vee (\neg p \vee q)$$

Conjunctive normal form (CNF)

Heuristic algorithm (can be made formal):
1st step: Eliminate implications by the law $A \to B \equiv \neg A \vee B$

2nd step: Use De Morgan's laws and double negation law to tranform the formula into a formula with the property that negations occur only right before the atoms
$\neg(A \wedge B) \equiv \neg A \vee \neg B, \quad \neg(A \vee B) \equiv \neg A \wedge \neg B$ and $\neg\neg A \equiv A$,

3rd step: Now, the formula is in the form of literals connected by $\wedge$'s and $\vee$'s. Use distributive laws,
$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$ and
$(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C)$
to eliminate conjunctions within disjunctions or vica versa.

Example:

$$\begin{aligned}(\neg p \to \neg q) \to (p \to q) \quad &\equiv \neg(\neg\neg p \vee \neg q) \vee (\neg p \vee q)\\ &\equiv (\neg\neg\neg p \wedge \neg\neg q) \vee (\neg p \vee q)\end{aligned}$$

Conjunctive normal form (CNF)

Heuristic algorithm (can be made formal):
1st step: Eliminate implications by the law $A \to B \equiv \neg A \vee B$

2nd step: Use De Morgan's laws and double negation law to tranform the formula into a formula with the property that negations occur only right before the atoms
$\neg(A \wedge B) \equiv \neg A \vee \neg B, \quad \neg(A \vee B) \equiv \neg A \wedge \neg B$ and $\neg\neg A \equiv A$,

3rd step: Now, the formula is in the form of literals connected by $\wedge$'s and $\vee$'s. Use distributive laws,
$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$ and
$(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C)$
to eliminate conjunctions within disjunctions or vica versa.

Example:

$$\begin{aligned}(\neg p \to \neg q) \to (p \to q) \quad &\equiv \neg(\neg\neg p \vee \neg q) \vee (\neg p \vee q)\\ &\equiv (\neg\neg\neg p \wedge \neg\neg q) \vee (\neg p \vee q)\\ &\equiv (\neg p \wedge q) \vee (\neg p \vee q)\end{aligned}$$

Conjunctive normal form (CNF)

Heuristic algorithm (can be made formal):
1st step: Eliminate implications by the law $A \to B \equiv \neg A \vee B$

2nd step: Use De Morgan's laws and double negation law to tranform the formula into a formula with the property that negations occur only right before the atoms
$\neg(A \wedge B) \equiv \neg A \vee \neg B, \quad \neg(A \vee B) \equiv \neg A \wedge \neg B$ and $\neg\neg A \equiv A$,

3rd step: Now, the formula is in the form of literals connected by $\wedge$'s and $\vee$'s. Use distributive laws,
$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$ and
$(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C)$
to eliminate conjunctions within disjunctions or vica versa.

Example:

$$\begin{aligned}
(\neg p \to \neg q) \to (p \to q) &\equiv \neg(\neg\neg p \vee \neg q) \vee (\neg p \vee q) \\
&\equiv (\neg\neg\neg p \wedge \neg\neg q) \vee (\neg p \vee q) \\
&\equiv (\neg p \wedge q) \vee (\neg p \vee q) \\
&\equiv (\neg p \vee \neg p \vee q) \wedge (q \vee \neg p \vee q).
\end{aligned}$$

Clausal form

Clausal form is another representation of a CNF.

Clausal form

Clausal form is another representation of a CNF.

- A *clause* is a set of literals. Example: $\{\neg q, \neg p, q\}$.

Clausal form

Clausal form is another representation of a CNF.

- A *clause* is a set of literals. Example: $\{\neg q, \neg p, q\}$.
- A clause is considered to be an implicit disjunction of its literals. Example: $\{\neg q, \neg p, q\}$ is $\neg q \vee \neg p \vee q$.

Clausal form

Clausal form is another representation of a CNF.

- A *clause* is a set of literals. Example: $\{\neg q, \neg p, q\}$.
- A clause is considered to be an implicit disjunction of its literals. Example: $\{\neg q, \neg p, q\}$ is $\neg q \lor \neg p \lor q$.
- A *unit clause* is a clause consisting of exactly one literal. Example: $\{\neg q\}$.

Clausal form

Clausal form is another representation of a CNF.

- A *clause* is a set of literals. Example: $\{\neg q, \neg p, q\}$.
- A clause is considered to be an implicit disjunction of its literals. Example: $\{\neg q, \neg p, q\}$ is $\neg q \vee \neg p \vee q$.
- A *unit clause* is a clause consisting of exactly one literal. Example: $\{\neg q\}$.
- The empty set of literals is the *empty clause*, denoted by $\square$.

Clausal form

Clausal form is another representation of a CNF.

- A *clause* is a set of literals. Example: $\{\neg q, \neg p, q\}$.
- A clause is considered to be an implicit disjunction of its literals. Example: $\{\neg q, \neg p, q\}$ is $\neg q \lor \neg p \lor q$.
- A *unit clause* is a clause consisting of exactly one literal. Example: $\{\neg q\}$.
- The empty set of literals is the *empty clause*, denoted by $\Box$.
- A formula in *clausal form* is a set of clauses. Example: $\{\{p, r\}, \{\neg q, \neg p, q\}\}$.

Clausal form

Clausal form is another representation of a CNF.

- A *clause* is a set of literals. Example: $\{\neg q, \neg p, q\}$.
- A clause is considered to be an implicit disjunction of its literals. Example: $\{\neg q, \neg p, q\}$ is $\neg q \vee \neg p \vee q$.
- A *unit clause* is a clause consisting of exactly one literal. Example: $\{\neg q\}$.
- The empty set of literals is the *empty clause*, denoted by $\square$.
- A formula in *clausal form* is a set of clauses. Example: $\{\{p, r\}, \{\neg q, \neg p, q\}\}$.
- A formula is considered to be an implicit conjunction of its clauses. Example: $(p \vee r) \wedge (\neg q \vee \neg p \vee q)$ for the previous one.

Clausal form

Clausal form is another representation of a CNF.

- A *clause* is a set of literals. Example: $\{\neg q, \neg p, q\}$.
- A clause is considered to be an implicit disjunction of its literals. Example: $\{\neg q, \neg p, q\}$ is $\neg q \vee \neg p \vee q$.
- A *unit clause* is a clause consisting of exactly one literal. Example: $\{\neg q\}$.
- The empty set of literals is the *empty clause*, denoted by $\Box$.
- A formula in *clausal form* is a set of clauses. Example: $\{\{p, r\}, \{\neg q, \neg p, q\}\}$.
- A formula is considered to be an implicit conjunction of its clauses. Example: $(p \vee r) \wedge (\neg q \vee \neg p \vee q)$ for the previous one.
- The formula that is the empty set of clauses is denoted by $\varnothing$.

Removing trivial clauses

Corollary

Every formula in propositional logic can be transformed into an logically equivalent formula in clausal form.

Removing trivial clauses

Corollary

Every formula in propositional logic can be transformed into an logically equivalent formula in clausal form.

multiple occurrences of literals and clauses $\Rightarrow$ single occurence

Removing trivial clauses

Corollary

Every formula in propositional logic can be transformed into an logically equivalent formula in clausal form.

multiple occurrences of literals and clauses $\Rightarrow$ single occurence
equivalence due to idempotent laws ($A \vee A \equiv A, \; A \wedge A \equiv A$)

Removing trivial clauses

Corollary

Every formula in propositional logic can be transformed into an logically equivalent formula in clausal form.

multiple occurrences of literals and clauses $\Rightarrow$ single occurence
equivalence due to idempotent laws ($A \vee A \equiv A,\ A \wedge A \equiv A$)

Example: CNF:
$(p \vee r) \wedge (\neg q \vee \neg p \vee q) \wedge (p \vee \neg p \vee q \vee p \vee \neg p) \wedge (r \vee p)$

Removing trivial clauses

Corollary

Every formula in propositional logic can be transformed into an logically equivalent formula in clausal form.

multiple occurrences of literals and clauses $\Rightarrow$ single occurence
equivalence due to idempotent laws ($A \vee A \equiv A,\ A \wedge A \equiv A$)

Example: CNF:
$(p \vee r) \wedge (\neg q \vee \neg p \vee q) \wedge (p \vee \neg p \vee q \vee p \vee \neg p) \wedge (r \vee p)$

Clausal form: $\{\{p, r\}, \{\neg q, \neg p, q\}, \{p, \neg p, q\}\}$

Removing trivial clauses

Corollary

Every formula in propositional logic can be transformed into an logically equivalent formula in clausal form.

multiple occurrences of literals and clauses $\Rightarrow$ single occurence
equivalence due to idempotent laws ($A \vee A \equiv A,\ A \wedge A \equiv A$)

Example: CNF:
$(p \vee r) \wedge (\neg q \vee \neg p \vee q) \wedge (p \vee \neg p \vee q \vee p \vee \neg p) \wedge (r \vee p)$

Clausal form: $\{\{p, r\}, \{\neg q, \neg p, q\}, \{p, \neg p, q\}\}$

A clause is called **trivial** if it contains a pair of clashing literals.

Removing trivial clauses

Corollary

Every formula in propositional logic can be transformed into an logically equivalent formula in clausal form.

multiple occurrences of literals and clauses $\Rightarrow$ single occurence equivalence due to idempotent laws ($A \vee A \equiv A,\ A \wedge A \equiv A$)

Example: CNF:
$(p \vee r) \wedge (\neg q \vee \neg p \vee q) \wedge (p \vee \neg p \vee q \vee p \vee \neg p) \wedge (r \vee p)$

Clausal form: $\{\{p, r\}, \{\neg q, \neg p, q\}, \{p, \neg p, q\}\}$

A clause is called **trivial** if it contains a pair of clashing literals.

Proposition

Let S be a set of clauses and let $C \in S$ be a trivial clause. Then $S - \{C\}$ is logically equivalent to S.

Removing trivial clauses

Corollary

Every formula in propositional logic can be transformed into an logically equivalent formula in clausal form.

multiple occurrences of literals and clauses $\Rightarrow$ single occurence
equivalence due to idempotent laws ($A \vee A \equiv A,\ A \wedge A \equiv A$)

Example: CNF:
$(p \vee r) \wedge (\neg q \vee \neg p \vee q) \wedge (p \vee \neg p \vee q \vee p \vee \neg p) \wedge (r \vee p)$

Clausal form: $\{\{p, r\}, \{\neg q, \neg p, q\}, \{p, \neg p, q\}\}$

A clause is called **trivial** if it contains a pair of clashing literals.

Proposition

Let S be a set of clauses and let $C \in S$ be a trivial clause. Then $S - \{C\}$ is logically equivalent to S.

True, because of $A \vee \top \equiv \top$ and $A \wedge \top \equiv A$.

Removing trivial clauses

Corollary

Every formula in propositional logic can be transformed into an logically equivalent formula in clausal form.

multiple occurrences of literals and clauses $\Rightarrow$ single occurence
equivalence due to idempotent laws ($A \vee A \equiv A, \ A \wedge A \equiv A$)

Example: CNF:
$(p \vee r) \wedge (\neg q \vee \neg p \vee q) \wedge (p \vee \neg p \vee q \vee p \vee \neg p) \wedge (r \vee p)$

Clausal form: $\{\{p, r\}, \{\neg q, \neg p, q\}, \{p, \neg p, q\}\}$

A clause is called **trivial** if it contains a pair of clashing literals.

Proposition

Let S be a set of clauses and let $C \in S$ be a trivial clause. Then $S - \{C\}$ is logically equivalent to S.

True, because of $A \vee \top \equiv \top$ and $A \wedge \top \equiv A$. So we can delete trivial clauses.

Empty clause and the empty set of clauses

Proposition

$\square$ (empty clause) is unsatisfiable. $\varnothing$ (the empty set of clauses) is valid.

Empty clause and the empty set of clauses

Proposition

$\square$ (empty clause) is unsatisfiable. $\varnothing$ (the empty set of clauses) is valid.

Proof: A clause is satisfiable iff there is some interpretation under which at least one literal in the clause is true.

Empty clause and the empty set of clauses

Proposition

$\square$ (empty clause) is unsatisfiable. $\varnothing$ (the empty set of clauses) is valid.

Proof: A clause is satisfiable iff there is some interpretation under which at least one literal in the clause is true.

Let I be an arbitrary interpretation. Since there are no literals in $\square$, there are no literals whose value is true under I.

Empty clause and the empty set of clauses

Proposition

$\Box$ (empty clause) is unsatisfiable. $\emptyset$ (the empty set of clauses) is valid.

Proof: A clause is satisfiable iff there is some interpretation under which at least one literal in the clause is true.

Let I be an arbitrary interpretation. Since there are no literals in $\Box$, there are no literals whose value is true under I.

But I was an arbitrary interpretation, so is unsatisfiable.

Empty clause and the empty set of clauses

Proposition

$\square$ (empty clause) is unsatisfiable. $\emptyset$ (the empty set of clauses) is valid.

Proof: A clause is satisfiable iff there is some interpretation under which at least one literal in the clause is true.

Let I be an arbitrary interpretation. Since there are no literals in $\square$, there are no literals whose value is true under I.

But I was an arbitrary interpretation, so is unsatisfiable.

A set of clauses is valid iff every clause in the set is true in every interpretation.

Empty clause and the empty set of clauses

Proposition

$\square$ (empty clause) is unsatisfiable. $\varnothing$ (the empty set of clauses) is valid.

Proof: A clause is satisfiable iff there is some interpretation under which at least one literal in the clause is true.

Let I be an arbitrary interpretation. Since there are no literals in $\square$, there are no literals whose value is true under I.

But I was an arbitrary interpretation, so is unsatisfiable.

A set of clauses is valid iff every clause in the set is true in every interpretation.

But there are no clauses in $\varnothing$ that need to be true, so $\varnothing$ is valid.

Empty clause and the empty set of clauses

Proposition

$\square$ (empty clause) is unsatisfiable. $\emptyset$ (the empty set of clauses) is valid.

Proof: A clause is satisfiable iff there is some interpretation under which at least one literal in the clause is true.

Let I be an arbitrary interpretation. Since there are no literals in $\square$, there are no literals whose value is true under I.

But I was an arbitrary interpretation, so is unsatisfiable.

A set of clauses is valid iff every clause in the set is true in every interpretation.

But there are no clauses in $\emptyset$ that need to be true, so $\emptyset$ is valid.

A short notation for clausal form

Notation

The set delimiters $\{$ and $\}$ are removed from each clause and a negated literal is denoted by a bar over the atomic proposition.

A short notation for clausal form

Notation

The set delimiters $\{$ and $\}$ are removed from each clause and a negated literal is denoted by a bar over the atomic proposition.

Let $\mathrm{CNF}(A)$ and $\mathrm{cf}(A)$ denote a CNF and a clausal form for a formula A, respectively.

A short notation for clausal form

Notation

The set delimiters { and } are removed from each clause and a negated literal is denoted by a bar over the atomic proposition.

Let CNF(A) and cf(A) denote a CNF and a clausal form for a formula A, respectively.

Example:

$$A = (p \vee r) \wedge (q \rightarrow \neg p \vee q) \wedge (p \vee \neg p \vee q \vee p \vee \neg p) \wedge (r \vee p)$$

A short notation for clausal form

Notation

The set delimiters $\{$ and $\}$ are removed from each clause and a negated literal is denoted by a bar over the atomic proposition.

Let $\mathrm{CNF}(A)$ and $\mathrm{cf}(A)$ denote a CNF and a clausal form for a formula A, respectively.

Example:

$$A = (p \vee r) \wedge (q \rightarrow \neg p \vee q) \wedge (p \vee \neg p \vee q \vee p \vee \neg p) \wedge (r \vee p)$$

$$\mathrm{CNF}(A) = (p \vee r) \wedge (\neg q \vee \neg p \vee q) \wedge (p \vee \neg p \vee q \vee p \vee \neg p) \wedge (r \vee p)$$

A short notation for clausal form

Notation

The set delimiters { and } are removed from each clause and a negated literal is denoted by a bar over the atomic proposition.

Let CNF(A) and cf(A) denote a CNF and a clausal form for a formula A, respectively.

Example:

$A = (p \vee r) \wedge (q \rightarrow \neg p \vee q) \wedge (p \vee \neg p \vee q \vee p \vee \neg p) \wedge (r \vee p)$

$\mathrm{CNF}(A) = (p \vee r) \wedge (\neg q \vee \neg p \vee q) \wedge (p \vee \neg p \vee q \vee p \vee \neg p) \wedge (r \vee p)$

$\mathrm{cf}(A) = \{\{p, r\}, \{\neg q, \neg p, q\}, \{p, \neg p, q\}\}$ which becomes

A short notation for clausal form

Notation

The set delimiters { and } are removed from each clause and a negated literal is denoted by a bar over the atomic proposition.

Let CNF(A) and cf(A) denote a CNF and a clausal form for a formula A, respectively.

Example:

$A = (p \vee r) \wedge (q \rightarrow \neg p \vee q) \wedge (p \vee \neg p \vee q \vee p \vee \neg p) \wedge (r \vee p)$

$\mathrm{CNF}(A) = (p \vee r) \wedge (\neg q \vee \neg p \vee q) \wedge (p \vee \neg p \vee q \vee p \vee \neg p) \wedge (r \vee p)$

$\mathrm{cf}(A) = \{\{p, r\}, \{\neg q, \neg p, q\}, \{p, \neg p, q\}\}$ which becomes

$\mathrm{cf}(A) = \{pr, \bar{q}\bar{p}q, p\bar{p}q\}$ with the shorter notation.

Logical consequence in propositional logic

Theorem

Let $U = \{A_1, \dots A_n\}$ be a finite set of formulas and let B be a formula. Then the following statements are equivalent.

- $\{A_1, \dots A_n\} \models B$

Logical consequence in propositional logic

Theorem

Let $U = \{A_1, \ldots A_n\}$ be a finite set of formulas and let B be a formula. Then the following statements are equivalent.

- $\{A_1, \ldots A_n\} \models B$
- $\{A_1 \wedge \cdots \wedge A_n\} \models B$

Logical consequence in propositional logic

Theorem

Let $U = \{A_1, \dots A_n\}$ be a finite set of formulas and let B be a formula. Then the following statements are equivalent.

- $\{A_1, \dots A_n\} \models B$
- $\{A_1 \wedge \dots \wedge A_n\} \models B$
- $\models A_1 \wedge \dots \wedge A_n \rightarrow B$

Logical consequence in propositional logic

Theorem

Let $U = \{A_1, \dots A_n\}$ be a finite set of formulas and let B be a formula. Then the following statements are equivalent.

- $\{A_1, \dots A_n\} \models B$
- $\{A_1 \wedge \cdots \wedge A_n\} \models B$
- $\models A_1 \wedge \cdots \wedge A_n \rightarrow B$
- $\models \neg A_1 \vee \cdots \vee \neg A_n \vee B$

Logical consequence in propositional logic

Theorem

Let $U = \{A_1, \ldots A_n\}$ be a finite set of formulas and let B be a formula. Then the following statements are equivalent.

- $\{A_1, \ldots A_n\} \models B$
- $\{A_1 \wedge \cdots \wedge A_n\} \models B$
- $\models A_1 \wedge \cdots \wedge A_n \rightarrow B$
- $\models \neg A_1 \vee \cdots \vee \neg A_n \vee B$
- $\models A_1 \rightarrow (A_2 \rightarrow (\cdots (A_n \rightarrow B) \cdots))$

Logical consequence in propositional logic

Theorem

Let $U = \{A_1, \ldots A_n\}$ be a finite set of formulas and let B be a formula. Then the following statements are equivalent.

- $\{A_1, \ldots A_n\} \models B$
- $\{A_1 \wedge \cdots \wedge A_n\} \models B$
- $\models A_1 \wedge \cdots \wedge A_n \rightarrow B$
- $\models \neg A_1 \vee \cdots \vee \neg A_n \vee B$
- $\models A_1 \rightarrow (A_2 \rightarrow (\cdots (A_n \rightarrow B) \cdots))$
- $A_1 \wedge \cdots \wedge A_n \wedge \neg B$ unsatisfiable

Logical consequence in propositional logic

Theorem

Let $U = \{A_1, \dots A_n\}$ be a finite set of formulas and let B be a formula. Then the following statements are equivalent.

- $\{A_1, \dots A_n\} \models B$
- $\{A_1 \wedge \cdots \wedge A_n\} \models B$
- $\models A_1 \wedge \cdots \wedge A_n \rightarrow B$
- $\models \neg A_1 \vee \cdots \vee \neg A_n \vee B$
- $\models A_1 \rightarrow (A_2 \rightarrow (\cdots (A_n \rightarrow B) \cdots)$
- $A_1 \wedge \cdots \wedge A_n \wedge \neg B$ unsatisfiable
- $\{A_1, \dots, A_n, \neg B\}$ unsatisfiable

Logical consequence in propositional logic

Theorem

Let $U = \{A_1, \ldots A_n\}$ be a finite set of formulas and let B be a formula. Then the following statements are equivalent.

- $\{A_1, \ldots A_n\} \models B$
- $\{A_1 \wedge \cdots \wedge A_n\} \models B$
- $\models A_1 \wedge \cdots \wedge A_n \rightarrow B$
- $\models \neg A_1 \vee \cdots \vee \neg A_n \vee B$
- $\models A_1 \rightarrow (A_2 \rightarrow (\cdots (A_n \rightarrow B) \cdots)$
- $A_1 \wedge \cdots \wedge A_n \wedge \neg B$ unsatisfiable
- $\{A_1, \ldots, A_n, \neg B\}$ unsatisfiable
- $\{\mathrm{CNF}(A_1), \ldots, \mathrm{CNF}(A_n), \mathrm{CNF}(\neg B)\}$ unsatisfiable

Logical consequence in propositional logic

Theorem

Let $U = \{A_1, \dots A_n\}$ be a finite set of formulas and let B be a formula. Then the following statements are equivalent.

- $\{A_1, \dots A_n\} \models B$
- $\{A_1 \wedge \dots \wedge A_n\} \models B$
- $\models A_1 \wedge \dots \wedge A_n \rightarrow B$
- $\models \neg A_1 \vee \dots \vee \neg A_n \vee B$
- $\models A_1 \rightarrow (A_2 \rightarrow (\cdots (A_n \rightarrow B) \cdots))$
- $A_1 \wedge \dots \wedge A_n \wedge \neg B$ unsatisfiable
- $\{A_1, \dots, A_n, \neg B\}$ unsatisfiable
- $\{\mathrm{CNF}(A_1), \dots, \mathrm{CNF}(A_n), \mathrm{CNF}(\neg B)\}$ unsatisfiable
- $\mathrm{cf}(A_1) \cup \dots \cup \mathrm{cf}(A_n) \cup \mathrm{cf}(\neg B)$ unsatisfiable

Logical consequence in propositional logic

Theorem

Let $U = \{A_1, \dots A_n\}$ be a finite set of formulas and let B be a formula. Then the following statements are equivalent.

- $\{A_1, \dots A_n\} \models B$
- $\{A_1 \wedge \cdots \wedge A_n\} \models B$
- $\models A_1 \wedge \cdots \wedge A_n \rightarrow B$
- $\models \neg A_1 \vee \cdots \vee \neg A_n \vee B$
- $\models A_1 \rightarrow (A_2 \rightarrow (\cdots (A_n \rightarrow B) \cdots))$
- $A_1 \wedge \cdots \wedge A_n \wedge \neg B$ unsatisfiable
- $\{A_1, \dots, A_n, \neg B\}$ unsatisfiable
- $\{\mathrm{CNF}(A_1), \dots, \mathrm{CNF}(A_n), \mathrm{CNF}(\neg B)\}$ unsatisfiable
- $\mathrm{cf}(A_1) \cup \cdots \cup \mathrm{cf}(A_n) \cup \mathrm{cf}(\neg B)$ unsatisfiable

In order to decide logical consequence it is enough to decide whether a set of clauses is unsatisfiable.

Resolution rule

Definition and example

If ℓ is a literal, let ℓ^c denote its complementary pair.

Resolution rule

Definition and example

If ℓ is a literal, let ℓ^c denote its complementary pair.

Definition

Let C_1, C_2 be clauses such that $\ell \in C_1, \ell^c \in C_2$. The clauses C_1, C_2 are said to be **clashing clauses** and to **clash on the complementary pair of literals** ℓ, ℓ^c . C, the **resolvent** of C_1 and C_2 , is the clause:

$$\mathrm{Res}(C_1, C_2) = (C_1 - \{l\}) \cup (C_2 - \{l^c\}).$$

C_1 and C_2 are the parent clauses of C.

Resolution rule

Definition and example

If ℓ is a literal, let ℓ^c denote its complementary pair.

Definition

Let C_1, C_2 be clauses such that $\ell \in C_1, \ell^c \in C_2$. The clauses C_1, C_2 are said to be clashing clauses and to clash on the complementary pair of literals ℓ, ℓ^c . C, the resolvent of C_1 and C_2 , is the clause:

$$\mathrm{Res}(C_1, C_2) = (C_1 - \{l\}) \cup (C_2 - \{l^c\}).$$

C_1 and C_2 are the parent clauses of C.

Example: $C_1 = ab\bar{c}$ and $C_2 = bc\bar{e}$

Resolution rule

Definition and example

If ℓ is a literal, let ℓ^c denote its complementary pair.

Definition

Let C_1, C_2 be clauses such that $\ell \in C_1, \ell^c \in C_2$. The clauses C_1, C_2 are said to be clashing clauses and to clash on the complementary pair of literals ℓ, ℓ^c . C, the resolvent of C_1 and C_2 , is the clause:

$$\mathrm{Res}(C_1, C_2) = (C_1 - \{l\}) \cup (C_2 - \{l^c\}).$$

C_1 and C_2 are the parent clauses of C.

Example: $C_1 = ab\bar{c}$ and $C_2 = bc\bar{e}$

They clash on the pair of complementary literals $c, \bar{c}$.

Resolution rule

Definition and example

If ℓ is a literal, let ℓ^c denote its complementary pair.

Definition

Let C_1, C_2 be clauses such that $\ell \in C_1, \ell^c \in C_2$. The clauses C_1, C_2 are said to be **clashing clauses** and to **clash on the complementary pair of literals** ℓ, ℓ^c . C, the **resolvent** of C_1 and C_2 , is the clause:

$$\mathrm{Res}(C_1, C_2) = (C_1 - \{l\}) \cup (C_2 - \{l^c\}).$$

C_1 and C_2 are the parent clauses of C.

Example: $C_1 = ab\bar{c}$ and $C_2 = bc\bar{e}$

They clash on the pair of complementary literals $c, \bar{c}$.

$\mathrm{Res}(C_1, C_2) = (ab\bar{c} - \{\bar{c}\}) \cup (bc\bar{e} - \{c\}) = ab \cup b\bar{e} = ab\bar{e}$.

Resolution rule

Definition and example

If ℓ is a literal, let ℓ^c denote its complementary pair.

Definition

Let C_1, C_2 be clauses such that $\ell \in C_1, \ell^c \in C_2$. The clauses C_1, C_2 are said to be clashing clauses and to clash on the complementary pair of literals ℓ, ℓ^c . C, the resolvent of C_1 and C_2 , is the clause:

$$\mathrm{Res}(C_1, C_2) = (C_1 - \{l\}) \cup (C_2 - \{l^c\}).$$

C_1 and C_2 are the parent clauses of C.

Example: $C_1 = ab\bar{c}$ and $C_2 = bc\bar{e}$

They clash on the pair of complementary literals $c, \bar{c}$.

$\mathrm{Res}(C_1, C_2) = (ab\bar{c} - \{\bar{c}\}) \cup (bc\bar{e} - \{c\}) = ab \cup b\bar{e} = ab\bar{e}$.

Recall that a clause is a set so duplicated literals are removed when taking the union.

Resolution

Case of more than one pair of clashing clauses

Resolution is only performed if the pair of clauses clash on exactly one pair of complementary literals due to the following.

Resolution

Case of more than one pair of clashing clauses

Resolution is only performed if the pair of clauses clash on exactly one pair of complementary literals due to the following.

Proposition

If two clauses clash on more than one literal, their resolvent is a trivial clause.

Resolution

Case of more than one pair of clashing clauses

Resolution is only performed if the pair of clauses clash on exactly one pair of complementary literals due to the following.

Proposition

If two clauses clash on more than one literal, their resolvent is a trivial clause.

Remark: It is not strictly incorrect to perform resolution on such clauses, but since trivial clauses contribute nothing to the satisfiability or unsatisfiability of a set of clauses, we agree to delete them from any set of clauses and not to perform resolution on clauses with two clashing pairs of literals.

Resolution procedure

Lemma

Let I be an interpretation. If $I \models \{C_1, C_2\}$ then $I \models \mathrm{Res}(C_1, C_2)$. If $I \models \mathrm{Res}(C_1, C_2)$, then I can be extended to $\hat{I}$, such that $\hat{I} \models \{C_1, C_2\}$.

Resolution procedure

Lemma

Let I be an interpretation. If $I \models \{C_1, C_2\}$ then $I \models \mathrm{Res}(C_1, C_2)$. If $I \models \mathrm{Res}(C_1, C_2)$, then I can be extended to $\hat{I}$, such that $\hat{I} \models \{C_1, C_2\}$.

Corollary

Let S be a set of clauses and let $C_1, C_2 \in S$ be a pair of clashing clauses. Then S is satisfiable if and only if $S \cup \{\mathrm{Res}(C_1, C_2)\}$ is satisfiable.

If a set of clauses S contains $\square$ then S is unsatisfiable.

Resolution procedure

Algorithm

Let $\binom{S}{2}$ denote the set of 2-element subsets of S.

Resolution procedure

Algorithm

Let $\binom{S}{2}$ denote the set of 2-element subsets of S.

Algorithm RESOLUTION PROCEDURE(S)

while there is an unmarked pair of $\binom{S}{2}$ **do**
    choose an unmarked pair $\{C_1, C_2\}$ of $\binom{S}{2}$ and mark it
    if $\{C_1, C_2\}$ is a clashing pair of clauses **then**
        $C \leftarrow \mathrm{Res}(C_1, C_2)$
        if $C = \square$ **then**
            return 'S is unsatisfiable'
        else
            if C is not the trivial clause **then**
                $S \leftarrow S \cup \{C\}$
return 'S is satisfiable'

Resolution procedure

Example

Consider the set of clauses

$$S = \{\, (1)\, p,\ \ (2)\, \bar{p}q,\ \ (3)\, \bar{r},\ \ (4)\, \bar{p}\bar{q}r \,\},$$

Resolution procedure

Example

Consider the set of clauses

$$S = \{\,(1)\, p,\ \ (2)\, \bar{p}q,\ \ (3)\, \bar{r},\ \ (4)\, \bar{p}\bar{q}r\,\},$$

where the clauses have been numbered. Here is a resolution derivation of $\square$ from S, where the justification for each line is the pair of the numbers of the parent clauses that have been resolved to give the resolvent clause:

Resolution procedure

Example

Consider the set of clauses

$$S = \{\, (1)\, p,\ (2)\, \bar{p}q,\ (3)\, \bar{r},\ (4)\, \bar{p}\bar{q}r \,\},$$

where the clauses have been numbered. Here is a resolution derivation of $\square$ from S, where the justification for each line is the pair of the numbers of the parent clauses that have been resolved to give the resolvent clause:

(5)	$\bar{p}\bar{q}$	Res((3),(4))
(6)	$\bar{p}$	Res((5),(2))
(7)	$\square$	Res((6),(1))

Resolution procedure

An algorithm for a decision problem (only 'yes' or 'no' outputs) is said to be **sound** if it never gives wrong answer for the 'yes' instances.

Resolution procedure

An algorithm for a decision problem (only 'yes' or 'no' outputs) is said to be **sound** if it never gives wrong answer for the 'yes' instances. An algorithm is said to be **complete** if it gives correct answer for all 'yes' instances.

Resolution procedure

An algorithm for a decision problem (only 'yes' or 'no' outputs) is said to be **sound** if it never gives wrong answer for the 'yes' instances. An algorithm is said to be **complete** if it gives correct answer for all 'yes' instances. So a sound and complete algorithm for a decision problem gives a 'yes' answer exactly for the 'yes' instances.

Resolution procedure

An algorithm for a decision problem (only 'yes' or 'no' outputs) is said to be **sound** if it never gives wrong answer for the 'yes' instances. An algorithm is said to be **complete** if it gives correct answer for all 'yes' instances. So a sound and complete algorithm for a decision problem gives a 'yes' answer exactly for the 'yes' instances.

Theorem

The resolution procedure always halts and it is sound and complete.

(without proof)