Principles of Knowledge Representation and Reasoning

Propositional Logic

Albert-Ludwigs-Universität Freiburg

UNI FREIBURG

Bernhard Nebel, Stefan Wölfl, and Julien Hué

October 24, 2012

Why Logic?

Why logic?

- Logic is one of the best developed systems for representing knowledge.
- Can be used for analysis, design and specification.
- Understanding formal logic is a prerequisite for understanding most research papers in KRR.

The right logic…

- Logics of different orders (1st, 2nd, ...)
- Modal logics
  - epistemic
  - temporal
  - dynamic (program)
  - multi-modal logics
  - …
- Many-valued logics
- Nonmonotonic logics
- Intuitionistic logics
- …

The right logic…

- Logics of different orders (1st, 2nd, ...)
- Modal logics
  - epistemic
  - temporal
  - dynamic (program)
  - multi-modal logics
  - …
- Many-valued logics
- Nonmonotonic logics
- Intuitionistic logics
- …

The right logic…

- Logics of different orders (1st, 2nd, ...)
- Modal logics
  - epistemic
  - temporal
  - dynamic (program)
  - multi-modal logics
  - …
- Many-valued logics
- Nonmonotonic logics
- Intuitionistic logics
- …

The right logic…

- Logics of different orders (1st, 2nd, ...)
- Modal logics
  - epistemic
  - temporal
  - dynamic (program)
  - multi-modal logics
  - …
- Many-valued logics
- Nonmonotonic logics
- Intuitionistic logics
- …

The right logic…

- Logics of different orders (1st, 2nd, ...)
- Modal logics
  - epistemic
  - temporal
  - dynamic (program)
  - multi-modal logics
  - …
- Many-valued logics
- Nonmonotonic logics
- Intuitionistic logics
- …

The right logic…

- Logics of different orders (1st, 2nd, ...)
- Modal logics
  - epistemic
  - temporal
  - dynamic (program)
  - multi-modal logics
  - …
- Many-valued logics
- Nonmonotonic logics
- Intuitionistic logics
- …

The right logic…

- Logics of different orders (1st, 2nd, ...)
- Modal logics
  - epistemic
  - temporal
  - dynamic (program)
  - multi-modal logics
  - …
- Many-valued logics
- Nonmonotonic logics
- Intuitionistic logics
- …

The logical approach

- Define a formal language: logical & non-logical symbols, syntax rules
- Provide language with compositional semantics
  - Fix universe of discourse
  - Specify how the non-logical symbols can be interpreted: interpretation
  - Rules how to combine interpretation of single symbols
  - Satisfying interpretation = model
  - Semantics often entails concept of logical implication/entailment
- Specify a calculus that allows to derive new formulae from old ones – according to the entailment relation

The logical approach

- Define a formal language: logical & non-logical symbols, syntax rules
- Provide language with compositional semantics
  - Fix universe of discourse
  - Specify how the non-logical symbols can be interpreted: interpretation
  - Rules how to combine interpretation of single symbols
  - Satisfying interpretation = model
  - Semantics often entails concept of logical implication/entailment
- Specify a calculus that allows to derive new formulae from old ones – according to the entailment relation

The logical approach

- Define a formal language: logical & non-logical symbols, syntax rules
- Provide language with compositional semantics
  - Fix universe of discourse
  - Specify how the non-logical symbols can be interpreted: interpretation
  - Rules how to combine interpretation of single symbols
  - Satisfying interpretation = model
  - Semantics often entails concept of logical implication/entailment
- Specify a calculus that allows to derive new formulae from old ones – according to the entailment relation

The logical approach

- Define a formal language: logical & non-logical symbols, syntax rules
- Provide language with compositional semantics
  - Fix universe of discourse
  - Specify how the non-logical symbols can be interpreted: interpretation
  - Rules how to combine interpretation of single symbols
  - Satisfying interpretation = model
  - Semantics often entails concept of logical implication/entailment
- Specify a calculus that allows to derive new formulae from old ones – according to the entailment relation

Propositional Logic

Propositional logic: main ideas

- Non-logical symbols: propositional variables or atoms
  - representing propositions which cannot be decomposed
  - which can be true or false (for example: “Snow is white”, “It rains”)
- Logical symbols: propositional connectives such as: and ($\wedge$), or ($\vee$), and not ($\neg$)
- Formulae: built out of atoms and connectives
- Universe of discourse: truth values

Propositional logic: main ideas

- Non-logical symbols: propositional variables or atoms
  - representing propositions which cannot be decomposed
  - which can be true or false (for example: “Snow is white”, “It rains”)
- Logical symbols: propositional connectives such as: and ($\wedge$), or ($\vee$), and not ($\neg$)
- Formulae: built out of atoms and connectives
- Universe of discourse: truth values

Propositional logic: main ideas

- Non-logical symbols: propositional variables or atoms
  - representing propositions which cannot be decomposed
  - which can be true or false (for example: “Snow is white”, “It rains”)
- Logical symbols: propositional connectives such as: and ($\wedge$), or ($\vee$), and not ($\neg$)
- Formulae: built out of atoms and connectives
- Universe of discourse: truth values

Syntax

Syntax

Countable alphabet Σ of atomic propositions: $a, b, c, \ldots$
Propositional formulae are built according to the following rule:

$$
\begin{array}{lcll}
\varphi & \longrightarrow & a & \text{atomic formula} \\
 & | & \bot & \text{falsity} \\
 & | & \top & \text{truth} \\
 & | & \neg\varphi' & \text{negation} \\
 & | & (\varphi' \wedge \varphi'') & \text{conjunction} \\
 & | & (\varphi' \vee \varphi'') & \text{disjunction} \\
 & | & (\varphi' \rightarrow \varphi'') & \text{implication} \\
 & | & (\varphi' \leftrightarrow \varphi'') & \text{equivalence}
\end{array}
$$

Parentheses can be omitted if no ambiguity arises.
Operator precedence: $\neg > \wedge > \vee > \rightarrow = \leftrightarrow$.

Syntax

Countable alphabet Σ of atomic propositions: $a, b, c, \ldots$
Propositional formulae are built according to the following rule:

φ	$\longrightarrow$	a	atomic formula
	$\mid$	$\bot$	falsity
	$\mid$	$\top$	truth
	$\mid$	$\neg\varphi'$	negation
	$\mid$	$(\varphi' \wedge \varphi'')$	conjunction
	$\mid$	$(\varphi' \vee \varphi'')$	disjunction
	$\mid$	$(\varphi' \to \varphi'')$	implication
	$\mid$	$(\varphi' \leftrightarrow \varphi'')$	equivalence

Parentheses can be omitted if no ambiguity arises.
Operator precedence: $\neg > \wedge > \vee > \ \to \ = \ \leftrightarrow$.

Syntax

Countable alphabet Σ of atomic propositions: $a, b, c, \ldots$
Propositional formulae are built according to the following rule:

φ	$\longrightarrow$	a	atomic formula
	$\mid$	$\perp$	falsity
	$\mid$	$\top$	truth
	$\mid$	$\neg\varphi'$	negation
	$\mid$	$(\varphi' \wedge \varphi'')$	conjunction
	$\mid$	$(\varphi' \vee \varphi'')$	disjunction
	$\mid$	$(\varphi' \rightarrow \varphi'')$	implication
	$\mid$	$(\varphi' \leftrightarrow \varphi'')$	equivalence

Parentheses can be omitted if no ambiguity arises.
Operator precedence: $\neg > \wedge > \vee > \rightarrow = \leftrightarrow$.

Countable alphabet Σ of atomic propositions: $a, b, c, \ldots$
Propositional formulae are built according to the following rule:

φ	$\longrightarrow$	a	atomic formula
	$\mid$	$\bot$	falsity
	$\mid$	$\top$	truth
	$\mid$	$\neg\varphi'$	negation
	$\mid$	$(\varphi' \wedge \varphi'')$	conjunction
	$\mid$	$(\varphi' \vee \varphi'')$	disjunction
	$\mid$	$(\varphi' \rightarrow \varphi'')$	implication
	$\mid$	$(\varphi' \leftrightarrow \varphi'')$	equivalence

Parentheses can be omitted if no ambiguity arises.
Operator precedence: $\neg > \wedge > \vee > \rightarrow = \leftrightarrow$.

Syntax

Countable alphabet Σ of atomic propositions: $a, b, c, \ldots$
Propositional formulae are built according to the following rule:

φ	$\longrightarrow$	a	atomic formula
	$\mid$	$\bot$	falsity
	$\mid$	$\top$	truth
	$\mid$	$\neg\varphi'$	negation
	$\mid$	$(\varphi' \wedge \varphi'')$	conjunction
	$\mid$	$(\varphi' \vee \varphi'')$	disjunction
	$\mid$	$(\varphi' \rightarrow \varphi'')$	implication
	$\mid$	$(\varphi' \leftrightarrow \varphi'')$	equivalence

Parentheses can be omitted if no ambiguity arises.
Operator precedence: $\neg > \wedge > \vee > \rightarrow = \leftrightarrow$.

Syntax

Countable alphabet Σ of atomic propositions: $a, b, c, \ldots$
Propositional formulae are built according to the following rule:

φ	$\longrightarrow$	a	atomic formula
	$\mid$	$\bot$	falsity
	$\mid$	$\top$	truth
	$\mid$	$\neg\varphi'$	negation
	$\mid$	$(\varphi' \land \varphi'')$	conjunction
	$\mid$	$(\varphi' \lor \varphi'')$	disjunction
	$\mid$	$(\varphi' \rightarrow \varphi'')$	implication
	$\mid$	$(\varphi' \leftrightarrow \varphi'')$	equivalence

Parentheses can be omitted if no ambiguity arises.
Operator precedence: $\neg > \land > \lor > \rightarrow = \leftrightarrow$.

Countable alphabet Σ of atomic propositions: $a, b, c, \ldots$
Propositional formulae are built according to the following rule:

φ	$\longrightarrow$	a	atomic formula
	$\mid$	$\bot$	falsity
	$\mid$	$\top$	truth
	$\mid$	$\neg\varphi'$	negation
	$\mid$	$(\varphi' \wedge \varphi'')$	conjunction
	$\mid$	$(\varphi' \vee \varphi'')$	disjunction
	$\mid$	$(\varphi' \rightarrow \varphi'')$	implication
	$\mid$	$(\varphi' \leftrightarrow \varphi'')$	equivalence

Parentheses can be omitted if no ambiguity arises.
Operator precedence: $\neg > \wedge > \vee > \rightarrow = \leftrightarrow$.

Syntax

Countable alphabet Σ of atomic propositions: $a, b, c, \ldots$
Propositional formulae are built according to the following rule:

φ	$\longrightarrow$	a	atomic formula
	$\mid$	$\bot$	falsity
	$\mid$	$\top$	truth
	$\mid$	$\neg\varphi'$	negation
	$\mid$	$(\varphi' \wedge \varphi'')$	conjunction
	$\mid$	$(\varphi' \vee \varphi'')$	disjunction
	$\mid$	$(\varphi' \rightarrow \varphi'')$	implication
	$\mid$	$(\varphi' \leftrightarrow \varphi'')$	equivalence

Parentheses can be omitted if no ambiguity arises.
Operator precedence: $\neg > \wedge > \vee > \rightarrow = \leftrightarrow$.

Countable alphabet Σ of atomic propositions: $a, b, c, \ldots$
Propositional formulae are built according to the following rule:

φ	$\longrightarrow$	a	atomic formula
	$\mid$	$\bot$	falsity
	$\mid$	$\top$	truth
	$\mid$	$\neg\varphi'$	negation
	$\mid$	$(\varphi' \land \varphi'')$	conjunction
	$\mid$	$(\varphi' \lor \varphi'')$	disjunction
	$\mid$	$(\varphi' \rightarrow \varphi'')$	implication
	$\mid$	$(\varphi' \leftrightarrow \varphi'')$	equivalence

Parentheses can be omitted if no ambiguity arises.
Operator precedence: $\neg > \land > \lor > \rightarrow = \leftrightarrow$.

Countable alphabet Σ of atomic propositions: $a, b, c, \ldots$
Propositional formulae are built according to the following rule:

φ	$\longrightarrow$	a	atomic formula
	$\mid$	$\bot$	falsity
	$\mid$	$\top$	truth
	$\mid$	$\neg\varphi'$	negation
	$\mid$	$(\varphi' \wedge \varphi'')$	conjunction
	$\mid$	$(\varphi' \vee \varphi'')$	disjunction
	$\mid$	$(\varphi' \rightarrow \varphi'')$	implication
	$\mid$	$(\varphi' \leftrightarrow \varphi'')$	equivalence

Parentheses can be omitted if no ambiguity arises.
Operator precedence: $\neg > \wedge > \vee > \rightarrow = \leftrightarrow$.

Language and meta-language

- $(a \vee b)$ is an expression of the language of propositional logic.
- $\varphi \longrightarrow a | \ldots | (\varphi' \leftrightarrow \varphi'')$ is a statement about how expressions in the language of propositional logic can be formed. It is stated using meta-language.
- In order to describe how expressions (in this case formulae) can be formed, we use meta-language.
- When we describe how to interpret formulae, we use meta-language expressions.

Language and meta-language

- $(a \vee b)$ is an expression of the language of propositional logic.
- $\varphi \longrightarrow a | \ldots | (\varphi' \leftrightarrow \varphi'')$ is a statement about how expressions in the language of propositional logic can be formed. It is stated using meta-language.
- In order to describe how expressions (in this case formulae) can be formed, we use meta-language.
- When we describe how to interpret formulae, we use meta-language expressions.

Language and meta-language

- $(a \vee b)$ is an expression of the language of propositional logic.
- $\varphi \longrightarrow a | \ldots | (\varphi' \leftrightarrow \varphi'')$ is a statement about how expressions in the language of propositional logic can be formed. It is stated using meta-language.
- In order to describe how expressions (in this case formulae) can be formed, we use meta-language.
- When we describe how to interpret formulae, we use meta-language expressions.

Semantics

Semantics: idea

- Atomic propositions can be true $(1, T)$ or false $(0, F)$.
- Provided the truth values of the atoms have been fixed (truth assignment or interpretation), the truth value of a formula can be computed from the truth values of the atoms and the connectives.
- Example:

$$(a \vee b) \wedge c$$

  is true iff c is true and, additionally, a or b is true.

Logical implication can then be defined as follows:

- φ is implied by a set of formulae Θ iff φ is true for all truth assignments (world states) that make all formulae in Θ true.

Semantics: idea

- Atomic propositions can be true $(1, T)$ or false $(0, F)$.
- Provided the truth values of the atoms have been fixed (truth assignment or interpretation), the truth value of a formula can be computed from the truth values of the atoms and the connectives.
- Example:

$$(a \lor b) \land c$$

  is true iff c is true and, additionally, a or b is true.

Logical implication can then be defined as follows:

- φ is implied by a set of formulae Θ iff φ is true for all truth assignments (world states) that make all formulae in Θ true.

Semantics: idea

- Atomic propositions can be true $(1, T)$ or false $(0, F)$.
- Provided the truth values of the atoms have been fixed (truth assignment or interpretation), the truth value of a formula can be computed from the truth values of the atoms and the connectives.
- Example:

$$(a \vee b) \wedge c$$

is true iff c is true and, additionally, a or b is true.

Logical implication can then be defined as follows:

- φ is implied by a set of formulae Θ iff φ is true for all truth assignments (world states) that make all formulae in Θ true.

Semantics: idea

- Atomic propositions can be true $(1, T)$ or false $(0, F)$.
- Provided the truth values of the atoms have been fixed (truth assignment or interpretation), the truth value of a formula can be computed from the truth values of the atoms and the connectives.
- Example:

$$(a \vee b) \wedge c$$

  is true iff c is true and, additionally, a or b is true.

Logical implication can then be defined as follows:

- φ is implied by a set of formulae Θ iff φ is true for all truth assignments (world states) that make all formulae in Θ true.

Formal semantics

An interpretation or truth assignment over Σ is a function:

$$\mathcal{I} \colon \Sigma \to \{T, F\}.$$

A formula ψ is true under $\mathcal{I}$ or is satisfied by $\mathcal{I}$ (symb. $\mathcal{I} \models \psi$):

$$\begin{array}{rcl}
\mathcal{I} \models a & \text{iff} & \mathcal{I}(a) = T \\
& \mathcal{I} \models \top & \\
& \mathcal{I} \not\models \bot & \\
\mathcal{I} \models \neg\varphi & \text{iff} & \mathcal{I} \not\models \varphi \\
\mathcal{I} \models \varphi \wedge \varphi' & \text{iff} & \mathcal{I} \models \varphi \text{ and } \mathcal{I} \models \varphi' \\
\mathcal{I} \models \varphi \vee \varphi' & \text{iff} & \mathcal{I} \models \varphi \text{ or } \mathcal{I} \models \varphi' \\
\mathcal{I} \models \varphi \rightarrow \varphi' & \text{iff} & \text{if } \mathcal{I} \models \varphi \text{ then } \mathcal{I} \models \varphi' \\
\mathcal{I} \models \varphi \leftrightarrow \varphi' & \text{iff} & \mathcal{I} \models \varphi \text{ if and only if } \mathcal{I} \models \varphi'
\end{array}$$

Formal semantics

An interpretation or truth assignment over Σ is a function:

$$\mathcal{I} \colon \Sigma \to \{T, F\}.$$

A formula ψ is true under $\mathcal{I}$ or is satisfied by $\mathcal{I}$ (symb. $\mathcal{I} \models \psi$):

$$
\begin{array}{rcl}
\mathcal{I} \models a & \text{iff} & \mathcal{I}(a) = T \\
 & \mathcal{I} \models \top & \\
 & \mathcal{I} \not\models \bot & \\
\mathcal{I} \models \neg\varphi & \text{iff} & \mathcal{I} \not\models \varphi \\
\mathcal{I} \models \varphi \land \varphi' & \text{iff} & \mathcal{I} \models \varphi \text{ and } \mathcal{I} \models \varphi' \\
\mathcal{I} \models \varphi \lor \varphi' & \text{iff} & \mathcal{I} \models \varphi \text{ or } \mathcal{I} \models \varphi' \\
\mathcal{I} \models \varphi \to \varphi' & \text{iff} & \text{if } \mathcal{I} \models \varphi \text{ then } \mathcal{I} \models \varphi' \\
\mathcal{I} \models \varphi \leftrightarrow \varphi' & \text{iff} & \mathcal{I} \models \varphi \text{ if and only if } \mathcal{I} \models \varphi'
\end{array}
$$

Example

Given

$$\mathcal{I}: a \mapsto T,\ b \mapsto F,\ c \mapsto F,\ d \mapsto T,$$

Is $((a \vee b) \leftrightarrow (c \vee d)) \wedge (\neg(a \wedge c) \vee (c \wedge \neg d))$ true or false?

$$((a \vee b) \leftrightarrow (c \vee d)) \wedge (\neg(a \wedge c) \vee (c \wedge \neg d))$$

$$((a \vee b) \leftrightarrow (c \vee d)) \wedge (\neg(a \wedge c) \vee (c \wedge \neg d))$$

$$((a \vee b) \leftrightarrow (c \vee d)) \wedge (\neg(a \wedge c) \vee (c \wedge \neg d))$$

$$((a \vee b) \leftrightarrow (c \vee d)) \wedge (\neg(a \wedge c) \vee (c \wedge \neg d))$$

$$((a \vee b) \leftrightarrow (c \vee d)) \wedge (\neg(a \wedge c) \vee (c \wedge \neg d))$$

Example

Given

$$\mathcal{I}: a \mapsto T,\ b \mapsto F,\ c \mapsto F,\ d \mapsto T,$$

Is $((a \vee b) \leftrightarrow (c \vee d)) \wedge (\neg(a \wedge c) \vee (c \wedge \neg d))$ true or false?

$$((\mathbf{a} \vee \mathbf{b}) \leftrightarrow (\mathbf{c} \vee \mathbf{d})) \wedge (\neg(\mathbf{a} \wedge \mathbf{c}) \vee (\mathbf{c} \wedge \neg \mathbf{d}))$$

$$((\mathbf{a} \vee \mathbf{b}) \leftrightarrow (\mathbf{c} \vee \mathbf{d})) \wedge (\neg(\mathbf{a} \wedge \mathbf{c}) \vee (\mathbf{c} \wedge \neg \mathbf{d}))$$

$$((\mathbf{a} \vee \mathbf{b}) \leftrightarrow (\mathbf{c} \vee \mathbf{d})) \wedge (\neg(\mathbf{a} \wedge \mathbf{c}) \vee (\mathbf{c} \wedge \neg \mathbf{d}))$$

$$((\mathbf{a} \vee \mathbf{b}) \leftrightarrow (\mathbf{c} \vee \mathbf{d})) \wedge (\neg(\mathbf{a} \wedge \mathbf{c}) \vee (\mathbf{c} \wedge \neg \mathbf{d}))$$

$$((\mathbf{a} \vee \mathbf{b}) \leftrightarrow (\mathbf{c} \vee \mathbf{d})) \wedge (\neg(\mathbf{a} \wedge \mathbf{c}) \vee (\mathbf{c} \wedge \neg \mathbf{d}))$$

Example

Given

$$\mathcal{I}: a \mapsto T,\ b \mapsto F,\ c \mapsto F,\ d \mapsto T,$$

Is $((a \vee b) \leftrightarrow (c \vee d)) \wedge (\neg(a \wedge c) \vee (c \wedge \neg d))$ true or false?

$$((\mathbf{a} \vee \mathbf{b}) \leftrightarrow (\mathbf{c} \vee \mathbf{d})) \wedge (\neg(\mathbf{a} \wedge \mathbf{c}) \vee (\mathbf{c} \wedge \neg \mathbf{d}))$$

$$((\mathbf{a} \vee \mathbf{b}) \leftrightarrow (\mathbf{c} \vee \mathbf{d})) \wedge (\neg(\mathbf{a} \wedge \mathbf{c}) \vee (\mathbf{c} \wedge \neg \mathbf{d}))$$

$$((\mathbf{a} \vee \mathbf{b}) \leftrightarrow (\mathbf{c} \vee \mathbf{d})) \wedge (\neg(\mathbf{a} \wedge \mathbf{c}) \vee (\mathbf{c} \wedge \neg \mathbf{d}))$$

$$((\mathbf{a} \vee \mathbf{b}) \leftrightarrow (\mathbf{c} \vee \mathbf{d})) \wedge (\neg(\mathbf{a} \wedge \mathbf{c}) \vee (\mathbf{c} \wedge \neg \mathbf{d}))$$

$$((\mathbf{a} \vee \mathbf{b}) \leftrightarrow (\mathbf{c} \vee \mathbf{d})) \wedge (\neg(\mathbf{a} \wedge \mathbf{c}) \vee (\mathbf{c} \wedge \neg \mathbf{d}))$$

Example

Given

$$\mathcal{I}: a \mapsto T,\ b \mapsto F,\ c \mapsto F,\ d \mapsto T,$$

Is $((a \vee b) \leftrightarrow (c \vee d)) \wedge (\neg(a \wedge c) \vee (c \wedge \neg d))$ true or false?

$$((\mathbf{a} \vee \mathbf{b}) \leftrightarrow (\mathbf{c} \vee \mathbf{d})) \wedge (\neg(\mathbf{a} \wedge \mathbf{c}) \vee (\mathbf{c} \wedge \neg \mathbf{d}))$$

$$((\mathbf{a} \vee \mathbf{b}) \leftrightarrow (\mathbf{c} \vee \mathbf{d})) \wedge (\neg(\mathbf{a} \wedge \mathbf{c}) \vee (\mathbf{c} \wedge \neg \mathbf{d}))$$

$$((\mathbf{a} \vee \mathbf{b}) \leftrightarrow (\mathbf{c} \vee \mathbf{d})) \wedge (\neg(\mathbf{a} \wedge \mathbf{c}) \vee (\mathbf{c} \wedge \neg \mathbf{d}))$$

$$((\mathbf{a} \vee \mathbf{b}) \leftrightarrow (\mathbf{c} \vee \mathbf{d})) \wedge (\neg(\mathbf{a} \wedge \mathbf{c}) \vee (\mathbf{c} \wedge \neg \mathbf{d}))$$

$$((\mathbf{a} \vee \mathbf{b}) \leftrightarrow (\mathbf{c} \vee \mathbf{d})) \wedge (\neg(\mathbf{a} \wedge \mathbf{c}) \vee (\mathbf{c} \wedge \neg \mathbf{d}))$$

Example

Given

$$\mathcal{I} : a \mapsto T,\ b \mapsto F,\ c \mapsto F,\ d \mapsto T,$$

Is $((a \vee b) \leftrightarrow (c \vee d)) \wedge (\neg(a \wedge c) \vee (c \wedge \neg d))$ true or false?

$$((\mathbf{a} \vee \mathbf{b}) \leftrightarrow (\mathbf{c} \vee \mathbf{d})) \wedge (\neg(\mathbf{a} \wedge \mathbf{c}) \vee (\mathbf{c} \wedge \neg \mathbf{d}))$$

$$((\mathbf{a} \vee \mathbf{b}) \leftrightarrow (\mathbf{c} \vee \mathbf{d})) \wedge (\neg(\mathbf{a} \wedge \mathbf{c}) \vee (\mathbf{c} \wedge \neg \mathbf{d}))$$

$$((\mathbf{a} \vee \mathbf{b}) \leftrightarrow (\mathbf{c} \vee \mathbf{d})) \wedge (\neg(\mathbf{a} \wedge \mathbf{c}) \vee (\mathbf{c} \wedge \neg \mathbf{d}))$$

$$((\mathbf{a} \vee \mathbf{b}) \leftrightarrow (\mathbf{c} \vee \mathbf{d})) \wedge (\neg(\mathbf{a} \wedge \mathbf{c}) \vee (\mathbf{c} \wedge \neg \mathbf{d}))$$

$$((\mathbf{a} \vee \mathbf{b}) \leftrightarrow (\mathbf{c} \vee \mathbf{d})) \wedge (\neg(\mathbf{a} \wedge \mathbf{c}) \vee (\mathbf{c} \wedge \neg \mathbf{d}))$$

Example

Given

$$\mathcal{I} : a \mapsto T,\ b \mapsto F,\ c \mapsto F,\ d \mapsto T,$$

Is $((a \vee b) \leftrightarrow (c \vee d)) \wedge (\neg(a \wedge c) \vee (c \wedge \neg d))$ true or false?

$$((\mathbf{a} \vee \mathbf{b}) \leftrightarrow (\mathbf{c} \vee \mathbf{d})) \wedge (\neg(\mathbf{a} \wedge \mathbf{c}) \vee (\mathbf{c} \wedge \neg \mathbf{d}))$$

$$((\mathbf{a} \vee \mathbf{b}) \leftrightarrow (\mathbf{c} \vee \mathbf{d})) \wedge (\neg(\mathbf{a} \wedge \mathbf{c}) \vee (\mathbf{c} \wedge \neg \mathbf{d}))$$

$$((\mathbf{a} \vee \mathbf{b}) \leftrightarrow (\mathbf{c} \vee \mathbf{d})) \wedge (\neg(\mathbf{a} \wedge \mathbf{c}) \vee (\mathbf{c} \wedge \neg \mathbf{d}))$$

$$((\mathbf{a} \vee \mathbf{b}) \leftrightarrow (\mathbf{c} \vee \mathbf{d})) \wedge (\neg(\mathbf{a} \wedge \mathbf{c}) \vee (\mathbf{c} \wedge \neg \mathbf{d}))$$

$$((\mathbf{a} \vee \mathbf{b}) \leftrightarrow (\mathbf{c} \vee \mathbf{d})) \wedge (\neg(\mathbf{a} \wedge \mathbf{c}) \vee (\mathbf{c} \wedge \neg \mathbf{d}))$$

Terminology

Terminology

An interpretation $\mathcal{I}$ is a model of φ iff

$$\mathcal{I} \models \varphi$$

A formula φ is

- satisfiable if there is an $\mathcal{I}$ such that $\mathcal{I} \models \varphi$;
- unsatisfiable, otherwise; and
- valid if $\mathcal{I} \models \varphi$ for each $\mathcal{I}$ (or tautology);
- falsifiable, otherwise.

Two formulae φ and ψ are logically equivalent (symb. $\varphi \equiv \psi$) if for all interpretations $\mathcal{I}$,

$$\mathcal{I} \models \varphi \text{ iff } \mathcal{I} \models \psi.$$

Terminology

An interpretation $\mathcal{I}$ is a model of φ iff

$$\mathcal{I} \models \varphi$$

A formula φ is

- satisfiable if there is an $\mathcal{I}$ such that $\mathcal{I} \models \varphi$;
- unsatisfiable, otherwise; and
- valid if $\mathcal{I} \models \varphi$ for each $\mathcal{I}$ (or tautology);
- falsifiable, otherwise.

Two formulae φ and ψ are logically equivalent (symb. $\varphi \equiv \psi$) if for all interpretations $\mathcal{I}$,

$$\mathcal{I} \models \varphi \text{ iff } \mathcal{I} \models \psi.$$

Terminology

An interpretation $\mathcal{I}$ is a model of φ iff

$$\mathcal{I} \models \varphi$$

A formula φ is

- satisfiable if there is an $\mathcal{I}$ such that $\mathcal{I} \models \varphi$;
- unsatisfiable, otherwise; and
- valid if $\mathcal{I} \models \varphi$ for each $\mathcal{I}$ (or tautology);
- falsifiable, otherwise.

Two formulae φ and ψ are logically equivalent (symb. $\varphi \equiv \psi$) if for all interpretations $\mathcal{I}$,

$$\mathcal{I} \models \varphi \text{ iff } \mathcal{I} \models \psi.$$

Terminology

An interpretation $\mathcal{I}$ is a model of φ iff

$$\mathcal{I} \models \varphi$$

A formula φ is

- satisfiable if there is an $\mathcal{I}$ such that $\mathcal{I} \models \varphi$;
- unsatisfiable, otherwise; and
- valid if $\mathcal{I} \models \varphi$ for each $\mathcal{I}$ (or tautology);
- falsifiable, otherwise.

Two formulae φ and ψ are logically equivalent (symb. $\varphi \equiv \psi$) if for all interpretations $\mathcal{I}$,

$$\mathcal{I} \models \varphi \text{ iff } \mathcal{I} \models \psi.$$

Examples

Satisfiable, unsatisfiable, falsifiable, valid?

$(a \lor b \lor \neg c) \land (\neg a \lor \neg b \lor d) \land (\neg a \lor b \lor \neg d)$

- $\rightsquigarrow$ satisfiable: $a \mapsto T, b \mapsto F, d \mapsto F, \ldots$
- $\rightsquigarrow$ falsifiable: $a \mapsto F, b \mapsto F, c \mapsto T, \ldots$

$((\neg a \rightarrow \neg b) \rightarrow (b \rightarrow a))$

- $\rightsquigarrow$ satisfiable: $a \mapsto T, b \mapsto T$
- $\rightsquigarrow$ valid: Consider all interpretations or argue about falsifying ones.

Equivalence? $\neg(a \lor b) \equiv \neg a \land \neg b$

- $\rightsquigarrow$ Of course, equivalent (de Morgan).

Examples

Satisfiable, unsatisfiable, falsifiable, valid?
$(a \vee b \vee \neg c) \wedge (\neg a \vee \neg b \vee d) \wedge (\neg a \vee b \vee \neg d)$

- $\rightsquigarrow$ satisfiable: $a \mapsto T, b \mapsto F, d \mapsto F, \ldots$
- $\rightsquigarrow$ falsifiable: $a \mapsto F, b \mapsto F, c \mapsto T, \ldots$

$((\neg a \rightarrow \neg b) \rightarrow (b \rightarrow a))$

- $\rightsquigarrow$ satisfiable: $a \mapsto T, b \mapsto T$
- $\rightsquigarrow$ valid: Consider all interpretations or argue about falsifying ones.

Equivalence? $\neg(a \vee b) \equiv \neg a \wedge \neg b$

- $\rightsquigarrow$ Of course, equivalent (de Morgan).

Examples

Satisfiable, unsatisfiable, falsifiable, valid?

$(a \vee b \vee \neg c) \wedge (\neg a \vee \neg b \vee d) \wedge (\neg a \vee b \vee \neg d)$

- $\rightsquigarrow$ satisfiable: $a \mapsto T, b \mapsto F, d \mapsto F, \ldots$
- $\rightsquigarrow$ falsifiable: $a \mapsto F, b \mapsto F, c \mapsto T, \ldots$

$((\neg a \rightarrow \neg b) \rightarrow (b \rightarrow a))$

- $\rightsquigarrow$ satisfiable: $a \mapsto T, b \mapsto T$
- $\rightsquigarrow$ valid: Consider all interpretations or argue about falsifying ones.

Equivalence? $\neg(a \vee b) \equiv \neg a \wedge \neg b$

- $\rightsquigarrow$ Of course, equivalent (de Morgan).

Some obvious consequences

Proposition

φ is valid iff $\neg\varphi$ is unsatisfiable and φ is satisfiable iff $\neg\varphi$ is falsifiable.

Proposition

$\varphi \equiv \psi$ iff $\varphi \leftrightarrow \psi$ is valid.

Theorem

If $\varphi \equiv \psi$ and χ' results from substituting φ by ψ in χ, then $\chi' \equiv \chi$.

Some obvious consequences

Proposition

φ is valid iff $\neg\varphi$ is unsatisfiable and φ is satisfiable iff $\neg\varphi$ is falsifiable.

Proposition

$\varphi \equiv \psi$ iff $\varphi \leftrightarrow \psi$ is valid.

Theorem

If $\varphi \equiv \psi$ and χ' results from substituting φ by ψ in χ, then $\chi' \equiv \chi$.

Some obvious consequences

Proposition

φ is valid iff $\neg\varphi$ is unsatisfiable and φ is satisfiable iff $\neg\varphi$ is falsifiable.

Proposition

$\varphi \equiv \psi$ iff $\varphi \leftrightarrow \psi$ is valid.

Theorem

If $\varphi \equiv \psi$ and χ' results from substituting φ by ψ in χ, then $\chi' \equiv \chi$.

Some equivalences

simplifications	$\varphi \rightarrow \psi$	$\equiv$	$\neg\varphi \vee \psi$	$\varphi \leftrightarrow \psi$	$\equiv$	$(\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$
idempotency	$\varphi \vee \varphi$	$\equiv$	φ	$\varphi \wedge \varphi$	$\equiv$	φ
commutativity	$\varphi \vee \psi$	$\equiv$	$\psi \vee \varphi$	$\varphi \wedge \psi$	$\equiv$	$\psi \wedge \varphi$
associativity	$(\varphi \vee \psi) \vee \chi$	$\equiv$	$\varphi \vee (\psi \vee \chi)$	$(\varphi \wedge \psi) \wedge \chi$	$\equiv$	$\varphi \wedge (\psi \wedge \chi)$
absorption	$\varphi \vee (\varphi \wedge \psi)$	$\equiv$	φ	$\varphi \wedge (\varphi \vee \psi)$	$\equiv$	φ
distributivity	$\varphi \wedge (\psi \vee \chi)$	$\equiv$	$(\varphi \wedge \psi) \vee (\varphi \wedge \chi)$	$\varphi \vee (\psi \wedge \chi)$	$\equiv$	$(\varphi \vee \psi) \wedge (\varphi \vee \chi)$
double negation	$\neg\neg\varphi$	$\equiv$	φ			
constants	$\neg\top$	$\equiv$	$\bot$	$\neg\bot$	$\equiv$	$\top$
De Morgan	$\neg(\varphi \vee \psi)$	$\equiv$	$\neg\varphi \wedge \neg\psi$	$\neg(\varphi \wedge \psi)$	$\equiv$	$\neg\varphi \vee \neg\psi$
truth	$\varphi \vee \top$	$\equiv$	$\top$	$\varphi \wedge \top$	$\equiv$	φ
falsity	$\varphi \vee \bot$	$\equiv$	φ	$\varphi \wedge \bot$	$\equiv$	$\bot$
taut./contrad.	$\varphi \vee \neg\varphi$	$\equiv$	$\top$	$\varphi \wedge \neg\varphi$	$\equiv$	$\bot$

How many different formulae are there . . .

. . . for a given finite alphabet Σ?

- Infinitely many: $a, a \vee a, a \wedge a, a \vee a \vee a, \ldots$
- How many different logically distinguishable (not equivalent) formulae?
  - For Σ with $n = |\Sigma|$, there are 2^n different interpretations.
  - A formula can be characterized by its set of models (if two formulae are not logically equivalent, then their sets of models differ).
  - There are $2^{(2^n)}$ different sets of interpretations.
  - There are $2^{(2^n)}$ (logical) equivalence classes of formulae.

How many different formulae are there …

…for a given finite alphabet Σ?

- Infinitely many: $a, a \vee a, a \wedge a, a \vee a \vee a, \ldots$
- How many different logically distinguishable (not equivalent) formulae?
  - For Σ with $n = |\Sigma|$, there are 2^n different interpretations.
  - A formula can be characterized by its set of models (if two formulae are not logically equivalent, then their sets of models differ).
  - There are $2^{(2^n)}$ different sets of interpretations.
  - There are $2^{(2^n)}$ (logical) equivalence classes of formulae.

How many different formulae are there …

…for a given finite alphabet Σ?

- Infinitely many: $a, a \vee a, a \wedge a, a \vee a \vee a, \ldots$
- How many different logically distinguishable (not equivalent) formulae?
  - For Σ with $n = |\Sigma|$, there are 2^n different interpretations.
  - A formula can be characterized by its set of models (if two formulae are not logically equivalent, then their sets of models differ).
  - There are $2^{(2^n)}$ different sets of interpretations.
  - There are $2^{(2^n)}$ (logical) equivalence classes of formulae.

How many different formulae are there …

…for a given finite alphabet Σ?

- Infinitely many: $a, a \vee a, a \wedge a, a \vee a \vee a, \ldots$
- How many different logically distinguishable (not equivalent) formulae?
  - For Σ with $n = |\Sigma|$, there are 2^n different interpretations.
  - A formula can be characterized by its set of models (if two formulae are not logically equivalent, then their sets of models differ).
  - There are $2^{(2^n)}$ different sets of interpretations.
  - There are $2^{(2^n)}$ (logical) equivalence classes of formulae.

How many different formulae are there …

…for a given finite alphabet Σ?

- Infinitely many: $a, a \vee a, a \wedge a, a \vee a \vee a, \ldots$
- How many different logically distinguishable (not equivalent) formulae?
  - For Σ with $n = |\Sigma|$, there are 2^n different interpretations.
  - A formula can be characterized by its set of models (if two formulae are not logically equivalent, then their sets of models differ).
  - There are $2^{(2^n)}$ different sets of interpretations.
  - There are $2^{(2^n)}$ (logical) equivalence classes of formulae.

How many different formulae are there …

…for a given finite alphabet Σ?

- Infinitely many: $a, a \vee a, a \wedge a, a \vee a \vee a, \ldots$
- How many different logically distinguishable (not equivalent) formulae?
  - For Σ with $n = |\Sigma|$, there are 2^n different interpretations.
  - A formula can be characterized by its set of models (if two formulae are not logically equivalent, then their sets of models differ).
  - There are $2^{(2^n)}$ different sets of interpretations.
  - There are $2^{(2^n)}$ (logical) equivalence classes of formulae.

How many different formulae are there …

…for a given finite alphabet Σ?

- Infinitely many: $a, a \vee a, a \wedge a, a \vee a \vee a, \ldots$
- How many different logically distinguishable (not equivalent) formulae?
  - For Σ with $n = |\Sigma|$, there are 2^n different interpretations.
  - A formula can be characterized by its set of models (if two formulae are not logically equivalent, then their sets of models differ).
  - There are $2^{(2^n)}$ different sets of interpretations.
  - There are $2^{(2^n)}$ (logical) equivalence classes of formulae.

Logical implication

- Extension of the relation $\models$ to sets Θ of formulae:

$$\mathcal{I} \models \Theta \text{ iff } \mathcal{I} \models \varphi \text{ for all } \varphi \in \Theta.$$

- φ is logically implied by Θ (symbolically $\Theta \models \varphi$) iff φ is true in all models of Θ:

$$\Theta \models \varphi \text{ iff } \mathcal{I} \models \varphi \text{ for all } \mathcal{I} \text{ such that } \mathcal{I} \models \Theta$$

- Some consequences:
  - Deduction theorem: $\Theta \cup \{\varphi\} \models \psi$ iff $\Theta \models \varphi \rightarrow \psi$
  - Contraposition: $\Theta \cup \{\varphi\} \models \neg\psi$ iff $\Theta \cup \{\psi\} \models \neg\varphi$
  - Contradiction: $\Theta \cup \{\varphi\}$ is unsatisfiable iff $\Theta \models \neg\varphi$

Logical implication

- Extension of the relation $\models$ to sets Θ of formulae:

$$\mathcal{I} \models \Theta \text{ iff } \mathcal{I} \models \varphi \text{ for all } \varphi \in \Theta.$$

- φ is logically implied by Θ (symbolically $\Theta \models \varphi$) iff φ is true in all models of Θ:

$$\Theta \models \varphi \text{ iff } \mathcal{I} \models \varphi \text{ for all } \mathcal{I} \text{ such that } \mathcal{I} \models \Theta$$

- Some consequences:
  - Deduction theorem: $\Theta \cup \{\varphi\} \models \psi$ iff $\Theta \models \varphi \rightarrow \psi$
  - Contraposition: $\Theta \cup \{\varphi\} \models \neg\psi$ iff $\Theta \cup \{\psi\} \models \neg\varphi$
  - Contradiction: $\Theta \cup \{\varphi\}$ is unsatisfiable iff $\Theta \models \neg\varphi$

Logical implication

- Extension of the relation $\models$ to sets Θ of formulae:

$$\mathcal{I} \models \Theta \text{ iff } \mathcal{I} \models \varphi \text{ for all } \varphi \in \Theta.$$

- φ is logically implied by Θ (symbolically $\Theta \models \varphi$) iff φ is true in all models of Θ:

$$\Theta \models \varphi \text{ iff } \mathcal{I} \models \varphi \text{ for all } \mathcal{I} \text{ such that } \mathcal{I} \models \Theta$$

- Some consequences:
  - Deduction theorem: $\Theta \cup \{\varphi\} \models \psi$ iff $\Theta \models \varphi \rightarrow \psi$
  - Contraposition: $\Theta \cup \{\varphi\} \models \neg\psi$ iff $\Theta \cup \{\psi\} \models \neg\varphi$
  - Contradiction: $\Theta \cup \{\varphi\}$ is unsatisfiable iff $\Theta \models \neg\varphi$

Logical implication

- Extension of the relation $\models$ to sets Θ of formulae:

$$\mathcal{I} \models \Theta \text{ iff } \mathcal{I} \models \varphi \text{ for all } \varphi \in \Theta.$$

- φ is logically implied by Θ (symbolically $\Theta \models \varphi$) iff φ is true in all models of Θ:

$$\Theta \models \varphi \text{ iff } \mathcal{I} \models \varphi \text{ for all } \mathcal{I} \text{ such that } \mathcal{I} \models \Theta$$

- Some consequences:
  - Deduction theorem: $\Theta \cup \{\varphi\} \models \psi$ iff $\Theta \models \varphi \rightarrow \psi$
  - Contraposition: $\Theta \cup \{\varphi\} \models \neg\psi$ iff $\Theta \cup \{\psi\} \models \neg\varphi$
  - Contradiction: $\Theta \cup \{\varphi\}$ is unsatisfiable iff $\Theta \models \neg\varphi$

Logical implication

- Extension of the relation $\models$ to sets Θ of formulae:

$$\mathcal{I} \models \Theta \text{ iff } \mathcal{I} \models \varphi \text{ for all } \varphi \in \Theta.$$

- φ is logically implied by Θ (symbolically $\Theta \models \varphi$) iff φ is true in all models of Θ:

$$\Theta \models \varphi \text{ iff } \mathcal{I} \models \varphi \text{ for all } \mathcal{I} \text{ such that } \mathcal{I} \models \Theta$$

- Some consequences:
  - Deduction theorem: $\Theta \cup \{\varphi\} \models \psi$ iff $\Theta \models \varphi \to \psi$
  - Contraposition: $\Theta \cup \{\varphi\} \models \neg\psi$ iff $\Theta \cup \{\psi\} \models \neg\varphi$
  - Contradiction: $\Theta \cup \{\varphi\}$ is unsatisfiable iff $\Theta \models \neg\varphi$

Normal forms

Terminology:

- Atomic formulae a, negated atomic formulae $\neg a$, truth $\top$ and falsity $\bot$ are literals.
- A disjunction of literals is a clause.
- If $\neg$ only occurs in front of an atom and there are no $\rightarrow$ and $\leftrightarrow$, the formula is in negation normal form (NNF).
  Example: $(\neg a \vee \neg b) \wedge c$, but not: $\neg(a \wedge b) \wedge c$
- A conjunction of clauses is in conjunctive normal form (CNF).
  Example: $(a \vee b) \wedge (\neg a \vee c)$
- The dual form (disjunction of conjunctions of literals) is in disjunctive normal form (DNF).
  Example: $(a \wedge b) \vee (\neg a \wedge c)$

Normal forms

Terminology:

- Atomic formulae a, negated atomic formulae $\neg a$, truth $\top$ and falsity $\bot$ are literals.
- A disjunction of literals is a clause.
- If $\neg$ only occurs in front of an atom and there are no $\rightarrow$ and $\leftrightarrow$, the formula is in negation normal form (NNF).
  Example: $(\neg a \vee \neg b) \wedge c$, but not: $\neg(a \wedge b) \wedge c$
- A conjunction of clauses is in conjunctive normal form (CNF).
  Example: $(a \vee b) \wedge (\neg a \vee c)$
- The dual form (disjunction of conjunctions of literals) is in disjunctive normal form (DNF).
  Example: $(a \wedge b) \vee (\neg a \wedge c)$

Normal forms

Terminology:

- Atomic formulae a, negated atomic formulae $\neg a$, truth $\top$ and falsity $\bot$ are literals.
- A disjunction of literals is a clause.
- If $\neg$ only occurs in front of an atom and there are no $\rightarrow$ and $\leftrightarrow$, the formula is in negation normal form (NNF).
  Example: $(\neg a \vee \neg b) \wedge c$, but not: $\neg(a \wedge b) \wedge c$
- A conjunction of clauses is in conjunctive normal form (CNF).
  Example: $(a \vee b) \wedge (\neg a \vee c)$
- The dual form (disjunction of conjunctions of literals) is in disjunctive normal form (DNF).
  Example: $(a \wedge b) \vee (\neg a \wedge c)$

Normal forms

Terminology:

- Atomic formulae a, negated atomic formulae $\neg a$, truth $\top$ and falsity $\bot$ are literals.
- A disjunction of literals is a clause.
- If $\neg$ only occurs in front of an atom and there are no $\rightarrow$ and $\leftrightarrow$, the formula is in negation normal form (NNF).
  Example: $(\neg a \vee \neg b) \wedge c$, but not: $\neg(a \wedge b) \wedge c$
- A conjunction of clauses is in conjunctive normal form (CNF).
  Example: $(a \vee b) \wedge (\neg a \vee c)$
- The dual form (disjunction of conjunctions of literals) is in disjunctive normal form (DNF).
  Example: $(a \wedge b) \vee (\neg a \wedge c)$

Negation normal form

Theorem

For each propositional formula there is a logically equivalent formula in NNF.

Proof.

First eliminate $\rightarrow$ and $\leftrightarrow$ by the appropriate equivalences.
Base case: Claim is true for a, $\neg a$, $\top$, $\bot$.
Inductive case: Assume claim is true for all formulae φ (up to a certain number of connectives) and call its NNF $\text{nnf}(\varphi)$.

- $\text{nnf}(\varphi \wedge \psi) = (\text{nnf}(\varphi) \wedge \text{nnf}(\psi))$
- $\text{nnf}(\varphi \vee \psi) = (\text{nnf}(\varphi) \vee \text{nnf}(\psi))$
- $\text{nnf}(\neg(\varphi \wedge \psi)) = (\text{nnf}(\neg\varphi) \vee \text{nnf}(\neg\psi))$
- $\text{nnf}(\neg(\varphi \vee \psi)) = (\text{nnf}(\neg\varphi) \wedge \text{nnf}(\neg\psi))$
- $\text{nnf}(\neg\neg\varphi) = \text{nnf}(\varphi)$

□

Negation normal form

Theorem

For each propositional formula there is a logically equivalent formula in NNF.

Proof.

First eliminate $\rightarrow$ and $\leftrightarrow$ by the appropriate equivalences.

Base case: Claim is true for a, $\neg a$, $\top$, $\bot$.

Inductive case: Assume claim is true for all formulae φ (up to a certain number of connectives) and call its NNF $\text{nnf}(\varphi)$.

- $\text{nnf}(\varphi \wedge \psi) = (\text{nnf}(\varphi) \wedge \text{nnf}(\psi))$
- $\text{nnf}(\varphi \vee \psi) = (\text{nnf}(\varphi) \vee \text{nnf}(\psi))$
- $\text{nnf}(\neg(\varphi \wedge \psi)) = (\text{nnf}(\neg\varphi) \vee \text{nnf}(\neg\psi))$
- $\text{nnf}(\neg(\varphi \vee \psi)) = (\text{nnf}(\neg\varphi) \wedge \text{nnf}(\neg\psi))$
- $\text{nnf}(\neg\neg\varphi) = \text{nnf}(\varphi)$

□

Negation normal form

Theorem

For each propositional formula there is a logically equivalent formula in NNF.

Proof.

First eliminate $\rightarrow$ and $\leftrightarrow$ by the appropriate equivalences.
Base case: Claim is true for a, $\neg a$, $\top$, $\bot$.
Inductive case: Assume claim is true for all formulae φ (up to a certain number of connectives) and call its NNF $\mathrm{nnf}(\varphi)$.

- $\mathrm{nnf}(\varphi \wedge \psi) = (\mathrm{nnf}(\varphi) \wedge \mathrm{nnf}(\psi))$
- $\mathrm{nnf}(\varphi \vee \psi) = (\mathrm{nnf}(\varphi) \vee \mathrm{nnf}(\psi))$
- $\mathrm{nnf}(\neg(\varphi \wedge \psi)) = (\mathrm{nnf}(\neg\varphi) \vee \mathrm{nnf}(\neg\psi))$
- $\mathrm{nnf}(\neg(\varphi \vee \psi)) = (\mathrm{nnf}(\neg\varphi) \wedge \mathrm{nnf}(\neg\psi))$
- $\mathrm{nnf}(\neg\neg\varphi) = \mathrm{nnf}(\varphi)$

□

Negation normal form

Theorem

For each propositional formula there is a logically equivalent formula in NNF.

Proof.

First eliminate $\rightarrow$ and $\leftrightarrow$ by the appropriate equivalences.
Base case: Claim is true for a, $\neg a$, $\top$, $\bot$.
Inductive case: Assume claim is true for all formulae φ (up to a certain number of connectives) and call its NNF $\mathrm{nnf}(\varphi)$.

- $\mathrm{nnf}(\varphi \wedge \psi) = (\mathrm{nnf}(\varphi) \wedge \mathrm{nnf}(\psi))$
- $\mathrm{nnf}(\varphi \vee \psi) = (\mathrm{nnf}(\varphi) \vee \mathrm{nnf}(\psi))$
- $\mathrm{nnf}(\neg(\varphi \wedge \psi)) = (\mathrm{nnf}(\neg\varphi) \vee \mathrm{nnf}(\neg\psi))$
- $\mathrm{nnf}(\neg(\varphi \vee \psi)) = (\mathrm{nnf}(\neg\varphi) \wedge \mathrm{nnf}(\neg\psi))$
- $\mathrm{nnf}(\neg\neg\varphi) = \mathrm{nnf}(\varphi)$

□

Negation normal form

Theorem

For each propositional formula there is a logically equivalent formula in NNF.

Proof.

First eliminate $\rightarrow$ and $\leftrightarrow$ by the appropriate equivalences.
Base case: Claim is true for a, $\neg a$, $\top$, $\bot$.
Inductive case: Assume claim is true for all formulae φ (up to a certain number of connectives) and call its NNF $\text{nnf}(\varphi)$.

- $\text{nnf}(\varphi \wedge \psi) = (\text{nnf}(\varphi) \wedge \text{nnf}(\psi))$
- $\text{nnf}(\varphi \vee \psi) = (\text{nnf}(\varphi) \vee \text{nnf}(\psi))$
- $\text{nnf}(\neg(\varphi \wedge \psi)) = (\text{nnf}(\neg\varphi) \vee \text{nnf}(\neg\psi))$
- $\text{nnf}(\neg(\varphi \vee \psi)) = (\text{nnf}(\neg\varphi) \wedge \text{nnf}(\neg\psi))$
- $\text{nnf}(\neg\neg\varphi) = \text{nnf}(\varphi)$

□

UNI FREIBURG

Negation normal form

Theorem

For each propositional formula there is a logically equivalent formula in NNF.

Proof.

First eliminate $\rightarrow$ and $\leftrightarrow$ by the appropriate equivalences.
Base case: Claim is true for a, $\neg a$, $\top$, $\bot$.
Inductive case: Assume claim is true for all formulae φ (up to a certain number of connectives) and call its NNF $\mathrm{nnf}(\varphi)$.

- $\mathrm{nnf}(\varphi \wedge \psi) = (\mathrm{nnf}(\varphi) \wedge \mathrm{nnf}(\psi))$
- $\mathrm{nnf}(\varphi \vee \psi) = (\mathrm{nnf}(\varphi) \vee \mathrm{nnf}(\psi))$
- $\mathrm{nnf}(\neg(\varphi \wedge \psi)) = (\mathrm{nnf}(\neg\varphi) \vee \mathrm{nnf}(\neg\psi))$
- $\mathrm{nnf}(\neg(\varphi \vee \psi)) = (\mathrm{nnf}(\neg\varphi) \wedge \mathrm{nnf}(\neg\psi))$
- $\mathrm{nnf}(\neg\neg\varphi) = \mathrm{nnf}(\varphi)$

□

Negation normal form

Theorem

For each propositional formula there is a logically equivalent formula in NNF.

Proof.

First eliminate $\rightarrow$ and $\leftrightarrow$ by the appropriate equivalences.
Base case: Claim is true for a, $\neg a$, $\top$, $\bot$.
Inductive case: Assume claim is true for all formulae φ (up to a certain number of connectives) and call its NNF $\text{nnf}(\varphi)$.

- $\text{nnf}(\varphi \wedge \psi) = (\text{nnf}(\varphi) \wedge \text{nnf}(\psi))$
- $\text{nnf}(\varphi \vee \psi) = (\text{nnf}(\varphi) \vee \text{nnf}(\psi))$
- $\text{nnf}(\neg(\varphi \wedge \psi)) = (\text{nnf}(\neg\varphi) \vee \text{nnf}(\neg\psi))$
- $\text{nnf}(\neg(\varphi \vee \psi)) = (\text{nnf}(\neg\varphi) \wedge \text{nnf}(\neg\psi))$
- $\text{nnf}(\neg\neg\varphi) = \text{nnf}(\varphi)$

□

Negation normal form

Theorem

For each propositional formula there is a logically equivalent formula in NNF.

Proof.

First eliminate $\rightarrow$ and $\leftrightarrow$ by the appropriate equivalences.
Base case: Claim is true for a, $\neg a$, $\top$, $\bot$.
Inductive case: Assume claim is true for all formulae φ (up to a certain number of connectives) and call its NNF $\text{nnf}(\varphi)$.

- $\text{nnf}(\varphi \wedge \psi) = (\text{nnf}(\varphi) \wedge \text{nnf}(\psi))$
- $\text{nnf}(\varphi \vee \psi) = (\text{nnf}(\varphi) \vee \text{nnf}(\psi))$
- $\text{nnf}(\neg(\varphi \wedge \psi)) = (\text{nnf}(\neg\varphi) \vee \text{nnf}(\neg\psi))$
- $\text{nnf}(\neg(\varphi \vee \psi)) = (\text{nnf}(\neg\varphi) \wedge \text{nnf}(\neg\psi))$
- $\text{nnf}(\neg\neg\varphi) = \text{nnf}(\varphi)$

□

Conjunctive normal form

Theorem

For each propositional formula there are logically equivalent formulae in CNF and DNF, respectively.

Beweis.

The claim is true for a, $\neg a$, $\top$, $\bot$.

Let us assume it is true for all formulae φ (up to a certain number of connectives) and call its CNF $\mathrm{cnf}(\varphi)$ (and its DNF $\mathrm{dnf}(\varphi)$).

- $\mathrm{cnf}(\neg\varphi) = \mathrm{nnf}(\neg\mathrm{dnf}(\varphi))$ and $\mathrm{cnf}(\varphi \wedge \psi) = \mathrm{cnf}(\varphi) \wedge \mathrm{cnf}(\psi)$.
- Assume $\mathrm{cnf}(\varphi) = \bigwedge_i \chi_i$ and $\mathrm{cnf}(\psi) = \bigwedge_j \rho_j$ with χ_i, ρ_j being clauses. Then
  $\mathrm{cnf}(\varphi \vee \psi) = \mathrm{cnf}((\bigwedge_i \chi_i) \vee (\bigwedge_j \rho_j)) = \bigwedge_i \bigwedge_j (\chi_i \vee \rho_j)$ (by distributivity)

□

Conjunctive normal form

UNI FREIBURG

Theorem

For each propositional formula there are logically equivalent formulae in CNF and DNF, respectively.

Beweis.

The claim is true for a, $\neg a$, $\top$, $\bot$.

Let us assume it is true for all formulae φ (up to a certain number of connectives) and call its CNF $\mathrm{cnf}(\varphi)$ (and its DNF $\mathrm{dnf}(\varphi)$).

- $\mathrm{cnf}(\neg\varphi) = \mathrm{nnf}(\neg\mathrm{dnf}(\varphi))$ and $\mathrm{cnf}(\varphi \wedge \psi) = \mathrm{cnf}(\varphi) \wedge \mathrm{cnf}(\psi)$.
- Assume $\mathrm{cnf}(\varphi) = \bigwedge_i \chi_i$ and $\mathrm{cnf}(\psi) = \bigwedge_j \rho_j$ with χ_i, ρ_j being clauses. Then
  $\mathrm{cnf}(\varphi \vee \psi) = \mathrm{cnf}((\bigwedge_i \chi_i) \vee (\bigwedge_j \rho_j)) = \bigwedge_i \bigwedge_j (\chi_i \vee \rho_j)$ (by distributivity)

□

Conjunctive normal form

Theorem

For each propositional formula there are logically equivalent formulae in CNF and DNF, respectively.

Beweis.

The claim is true for a, $\neg a$, $\top$, $\bot$.

Let us assume it is true for all formulae φ (up to a certain number of connectives) and call its CNF $\text{cnf}(\varphi)$ (and its DNF $\text{dnf}(\varphi)$).

- $\text{cnf}(\neg\varphi) = \text{nnf}(\neg\text{dnf}(\varphi))$ and $\text{cnf}(\varphi \wedge \psi) = \text{cnf}(\varphi) \wedge \text{cnf}(\psi)$.
- Assume $\text{cnf}(\varphi) = \bigwedge_i \chi_i$ and $\text{cnf}(\psi) = \bigwedge_j \rho_j$ with χ_i, ρ_j being clauses. Then
  $\text{cnf}(\varphi \vee \psi) = \text{cnf}((\bigwedge_i \chi_i) \vee (\bigwedge_j \rho_j)) = \bigwedge_i \bigwedge_j (\chi_i \vee \rho_j)$ (by distributivity)

□

Conjunctive normal form

UNI FREIBURG

Theorem

For each propositional formula there are logically equivalent formulae in CNF and DNF, respectively.

Beweis.

The claim is true for a, $\neg a$, $\top$, $\bot$.

Let us assume it is true for all formulae φ (up to a certain number of connectives) and call its CNF $\text{cnf}(\varphi)$ (and its DNF $\text{dnf}(\varphi)$).

- $\text{cnf}(\neg\varphi) = \text{nnf}(\neg\text{dnf}(\varphi))$ and $\text{cnf}(\varphi \wedge \psi) = \text{cnf}(\varphi) \wedge \text{cnf}(\psi)$.
- Assume $\text{cnf}(\varphi) = \bigwedge_i \chi_i$ and $\text{cnf}(\psi) = \bigwedge_j \rho_j$ with χ_i, ρ_j being clauses. Then
  $\text{cnf}(\varphi \vee \psi) = \text{cnf}((\bigwedge_i \chi_i) \vee (\bigwedge_j \rho_j)) = \bigwedge_i \bigwedge_j (\chi_i \vee \rho_j)$ (by distributivity)

□

Conjunctive normal form

Theorem

For each propositional formula there are logically equivalent formulae in CNF and DNF, respectively.

Beweis.

The claim is true for a, $\neg a$, $\top$, $\bot$.

Let us assume it is true for all formulae φ (up to a certain number of connectives) and call its CNF $\text{cnf}(\varphi)$ (and its DNF $\text{dnf}(\varphi)$).

- $\text{cnf}(\neg\varphi) = \text{nnf}(\neg\text{dnf}(\varphi))$ and $\text{cnf}(\varphi \wedge \psi) = \text{cnf}(\varphi) \wedge \text{cnf}(\psi)$.
- Assume $\text{cnf}(\varphi) = \bigwedge_i \chi_i$ and $\text{cnf}(\psi) = \bigwedge_j \rho_j$ with χ_i, ρ_j being clauses. Then
  $\text{cnf}(\varphi \vee \psi) = \text{cnf}((\bigwedge_i \chi_i) \vee (\bigwedge_j \rho_j)) = \bigwedge_i \bigwedge_j (\chi_i \vee \rho_j)$ (by distributivity)

□

Conjunctive normal form

Theorem

For each propositional formula there are logically equivalent formulae in CNF and DNF, respectively.

Beweis.

The claim is true for a, $\neg a$, $\top$, $\bot$.

Let us assume it is true for all formulae φ (up to a certain number of connectives) and call its CNF $\text{cnf}(\varphi)$ (and its DNF $\text{dnf}(\varphi)$).

- $\text{cnf}(\neg\varphi) = \text{nnf}(\neg\text{dnf}(\varphi))$ and $\text{cnf}(\varphi \wedge \psi) = \text{cnf}(\varphi) \wedge \text{cnf}(\psi)$.
- Assume $\text{cnf}(\varphi) = \bigwedge_i \chi_i$ and $\text{cnf}(\psi) = \bigwedge_j \rho_j$ with χ_i, ρ_j being clauses. Then
  $\text{cnf}(\varphi \vee \psi) = \text{cnf}((\bigwedge_i \chi_i) \vee (\bigwedge_j \rho_j)) = \bigwedge_i \bigwedge_j (\chi_i \vee \rho_j)$ (by distributivity)

□

Decision Problems and Resolution

How to decide properties of formulae

How do we decide whether a formula is satisfiable, unsatisfiable, valid, or falsifiable?

Note: Satisfiability and falsifiability are NP-complete. Validity and unsatisfiability are co-NP-complete.

- A CNF formula is valid iff all clauses contain two complementary literals or $\top$.
- A DNF formula is satisfiable iff one disjunct does not contain $\bot$ or two complementary literals.
- However, transformation to CNF or DNF may take exponential time (and space!).
- One can try out all truth assignments.
- One can test systematically for satisfying truth assignments (backtracking) $\rightsquigarrow$ Davis-Putnam-Logemann-Loveland.

How to decide properties of formulae

How do we decide whether a formula is satisfiable, unsatisfiable, valid, or falsifiable?

Note: Satisfiability and falsifiability are NP-complete. Validity and unsatisfiability are co-NP-complete.

- A CNF formula is valid iff all clauses contain two complementary literals or $\top$.
- A DNF formula is satisfiable iff one disjunct does not contain $\bot$ or two complementary literals.
- However, transformation to CNF or DNF may take exponential time (and space!).
- One can try out all truth assignments.
- One can test systematically for satisfying truth assignments (backtracking) $\rightsquigarrow$ Davis-Putnam-Logemann-Loveland.

How to decide properties of formulae

How do we decide whether a formula is satisfiable, unsatisfiable, valid, or falsifiable?

Note: Satisfiability and falsifiability are NP-complete. Validity and unsatisfiability are co-NP-complete.

- A CNF formula is valid iff all clauses contain two complementary literals or $\top$.
- A DNF formula is satisfiable iff one disjunct does not contain $\bot$ or two complementary literals.
- However, transformation to CNF or DNF may take exponential time (and space!).
- One can try out all truth assignments.
- One can test systematically for satisfying truth assignments (backtracking) $\rightsquigarrow$ Davis-Putnam-Logemann-Loveland.

How to decide properties of formulae

How do we decide whether a formula is satisfiable, unsatisfiable, valid, or falsifiable?

Note: Satisfiability and falsifiability are NP-complete. Validity and unsatisfiability are co-NP-complete.

- A CNF formula is valid iff all clauses contain two complementary literals or $\top$.
- A DNF formula is satisfiable iff one disjunct does not contain $\perp$ or two complementary literals.
- However, transformation to CNF or DNF may take exponential time (and space!).
- One can try out all truth assignments.
- One can test systematically for satisfying truth assignments (backtracking) $\rightsquigarrow$ Davis-Putnam-Logemann-Loveland.

How to decide properties of formulae

How do we decide whether a formula is satisfiable, unsatisfiable, valid, or falsifiable?

Note: Satisfiability and falsifiability are NP-complete. Validity and unsatisfiability are co-NP-complete.

- A CNF formula is valid iff all clauses contain two complementary literals or $\top$.
- A DNF formula is satisfiable iff one disjunct does not contain $\bot$ or two complementary literals.
- However, transformation to CNF or DNF may take exponential time (and space!).
- One can try out all truth assignments.
- One can test systematically for satisfying truth assignments (backtracking) $\rightsquigarrow$ Davis-Putnam-Logemann-Loveland.

How to decide properties of formulae

How do we decide whether a formula is satisfiable, unsatisfiable, valid, or falsifiable?

Note: Satisfiability and falsifiability are NP-complete. Validity and unsatisfiability are co-NP-complete.

- A CNF formula is valid iff all clauses contain two complementary literals or $\top$.
- A DNF formula is satisfiable iff one disjunct does not contain $\bot$ or two complementary literals.
- However, transformation to CNF or DNF may take exponential time (and space!).
- One can try out all truth assignments.
- One can test systematically for satisfying truth assignments (backtracking) $\rightsquigarrow$ Davis-Putnam-Logemann-Loveland.

Deciding entailment

- We want to decide $\Theta \models \varphi$.
- Use deduction theorem and reduce to validity:

$$\Theta \models \varphi \text{ iff } \bigwedge \Theta \rightarrow \varphi \text{ is valid.}$$

- Now negate and test for unsatisfiability using DPLL.
- Different approach: Try to derive φ from Θ – find a proof of φ from Θ.
- Use inference rules to derive new formulae from Θ. Continue to deduce new formulae until φ can be deduced.
- One particular calculus: resolution.

Deciding entailment

- We want to decide $\Theta \models \varphi$.
- Use deduction theorem and reduce to validity:

$$\Theta \models \varphi \text{ iff } \bigwedge \Theta \rightarrow \varphi \text{ is valid.}$$

- Now negate and test for unsatisfiability using DPLL.
- Different approach: Try to derive φ from Θ – find a proof of φ from Θ.
- Use inference rules to derive new formulae from Θ. Continue to deduce new formulae until φ can be deduced.
- One particular calculus: resolution.

Deciding entailment

- We want to decide $\Theta \models \varphi$.
- Use deduction theorem and reduce to validity:

$$\Theta \models \varphi \text{ iff } \bigwedge \Theta \rightarrow \varphi \text{ is valid.}$$

- Now negate and test for unsatisfiability using DPLL.
- Different approach: Try to derive φ from Θ – find a proof of φ from Θ.
- Use inference rules to derive new formulae from Θ. Continue to deduce new formulae until φ can be deduced.
- One particular calculus: resolution.

Deciding entailment

- We want to decide $\Theta \models \varphi$.
- Use deduction theorem and reduce to validity:

$$\Theta \models \varphi \text{ iff } \bigwedge \Theta \rightarrow \varphi \text{ is valid.}$$

- Now negate and test for unsatisfiability using DPLL.
- Different approach: Try to derive φ from Θ – find a proof of φ from Θ.
- Use inference rules to derive new formulae from Θ. Continue to deduce new formulae until φ can be deduced.
- One particular calculus: resolution.

Deciding entailment

- We want to decide $\Theta \models \varphi$.
- Use deduction theorem and reduce to validity:

$$\Theta \models \varphi \text{ iff } \bigwedge \Theta \rightarrow \varphi \text{ is valid.}$$

- Now negate and test for unsatisfiability using DPLL.
- Different approach: Try to derive φ from Θ – find a proof of φ from Θ.
- Use inference rules to derive new formulae from Θ. Continue to deduce new formulae until φ can be deduced.
- One particular calculus: resolution.

Resolution: representation

- We assume that all formulae are in CNF.
  - Can be generated using the described method.
  - Often formulae are already close to CNF.
  - There is a “cheap” conversion from arbitrary formulae to CNF that preserves satisfiability – which is enough as we will see.
- More convenient representation:
  - CNF formula is represented as a set.
  - Each clause is a set of literals.
  - $(a \vee \neg b) \wedge (\neg a \vee c) \rightsquigarrow \{\{a, \neg b\}, \{\neg a, c\}\}$
- Empty clause (symbolically $\square$) and empty set of clauses (symbolically $\emptyset$) are different!

Resolution: representation

- We assume that all formulae are in CNF.
  - Can be generated using the described method.
  - Often formulae are already close to CNF.
  - There is a "cheap" conversion from arbitrary formulae to CNF that preserves satisfiability – which is enough as we will see.
- More convenient representation:
  - CNF formula is represented as a set.
  - Each clause is a set of literals.
  - $(a \vee \neg b) \wedge (\neg a \vee c) \rightsquigarrow \{\{a, \neg b\}, \{\neg a, c\}\}$
- Empty clause (symbolically $\square$) and empty set of clauses (symbolically $\emptyset$) are different!

Resolution: representation

- We assume that all formulae are in CNF.
  - Can be generated using the described method.
  - Often formulae are already close to CNF.
  - There is a “cheap” conversion from arbitrary formulae to CNF that preserves satisfiability – which is enough as we will see.
- More convenient representation:
  - CNF formula is represented as a set.
  - Each clause is a set of literals.
  - $(a \vee \neg b) \wedge (\neg a \vee c) \rightsquigarrow \{\{a, \neg b\}, \{\neg a, c\}\}$
- Empty clause (symbolically $\square$) and empty set of clauses (symbolically $\emptyset$) are different!

Resolution: the inference rule

Let l be a literal and $\bar{l}$ its complement.

The resolution rule

$$\frac{C_1 \dot{\cup} \{l\}, C_2 \dot{\cup} \{\bar{l}\}}{C_1 \cup C_2}$$

$C_1 \cup C_2$ is the resolvent of the parent clauses $C_1 \cup \{l\}$ and $C_2 \cup \{\bar{l}\}$. l and $\bar{l}$ are the resolution literals.

Example: $\{a, b, \neg c\}$ resolves with $\{a, d, c\}$ to $\{a, b, d\}$.

Note: The resolvent is not logically equivalent to the set of parent clauses!

Notation:

$$R(\Delta) = \{C \mid C \text{ is resolvent of two clauses in } \Delta\}$$

Resolution: the inference rule

Let l be a literal and $\bar{l}$ its complement.

The resolution rule

$$\frac{C_1 \,\dot{\cup}\, \{l\}, C_2 \,\dot{\cup}\, \{\bar{l}\}}{C_1 \cup C_2}$$

$C_1 \cup C_2$ is the resolvent of the parent clauses $C_1 \cup \{l\}$ and $C_2 \cup \{\bar{l}\}$. l and $\bar{l}$ are the resolution literals.

Example: $\{a, b, \neg c\}$ resolves with $\{a, d, c\}$ to $\{a, b, d\}$.

Note: The resolvent is not logically equivalent to the set of parent clauses!

Notation:

$$R(\Delta) = \{C \mid C \text{ is resolvent of two clauses in } \Delta\}$$

Resolution: the inference rule

Let l be a literal and $\bar{l}$ its complement.

The resolution rule

$$\frac{C_1 \dot\cup \{l\}, C_2 \dot\cup \{\bar{l}\}}{C_1 \cup C_2}$$

$C_1 \cup C_2$ is the resolvent of the parent clauses $C_1 \cup \{l\}$ and $C_2 \cup \{\bar{l}\}$. l and $\bar{l}$ are the resolution literals.

Example: $\{a, b, \neg c\}$ resolves with $\{a, d, c\}$ to $\{a, b, d\}$.

Note: The resolvent is not logically equivalent to the set of parent clauses!

Notation:

$$R(\Delta) = \{C \mid C \text{ is resolvent of two clauses in } \Delta\}$$

Resolution: the inference rule

Let l be a literal and $\bar{l}$ its complement.

The resolution rule

$$\frac{C_1 \dot\cup \{l\}, C_2 \dot\cup \{\bar{l}\}}{C_1 \cup C_2}$$

$C_1 \cup C_2$ is the resolvent of the parent clauses $C_1 \cup \{l\}$ and $C_2 \cup \{\bar{l}\}$. l and $\bar{l}$ are the resolution literals.

Example: $\{a, b, \neg c\}$ resolves with $\{a, d, c\}$ to $\{a, b, d\}$.

Note: The resolvent is not logically equivalent to the set of parent clauses!

Notation:

$$R(\Delta) = \{C \mid C \text{ is resolvent of two clauses in } \Delta\}$$

Resolution: the inference rule

Let l be a literal and $\bar{l}$ its complement.

The resolution rule

$$\frac{C_1 \dot{\cup} \{l\}, C_2 \dot{\cup} \{\bar{l}\}}{C_1 \cup C_2}$$

$C_1 \cup C_2$ is the resolvent of the parent clauses $C_1 \cup \{l\}$ and $C_2 \cup \{\bar{l}\}$. l and $\bar{l}$ are the resolution literals.

Example: $\{a, b, \neg c\}$ resolves with $\{a, d, c\}$ to $\{a, b, d\}$.

Note: The resolvent is not logically equivalent to the set of parent clauses!

Notation:

$$R(\Delta) \quad = \quad \{C \mid C \text{ is resolvent of two clauses in } \Delta\}$$

Resolution: derivations

D can be derived from Δ by resolution (symbolically $\Delta \vdash D$) if there is a sequence $C_1, \dots, C_n$ of clauses such that

1. $C_n = D$ and
   $C_i \in R(\Delta \cup \{C_1, \dots, C_{i-1}\})$, for all $i \in \{1, \dots, n\}$.

Define $R^*(\Delta) = \{D \mid \Delta \vdash D\}$.

Theorem (Soundness of resolution)

Let D be a clause. If $\Delta \vdash D$ then $\Delta \models D$.

Proof idea.

Show $\Delta \models D$ if $D \in R(\Delta)$ and use induction on proof length.
Let $C_1 \cup \{l\}$ and $C_2 \cup \{\bar{l}\}$ be the parent clauses of $D = C_1 \cup C_2$.
Assume $\mathcal{I} \models \Delta$, we have to show $\mathcal{I} \models D$.
Case 1: $\mathcal{I} \models l$ then $\exists m \in C_2$ s.t. $\mathcal{I} \models m$. This implies $\mathcal{I} \models D$.
Case 2: $\mathcal{I} \models \bar{l}$ similarly, $\exists m \in C_1$ s.t. $\mathcal{I} \models m$.

This means that each model $\mathcal{I}$ of Δ also satisfies D, i.e., $\Delta \models D$. □

Resolution: derivations

D can be derived from Δ by resolution (symbolically $\Delta \vdash D$) if there is a sequence $C_1, \dots, C_n$ of clauses such that

1. $C_n = D$ and
   $C_i \in R(\Delta \cup \{C_1, \dots, C_{i-1}\})$, for all $i \in \{1, \dots, n\}$.

Define $R^*(\Delta) = \{D \mid \Delta \vdash D\}$.

Theorem (Soundness of resolution)

Let D be a clause. If $\Delta \vdash D$ then $\Delta \models D$.

Proof idea.

Show $\Delta \models D$ if $D \in R(\Delta)$ and use induction on proof length.
Let $C_1 \cup \{l\}$ and $C_2 \cup \{\bar{l}\}$ be the parent clauses of $D = C_1 \cup C_2$.
Assume $\mathcal{I} \models \Delta$, we have to show $\mathcal{I} \models D$.
Case 1: $\mathcal{I} \models l$ then $\exists m \in C_2$ s.t. $\mathcal{I} \models m$. This implies $\mathcal{I} \models D$.
Case 2: $\mathcal{I} \models \bar{l}$ similarly, $\exists m \in C_1$ s.t. $\mathcal{I} \models m$.

This means that each model $\mathcal{I}$ of Δ also satisfies D, i.e., $\Delta \models D$. □

Resolution: derivations

D can be derived from Δ by resolution (symbolically $\Delta \vdash D$) if there is a sequence $C_1, \ldots, C_n$ of clauses such that

1. $C_n = D$ and
   $C_i \in R(\Delta \cup \{C_1, \ldots, C_{i-1}\})$, for all $i \in \{1, \ldots, n\}$.

Define $R^*(\Delta) = \{D \mid \Delta \vdash D\}$.

Theorem (Soundness of resolution)

Let D be a clause. If $\Delta \vdash D$ then $\Delta \models D$.

Proof idea.

Show $\Delta \models D$ if $D \in R(\Delta)$ and use induction on proof length.
Let $C_1 \cup \{l\}$ and $C_2 \cup \{\bar{l}\}$ be the parent clauses of $D = C_1 \cup C_2$.
Assume $\mathcal{I} \models \Delta$, we have to show $\mathcal{I} \models D$.
Case 1: $\mathcal{I} \models l$ then $\exists m \in C_2$ s.t. $\mathcal{I} \models m$. This implies $\mathcal{I} \models D$.
Case 2: $\mathcal{I} \models \bar{l}$ similarly, $\exists m \in C_1$ s.t. $\mathcal{I} \models m$.
This means that each model $\mathcal{I}$ of Δ also satisfies D, i.e., $\Delta \models D$. □

Resolution: derivations

D can be derived from Δ by resolution (symbolically $\Delta \vdash D$) if there is a sequence $C_1, \dots, C_n$ of clauses such that

1. $C_n = D$ and
   $C_i \in R(\Delta \cup \{C_1, \dots, C_{i-1}\})$, for all $i \in \{1, \dots, n\}$.

Define $R^*(\Delta) = \{D \mid \Delta \vdash D\}$.

Theorem (Soundness of resolution)

Let D be a clause. If $\Delta \vdash D$ then $\Delta \models D$.

Proof idea.

Show $\Delta \models D$ if $D \in R(\Delta)$ and use induction on proof length.
Let $C_1 \cup \{l\}$ and $C_2 \cup \{\bar{l}\}$ be the parent clauses of $D = C_1 \cup C_2$.
Assume $\mathcal{I} \models \Delta$, we have to show $\mathcal{I} \models D$.
Case 1: $\mathcal{I} \models l$ then $\exists m \in C_2$ s.t. $\mathcal{I} \models m$. This implies $\mathcal{I} \models D$.
Case 2: $\mathcal{I} \models \bar{l}$ similarly, $\exists m \in C_1$ s.t. $\mathcal{I} \models m$.
This means that each model $\mathcal{I}$ of Δ also satisfies D, i.e., $\Delta \models D$. □

UNI FREIBURG

Resolution: derivations

D can be derived from Δ by resolution (symbolically $\Delta \vdash D$) if there is a sequence $C_1, \dots, C_n$ of clauses such that

1. $C_n = D$ and
   $C_i \in R(\Delta \cup \{C_1, \dots, C_{i-1}\})$, for all $i \in \{1, \dots, n\}$.

Define $R^*(\Delta) = \{D \mid \Delta \vdash D\}$.

Theorem (Soundness of resolution)

Let D be a clause. If $\Delta \vdash D$ then $\Delta \models D$.

Proof idea.

Show $\Delta \models D$ if $D \in R(\Delta)$ and use induction on proof length.
Let $C_1 \cup \{l\}$ and $C_2 \cup \{\bar{l}\}$ be the parent clauses of $D = C_1 \cup C_2$.
Assume $\mathcal{I} \models \Delta$, we have to show $\mathcal{I} \models D$.
Case 1: $\mathcal{I} \models l$ then $\exists m \in C_2$ s.t. $\mathcal{I} \models m$. This implies $\mathcal{I} \models D$.
Case 2: $\mathcal{I} \models \bar{l}$ similarly, $\exists m \in C_1$ s.t. $\mathcal{I} \models m$.

This means that each model $\mathcal{I}$ of Δ also satisfies D, i.e., $\Delta \models D$. □

UNI FREIBURG

Resolution: derivations

D can be derived from Δ by resolution (symbolically $\Delta \vdash D$) if there is a sequence $C_1, \ldots, C_n$ of clauses such that

1. $C_n = D$ and
   $C_i \in R(\Delta \cup \{C_1, \ldots, C_{i-1}\})$, for all $i \in \{1, \ldots, n\}$.

Define $R^*(\Delta) = \{D \mid \Delta \vdash D\}$.

Theorem (Soundness of resolution)

Let D be a clause. If $\Delta \vdash D$ then $\Delta \models D$.

Proof idea.

Show $\Delta \models D$ if $D \in R(\Delta)$ and use induction on proof length.
Let $C_1 \cup \{l\}$ and $C_2 \cup \{\bar{l}\}$ be the parent clauses of $D = C_1 \cup C_2$.
Assume $\mathcal{I} \models \Delta$, we have to show $\mathcal{I} \models D$.
Case 1: $\mathcal{I} \models l$ then $\exists m \in C_2$ s.t. $\mathcal{I} \models m$. This implies $\mathcal{I} \models D$.
Case 2: $\mathcal{I} \models \bar{l}$ similarly, $\exists m \in C_1$ s.t. $\mathcal{I} \models m$.
This means that each model $\mathcal{I}$ of Δ also satisfies D, i.e., $\Delta \models D$. □

Resolution: derivations

D can be derived from Δ by resolution (symbolically $\Delta \vdash D$) if there is a sequence $C_1, \dots, C_n$ of clauses such that

1. $C_n = D$ and
   $C_i \in R(\Delta \cup \{C_1, \dots, C_{i-1}\})$, for all $i \in \{1, \dots, n\}$.

Define $R^*(\Delta) = \{D \mid \Delta \vdash D\}$.

Theorem (Soundness of resolution)

Let D be a clause. If $\Delta \vdash D$ then $\Delta \models D$.

Proof idea.

Show $\Delta \models D$ if $D \in R(\Delta)$ and use induction on proof length.
Let $C_1 \cup \{l\}$ and $C_2 \cup \{\bar{l}\}$ be the parent clauses of $D = C_1 \cup C_2$.
Assume $\mathcal{I} \models \Delta$, we have to show $\mathcal{I} \models D$.
Case 1: $\mathcal{I} \models l$ then $\exists m \in C_2$ s.t. $\mathcal{I} \models m$. This implies $\mathcal{I} \models D$.
Case 2: $\mathcal{I} \models \bar{l}$ similarly, $\exists m \in C_1$ s.t. $\mathcal{I} \models m$.

This means that each model $\mathcal{I}$ of Δ also satisfies D, i.e., $\Delta \models D$. □

Resolution: derivations

D can be derived from Δ by resolution (symbolically $\Delta \vdash D$) if there is a sequence $C_1, \ldots, C_n$ of clauses such that

1. $C_n = D$ and
   $C_i \in R(\Delta \cup \{C_1, \ldots, C_{i-1}\})$, for all $i \in \{1, \ldots, n\}$.

Define $R^*(\Delta) = \{D \mid \Delta \vdash D\}$.

Theorem (Soundness of resolution)

Let D be a clause. If $\Delta \vdash D$ then $\Delta \models D$.

Proof idea.

Show $\Delta \models D$ if $D \in R(\Delta)$ and use induction on proof length.
Let $C_1 \cup \{l\}$ and $C_2 \cup \{\bar{l}\}$ be the parent clauses of $D = C_1 \cup C_2$.
Assume $\mathcal{I} \models \Delta$, we have to show $\mathcal{I} \models D$.
Case 1: $\mathcal{I} \models l$ then $\exists m \in C_2$ s.t. $\mathcal{I} \models m$. This implies $\mathcal{I} \models D$.
Case 2: $\mathcal{I} \models \bar{l}$ similarly, $\exists m \in C_1$ s.t. $\mathcal{I} \models m$.
This means that each model $\mathcal{I}$ of Δ also satisfies D, i.e., $\Delta \models D$. □

Resolution: derivations

D can be derived from Δ by resolution (symbolically $\Delta \vdash D$) if there is a sequence $C_1, \ldots, C_n$ of clauses such that

1. $C_n = D$ and
   $C_i \in R(\Delta \cup \{C_1, \ldots, C_{i-1}\})$, for all $i \in \{1, \ldots, n\}$.

Define $R^*(\Delta) = \{D \mid \Delta \vdash D\}$.

Theorem (Soundness of resolution)

Let D be a clause. If $\Delta \vdash D$ then $\Delta \models D$.

Proof idea.

Show $\Delta \models D$ if $D \in R(\Delta)$ and use induction on proof length.
Let $C_1 \cup \{l\}$ and $C_2 \cup \{\bar{l}\}$ be the parent clauses of $D = C_1 \cup C_2$.
Assume $\mathcal{I} \models \Delta$, we have to show $\mathcal{I} \models D$.
Case 1: $\mathcal{I} \models l$ then $\exists m \in C_2$ s.t. $\mathcal{I} \models m$. This implies $\mathcal{I} \models D$.
Case 2: $\mathcal{I} \models \bar{l}$ similarly, $\exists m \in C_1$ s.t. $\mathcal{I} \models m$.

This means that each model $\mathcal{I}$ of Δ also satisfies D, i.e., $\Delta \models D$. □

Resolution: completeness?

Do we have

$$\Delta \models \varphi \text{ implies } \Delta \vdash \varphi?$$

Of course, could only hold for CNF. However:

$$\left\{ \{a,b\}, \{\neg b, c\} \right\} \begin{array}{l} \models \{a,b,c\} \\ \not\vdash \{a,b,c\} \end{array}$$

However, one can show that resolution is refutation-complete:

$$\Delta \text{ is unsatisfiable iff } \Delta \vdash \square.$$

Entailment: Reduce to unsatisfiability testing and decide by resolution.

Resolution: completeness?

Do we have

$$\Delta \models \varphi \text{ implies } \Delta \vdash \varphi?$$

Of course, could only hold for CNF. However:

$$\left\{ \{a,b\}, \{\neg b, c\} \right\} \begin{array}{l} \models \{a,b,c\} \\ \not\vdash \{a,b,c\} \end{array}$$

However, one can show that resolution is refutation-complete:

$$\Delta \text{ is unsatisfiable iff } \Delta \vdash \square.$$

Entailment: Reduce to unsatisfiability testing and decide by resolution.

Resolution: completeness?

Do we have

$$\Delta \models \varphi \text{ implies } \Delta \vdash \varphi?$$

Of course, could only hold for CNF. However:

$$\left\{ \{a,b\}, \{\neg b, c\} \right\} \begin{array}{ll} \models & \{a,b,c\} \\ \not\vdash & \{a,b,c\} \end{array}$$

However, one can show that resolution is refutation-complete:

$$\Delta \text{ is unsatisfiable iff } \Delta \vdash \square.$$

Entailment: Reduce to unsatisfiability testing and decide by resolution.

Resolution: completeness?

Do we have

$$\Delta \models \varphi \text{ implies } \Delta \vdash \varphi?$$

Of course, could only hold for CNF. However:

$$\left\{ \{a,b\}, \{\neg b, c\} \right\} \begin{array}{ll} \models & \{a,b,c\} \\ \not\vdash & \{a,b,c\} \end{array}$$

However, one can show that resolution is refutation-complete:

$$\Delta \text{ is unsatisfiable iff } \Delta \vdash \square.$$

Entailment: Reduce to unsatisfiability testing and decide by resolution.

Resolution: completeness?

Do we have

$$\Delta \models \varphi \text{ implies } \Delta \vdash \varphi?$$

Of course, could only hold for CNF. However:

$$\left\{ \{a,b\}, \{\neg b, c\} \right\} \begin{array}{ll} \models & \{a,b,c\} \\ \nvdash & \{a,b,c\} \end{array}$$

However, one can show that resolution is refutation-complete:

$$\Delta \text{ is unsatisfiable iff } \Delta \vdash \square.$$

Entailment: Reduce to unsatisfiability testing and decide by resolution.

Resolution strategies

- Trying out all different resolutions can be very costly,
- and might not be necessary.
- There are different resolution strategies.
- Examples:
  - Input resolution ($R_I(\cdot)$): In each resolution step, one of the parent clauses must be a clause of the input set.
  - Unit resolution ($R_U(\cdot)$): In each resolution step, one of the parent clauses must be a unit clause.
  - Not all strategies are (refutation) completeness preserving. Neither input nor unit resolution is. However, there are others.

Resolution strategies

- Trying out all different resolutions can be very costly,
- and might not be necessary.
- There are different resolution strategies.
- Examples:
  - Input resolution ($R_I(\cdot)$): In each resolution step, one of the parent clauses must be a clause of the input set.
  - Unit resolution ($R_U(\cdot)$): In each resolution step, one of the parent clauses must be a unit clause.
  - Not all strategies are (refutation) completeness preserving. Neither input nor unit resolution is. However, there are others.

Resolution strategies

- Trying out all different resolutions can be very costly,
- and might not be necessary.
- There are different resolution strategies.
- Examples:
  - Input resolution ($R_I(\cdot)$): In each resolution step, one of the parent clauses must be a clause of the input set.
  - Unit resolution ($R_U(\cdot)$): In each resolution step, one of the parent clauses must be a unit clause.
  - Not all strategies are (refutation) completeness preserving. Neither input nor unit resolution is. However, there are others.

Resolution strategies

- Trying out all different resolutions can be very costly,
- and might not be necessary.
- There are different resolution strategies.
- Examples:
  - Input resolution ($R_I(\cdot)$): In each resolution step, one of the parent clauses must be a clause of the input set.
  - Unit resolution ($R_U(\cdot)$): In each resolution step, one of the parent clauses must be a unit clause.
  - Not all strategies are (refutation) completeness preserving. Neither input nor unit resolution is. However, there are others.

Horn clauses & resolution

Horn clauses: Clauses with at most one positive literal

Example: $(a \vee \neg b \vee \neg c), (\neg b \vee \neg c)$

Proposition

Unit resolution is refutation-complete for Horn clauses.

Proof idea.

Consider $R^*_U(\Delta)$ of Horn clause set Δ. We have to show that if $\square \notin R^*_U(\Delta)$, then $\Delta (\equiv R^*_U(\Delta))$ is satisfiable.

- Assign true to all unit clauses in $R^*_U(\Delta)$.
- Those clauses that do not contain a literal l such that $\{l\}$ is one of the unit clauses have at least one negative literal.
- Assign true to these literals.
- Results in satisfying truth assignment for $R^*_U(\Delta)$ (and $\Delta \subseteq R^*_U(\Delta)$).

□

Horn clauses & resolution

Horn clauses: Clauses with at most one positive literal

Example: $(a \vee \neg b \vee \neg c), (\neg b \vee \neg c)$

Proposition

Unit resolution is refutation-complete for Horn clauses.

Proof idea.

Consider $R_U^*(\Delta)$ of Horn clause set Δ. We have to show that if $\square \notin R_U^*(\Delta)$, then $\Delta (\equiv R_U^*(\Delta))$ is satisfiable.

- Assign true to all unit clauses in $R_U^*(\Delta)$.
- Those clauses that do not contain a literal l such that $\{l\}$ is one of the unit clauses have at least one negative literal.
- Assign true to these literals.
- Results in satisfying truth assignment for $R_U^*(\Delta)$ (and $\Delta \subseteq R_U^*(\Delta)$).

□

Horn clauses & resolution

Horn clauses: Clauses with at most one positive literal

Example: $(a \lor \neg b \lor \neg c), (\neg b \lor \neg c)$

Proposition

Unit resolution is refutation-complete for Horn clauses.

Proof idea.

Consider $R_U^*(\Delta)$ of Horn clause set Δ. We have to show that if $\square \notin R_U^*(\Delta)$, then $\Delta (\equiv R_U^*(\Delta))$ is satisfiable.

- Assign true to all unit clauses in $R_U^*(\Delta)$.
- Those clauses that do not contain a literal l such that $\{l\}$ is one of the unit clauses have at least one negative literal.
- Assign true to these literals.
- Results in satisfying truth assignment for $R_U^*(\Delta)$ (and $\Delta \subseteq R_U^*(\Delta)$).

□

Horn clauses & resolution

Horn clauses: Clauses with at most one positive literal

Example: $(a \vee \neg b \vee \neg c), (\neg b \vee \neg c)$

Proposition

Unit resolution is refutation-complete for Horn clauses.

Proof idea.

Consider $R^*_U(\Delta)$ of Horn clause set Δ. We have to show that if $\square \notin R^*_U(\Delta)$, then $\Delta (\equiv R^*_U(\Delta))$ is satisfiable.

- Assign true to all unit clauses in $R^*_U(\Delta)$.
- Those clauses that do not contain a literal l such that $\{l\}$ is one of the unit clauses have at least one negative literal.
- Assign true to these literals.
- Results in satisfying truth assignment for $R^*_U(\Delta)$ (and $\Delta \subseteq R^*_U(\Delta)$).

□

Horn clauses & resolution

Horn clauses: Clauses with at most one positive literal

Example: $(a \vee \neg b \vee \neg c), (\neg b \vee \neg c)$

Proposition

Unit resolution is refutation-complete for Horn clauses.

Proof idea.

Consider $R_U^*(\Delta)$ of Horn clause set Δ. We have to show that if $\square \notin R_U^*(\Delta)$, then $\Delta (\equiv R_U^*(\Delta))$ is satisfiable.

- Assign true to all unit clauses in $R_U^*(\Delta)$.
- Those clauses that do not contain a literal l such that $\{l\}$ is one of the unit clauses have at least one negative literal.
- Assign true to these literals.
- Results in satisfying truth assignment for $R_U^*(\Delta)$ (and $\Delta \subseteq R_U^*(\Delta)$).

$\square$

Horn clauses & resolution

Horn clauses: Clauses with at most one positive literal
Example: $(a \vee \neg b \vee \neg c), (\neg b \vee \neg c)$

Proposition

Unit resolution is refutation-complete for Horn clauses.

Proof idea.

Consider $R^*_U(\Delta)$ of Horn clause set Δ. We have to show that if $\square \notin R^*_U(\Delta)$, then $\Delta (\equiv R^*_U(\Delta))$ is satisfiable.

- Assign true to all unit clauses in $R^*_U(\Delta)$.
- Those clauses that do not contain a literal l such that $\{l\}$ is one of the unit clauses have at least one negative literal.
- Assign true to these literals.
- Results in satisfying truth assignment for $R^*_U(\Delta)$ (and $\Delta \subseteq R^*_U(\Delta)$).

□

Literature

Harry R. Lewis and Christos H. Papadimitriou.
Elements of the Theory of Computation.
Prentice-Hall, Englewood Cliffs, NJ, 1981 (Chapters 8 & 9).

Volker Sperschneider and Grigorios Antoniou.
Logic – A Foundation for Computer Science.
Addison-Wesley, Reading, MA, 1991 (Chapters 1–3).

H.-P. Ebbinghaus, J. Flum, and W. Thomas.
Einführung in die mathematische Logik.
Wissenschaftliche Buchgesellschaft, Darmstadt, 1986.

U. Schöning.
Logik für Informatiker.
Spektrum-Verlag, 5th edition, 2000.