Theoretical Computer Science II (ACS II)

2. Propositional logic

Malte Helmert   Andreas Karwath

Albert-Ludwigs-Universität Freiburg

October 22th, 2009

Theoretical Computer Science II (ACS II)

October 22th, 2009 — 2. Propositional logic

Why logic?

- formalizing valid reasoning
- used throughout mathematics, computer science
- the basis of many tools in computer science

Examples of reasoning

Which are valid?

- If it is Sunday, then I don't need to work.
  It is Sunday.
  Therefore I don't need to work.
- It will rain or snow.
  It is too warm for snow.
  Therefore it will rain.
- The butler is guilty or the maid is guilty.
  The maid is guilty or the cook is guilty.
  Therefore either the butler is guilty or the cook is guilty.

Elements of logic

- Which elements are well-formed? $\rightsquigarrow$ syntax
- What does it mean for a formula to be true? $\rightsquigarrow$ semantics
- When does one formula follow from another? $\rightsquigarrow$ inference

Two logics:

- propositional logic
- first-order logic (aka predicate logic)

Building blocks of propositional logic

Building blocks of propositional logic:

- atomic propositions (atoms)
- connectives

Atomic propositions

indivisible statements

Examples:

- "The cook is guilty."
- "It rains."
- "The girl has red hair."

Connectives

operators to build composite formulae out of atoms

Examples:

- "and", "or", "not", ...

Logic: basic questions

We are interested in knowing the following:

- When is a formula true?
- When does one formula logically follow from (= is logically entailed by) a knowledge base (a set of formulae)?
  - symbolically: $\mathrm{KB} \models \varphi$ if KB entails φ
- How can we define an inference mechanism ($\approx$ proof procedure) that allows us to systematically derive consequences of a knowledge base?
  - symbolically: $\mathrm{KB} \vdash \varphi$ if φ can be derived from KB
- Can we find an inference mechanism in such a way that $\mathrm{KB} \models \varphi$ iff $\mathrm{KB} \vdash \varphi$?

Syntax of propositional logic

Given: finite or countable set Σ of atoms $p, q, r, \ldots$

Propositional formulae: inductively defined as

$p \in \Sigma$	atomic formulae
$\top$	truth
$\bot$	falseness
$\neg\varphi$	negation
$(\varphi \wedge \psi)$	conjunction
$(\varphi \vee \psi)$	disjunction
$(\varphi \rightarrow \psi)$	material conditional
$(\varphi \leftrightarrow \psi)$	biconditional

where φ and ψ are constructed in the same way

Logic terminology and notations

- atom/atomic formula (p)
- literal: atom or negated atom (p, $\neg p$)
- clause: disjunction of literals ($p \vee \neg q$, $p \vee q \vee r$, p)

Parentheses may be omitted according to the following rules:

- $\neg$ binds more tightly than $\wedge$
- $\wedge$ binds more tightly than $\vee$
- $\vee$ binds more tightly than $\rightarrow$ and $\leftrightarrow$
- $p \wedge q \wedge r \wedge s \ldots$ is read as $(\ldots(((p \wedge q) \wedge r) \wedge s) \wedge \ldots)$
- $p \vee q \vee r \vee s \ldots$ is read as $(\ldots(((p \vee q) \vee r) \vee s) \vee \ldots)$
- outermost parentheses can always be omitted

Alternative notations

our notation	alternative notations		
$\neg\varphi$	$\sim\varphi$	$\overline{\varphi}$	
$\varphi \wedge \psi$	$\varphi \,\&\, \psi$	φ, ψ	$\varphi \cdot \psi$
$\varphi \vee \psi$	$\varphi \mid \psi$	$\varphi ; \psi$	$\varphi + \psi$
$\varphi \rightarrow \psi$	$\varphi \Rightarrow \psi$	$\varphi \supset \psi$	
$\varphi \leftrightarrow \psi$	$\varphi \Leftrightarrow \psi$	$\varphi \equiv \psi$	

Semantics of propositional logic

Definition (truth assignment)

A truth assignment of the atoms in Σ, or interpretation over Σ, is a function $I : \Sigma \rightarrow \{\mathbf{T}, \mathbf{F}\}$

Idea: extend from atoms to arbitrary formulae

Semantics of propositional logic (ctd.)

Definition (satisfaction/truth)

I satisfies φ (alternatively: φ is true under I), in symbols $I \models \varphi$, according to the following inductive rules:

$I \models p$	iff $I(p) = \mathbf{T}$ for $p \in \Sigma$
$I \models \top$	always (i. e., for all I)
$I \models \bot$	never (i. e., for no I)
$I \models \neg\varphi$	iff $I \not\models \varphi$
$I \models \varphi \wedge \psi$	iff $I \models \varphi$ and $I \models \psi$
$I \models \varphi \vee \psi$	iff $I \models \varphi$ or $I \models \psi$
$I \models \varphi \rightarrow \psi$	iff $I \not\models \varphi$ or $I \models \psi$
$I \models \varphi \leftrightarrow \psi$	iff ($I \models \varphi$ and $I \models \psi$) or ($I \not\models \varphi$ and $I \not\models \psi$)

Semantics of propositional logic: example

Example

$\Sigma = \{p, q, r, s\}$
$I = \{p \mapsto \mathbf{T}, q \mapsto \mathbf{F}, r \mapsto \mathbf{F}, s \mapsto \mathbf{T}\}$
$\varphi = ((p \lor q) \leftrightarrow (r \lor s)) \land (\neg(p \land q) \lor (r \land \neg s))$

Question: $I \models \varphi$?

More logic terminology

Definition (model)

An interpretation I is called a model of a formula φ if $I \models \varphi$.

An interpretation I is called a model of a set of formula KB if it is a model of all formulae $\varphi \in$ KB.

Definition (properties of formulae)

A formula φ is called

- satisfiable if there exists a model of φ
- unsatisfiable if it is not satisfiable
- valid/a tautology if all interpretations are models of φ
- falsifiable if it is not a tautology

Note: All valid formulae are satisfiable.
All unsatisfiable formulae are falsifiable.

More logic terminology (ctd.)

Definition (logical equivalence)

Two formulae φ and ψ are logically equivalent, written $\varphi \equiv \psi$, if they have the same set of models.

In other words, $\varphi \equiv \psi$ holds if for all interpretations I, we have that $I \models \varphi$ iff $I \models \psi$.

The truth table method

How can we decide if a formula is satisfiable, valid, etc.?
$\leadsto$ one simple idea: generate a truth table

The characteristic truth table

p	q	$\neg p$	$p \land q$	$p \lor q$	$p \rightarrow q$	$p \leftrightarrow q$
F	**F**	**T**	**F**	**F**	**T**	**T**
F	**T**	**T**	**F**	**T**	**T**	**F**
T	**F**	**F**	**F**	**T**	**F**	**F**
T	**T**	**F**	**T**	**T**	**T**	**T**

Truth table method: example

Question: Is $((p \lor q) \land \neg q) \rightarrow p$ valid?

p	q	$p \lor q$	$(p \lor q) \land \neg q$	$((p \lor q) \land \neg q) \rightarrow p$
F	**F**	**F**	**F**	**T**
F	**T**	**T**	**F**	**T**
T	**F**	**T**	**T**	**T**
T	**T**	**T**	**F**	**T**

- φ is true for all possible combinations of truth values
- ⇝ all interpretations are models
- ⇝ φ is valid
- satisfiability, unsatisfiability, falsifiability likewise
- logical equivalence likewise

Some well known equivalences

Idempotence	$\varphi \land \varphi \equiv \varphi$
	$\varphi \lor \varphi \equiv \varphi$
Commutativity	$\varphi \land \psi \equiv \psi \land \varphi$
	$\varphi \lor \psi \equiv \psi \lor \varphi$
Associativity	$(\varphi \land \psi) \land \chi \equiv \varphi \land (\psi \land \chi)$
	$(\varphi \lor \psi) \lor \chi \equiv \varphi \lor (\psi \lor \chi)$
Absorption	$\varphi \land (\varphi \lor \psi) \equiv \varphi$
	$\varphi \lor (\varphi \land \psi) \equiv \varphi$
Distributivity	$\varphi \land (\psi \lor \chi) \equiv (\varphi \land \psi) \lor (\varphi \land \chi)$
	$\varphi \lor (\psi \land \chi) \equiv (\varphi \lor \psi) \land (\varphi \lor \chi)$
De Morgan	$\neg(\varphi \land \psi) \equiv \neg\varphi \lor \neg\psi$
	$\neg(\varphi \lor \psi) \equiv \neg\varphi \land \neg\psi$
Double negation	$\neg\neg\varphi \equiv \varphi$
$(\rightarrow)$-Elimination	$\varphi \rightarrow \psi \equiv \neg\varphi \lor \psi$
$(\leftrightarrow)$-Elimination	$\varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \land (\psi \rightarrow \varphi)$

Substitutability

Theorem (Substitutability)

Let φ and ψ be two equivalent formulae, i. e., $\varphi \equiv \psi$.

Let χ be a formula in which φ occurs as a subformula, and let χ' be the formula obtained from χ by substituting ψ for φ.

Then $\chi \equiv \chi'$.

Example: $p \lor \neg(q \lor r) \equiv p \lor (\neg q \land \neg r)$
by De Morgan's law and substitutability.

Applying equivalences: examples (1)

$$
\begin{aligned}
& p \land (\neg q \lor p) \\
\equiv\ & (p \land \neg q) \lor (p \land p) && \text{(Distributivity)} \\
\equiv\ & (p \land \neg q) \lor p && \text{(Idempotence)} \\
\equiv\ & p \lor (p \land \neg q) && \text{(Commutativity)} \\
\equiv\ & p && \text{(Absorption)}
\end{aligned}
$$

Applying equivalences: examples (2)

$p \leftrightarrow q$	
$\equiv (p \rightarrow q) \wedge (q \rightarrow p)$	$((\leftrightarrow)$-Elimination)
$\equiv (\neg p \vee q) \wedge (\neg q \vee p)$	$((\rightarrow)$-Elimination)
$\equiv ((\neg p \vee q) \wedge \neg q) \vee ((\neg p \vee q) \wedge p)$	(Distributivity)
$\equiv (\neg q \wedge (\neg p \vee q)) \vee (p \wedge (\neg p \vee q))$	(Commutativity)
$\equiv ((\neg q \wedge \neg p) \vee (\neg q \wedge q)) \vee$ $((p \wedge \neg p) \vee (p \wedge q))$	(Distributivity)
$\equiv ((\neg q \wedge \neg p) \vee \bot) \vee (\bot \vee (p \wedge q))$	$(\varphi \wedge \neg\varphi \equiv \bot)$
$\equiv (\neg q \wedge \neg p) \vee (p \wedge q)$	$(\varphi \vee \bot \equiv \varphi \equiv \bot \vee \varphi)$

Conjunctive normal form

Definition (conjunctive normal form)

A formula is in conjunctive normal form (CNF) if it consists of a conjunction of clauses, i. e., if it has the form

$$\bigwedge_{i=1}^{n} \left(\bigvee_{j=1}^{m_i} l_{ij} \right),$$

where the l_{ij} are literals.

Theorem: For each formula φ, there exists a logically equivalent formula in CNF.

Note: A CNF formula is valid iff every clause is valid.

Disjunctive normal form

Definition (disjunctive normal form)

A formula is in disjunctive normal form (DNF) if it consists of a disjunction of conjunctions of literals, i. e., if it has the form

$$\bigvee_{i=1}^{n} \left(\bigwedge_{j=1}^{m_i} l_{ij} \right),$$

where the l_{ij} are literals.

Theorem: For each formula φ, there exists a logically equivalent formula in DNF.

Note: A DNF formula is satisfiable iff at least one disjunct is satisfiable.

CNF and DNF examples

Examples

- $(p \vee \neg q) \wedge p$ is in CNF
- $(r \vee q) \wedge p \wedge (r \vee s)$ is in CNF
- $p \vee (\neg q \wedge r)$ is in DNF
- $p \vee \neg q \rightarrow p$ is neither in CNF nor in DNF
- p is in CNF and in DNF

Producing CNF

Algorithm for producing CNF

1. Get rid of $\rightarrow$ and $\leftrightarrow$ with $(\rightarrow)$-Elimination and $(\leftrightarrow)$-Elimination.
   $\rightsquigarrow$ formula structure: only $\vee$, $\wedge$, $\neg$
2. Move negations inwards with De Morgan and Double negation.
   $\rightsquigarrow$ formula structure: only $\vee$, $\wedge$, literals
3. Distribute $\vee$ over $\wedge$ with Distributivity (strictly speaking, also Commutativity).
   $\rightsquigarrow$ formula structure: CNF
4. Optionally, simplify (e. g., using Idempotence) at the end or at any previous point.

Note: For DNF, just distribute $\wedge$ over $\vee$ instead.
Question: runtime?

Producing CNF: example

Producing CNF

Given: $\varphi = ((p \vee r) \wedge \neg q) \rightarrow p$

$$\begin{aligned}
\varphi &\equiv \neg((p \vee r) \wedge \neg q) \vee p && \text{Step 1}\\
&\equiv (\neg(p \vee r) \vee \neg\neg q) \vee p && \text{Step 2}\\
&\equiv ((\neg p \wedge \neg r) \vee q) \vee p && \text{Step 2}\\
&\equiv ((\neg p \vee q) \wedge (\neg r \vee q)) \vee p && \text{Step 3}\\
&\equiv (\neg p \vee q \vee p) \wedge (\neg r \vee q \vee p) && \text{Step 3}\\
&\equiv \top \wedge (\neg r \vee q \vee p) && \text{Step 4}\\
&\equiv \neg r \vee q \vee p && \text{Step 4}
\end{aligned}$$

Logical entailment

A set of formulae (a knowledge base) usually provides an incomplete description of the world, i. e., it leaves the truth values of some propositions open.

Example: $\text{KB} = \{p \vee q, r \vee \neg p, s\}$ is definitive w.r.t. s, but leaves p, q, r open (though not completely!)

Models of the KB

p	q	r	s
F	**T**	**F**	**T**
F	**T**	**F**	**T**
T	**F**	**T**	**T**
T	**T**	**T**	**T**

In all models, $q \vee r$ is true. Hence, $q \vee r$ is logically entailed by KB (a logical consequence of KB).

Logical entailment: formally

Definition (entailment)

Let KB be a set of formulae and φ be a formula.
We say that KB entails φ (also: φ follows logically from KB; φ is a logical consequence of KB), in symbols $\text{KB} \models \varphi$, if all models of KB are models of φ.

Properties of entailment

Some properties of logical entailment:

- Deduction theorem:
  $\mathrm{KB} \cup \{\varphi\} \models \psi$ iff $\mathrm{KB} \models \varphi \rightarrow \psi$
- Contraposition theorem:
  $\mathrm{KB} \cup \{\varphi\} \models \neg\psi$ iff $\mathrm{KB} \cup \{\psi\} \models \neg\varphi$
- Contradiction theorem:
  $\mathrm{KB} \cup \{\varphi\}$ is unsatisfiable iff $\mathrm{KB} \models \neg\varphi$

Proof of the deduction theorem

Deduction theorem: $\mathrm{KB} \cup \{\varphi\} \models \psi$ iff $\mathrm{KB} \models \varphi \rightarrow \psi$

Proof.
"$\Rightarrow$": The premise is that $\mathrm{KB} \cup \{\varphi\} \models \psi$.
We must show that $\mathrm{KB} \models \varphi \rightarrow \psi$, i. e., that all models of KB satisfy $\varphi \rightarrow \psi$. Consider any such model I.
We distinguish two cases:

- Case 1: $I \models \varphi$.
  Then I is a model of $\mathrm{KB} \cup \{\varphi\}$, and by the premise, $I \models \psi$, from which we conclude that $I \models \varphi \rightarrow \psi$.
- Case 2: $I \not\models \varphi$.
  Then we can directly conclude that $I \models \varphi \rightarrow \psi$.

...

Proof of the deduction theorem

Deduction theorem: $\mathrm{KB} \cup \{\varphi\} \models \psi$ iff $\mathrm{KB} \models \varphi \rightarrow \psi$

Proof (ctd.)
"$\Leftarrow$": The premise is that $\mathrm{KB} \models \varphi \rightarrow \psi$.
We must show that $\mathrm{KB} \cup \{\varphi\} \models \psi$, i. e., that all models of $\mathrm{KB} \cup \{\varphi\}$ satisfy ψ. Consider any such model I.

By definition, $I \models \varphi$. Moreover, as I is a model of KB, we have $I \models \varphi \rightarrow \psi$ by the premise.

Putting this together, we get $I \models \varphi \wedge (\varphi \rightarrow \psi) \equiv \varphi \wedge \psi$, which implies that $I \models \psi$. □

Proof of the contraposition theorem

Contraposition theorem: $\mathrm{KB} \cup \{\varphi\} \models \neg\psi$ iff $\mathrm{KB} \cup \{\psi\} \models \neg\varphi$

Proof.
By the deduction theorem, $\mathrm{KB} \cup \{\varphi\} \models \neg\psi$ iff $\mathrm{KB} \models \varphi \rightarrow \neg\psi$.
For the same reason, $\mathrm{KB} \cup \{\psi\} \models \neg\varphi$ iff $\mathrm{KB} \models \psi \rightarrow \neg\varphi$.

We have $\varphi \rightarrow \neg\psi \equiv \neg\varphi \vee \neg\psi \equiv \neg\psi \vee \neg\varphi \equiv \psi \rightarrow \neg\varphi$.

Putting this together, we get

$$\begin{aligned} & \mathrm{KB} \cup \{\varphi\} \models \neg\psi \\ \text{iff} \quad & \mathrm{KB} \models \neg\varphi \vee \neg\psi \\ \text{iff} \quad & \mathrm{KB} \cup \{\psi\} \models \neg\varphi \end{aligned}$$

as required. □

Inference rules, calculi and proofs

Question: Can we determine whether KB $\models \varphi$ without considering all interpretations (the truth table method)?

- Yes! There are various ways of doing this.
- One is to use inference rules that produce formulae that follow logically from a given set of formulae.
- Inference rules are written in the form

$$\frac{\varphi_1, \ldots, \varphi_k}{\psi},$$

  meaning "if $\varphi_1, \ldots, \varphi_k$ are true, then ψ is also true."
- $k = 0$ is allowed; such inference rules are called axioms.
- A set of inference rules is called a calculus or proof system.

Some inference rules for propositional logic

Modus ponens	$\frac{\varphi,\ \varphi \rightarrow \psi}{\psi}$
Modus tolens	$\frac{\neg\psi,\ \varphi \rightarrow \psi}{\neg\varphi}$
And elimination	$\frac{\varphi \wedge \psi}{\varphi}$ $\quad$ $\frac{\varphi \wedge \psi}{\psi}$
And introduction	$\frac{\varphi,\ \psi}{\varphi \wedge \psi}$
Or introduction	$\frac{\varphi}{\varphi \vee \psi}$
$(\bot)$ elimination	$\frac{\bot}{\varphi}$
$(\leftrightarrow)$ elimination	$\frac{\varphi \leftrightarrow \psi}{\varphi \rightarrow \psi}$ $\quad$ $\frac{\varphi \leftrightarrow \psi}{\psi \rightarrow \varphi}$

Derivations

Definition (derivation)

A derivation or proof of a formula φ from a knowledge base KB is a sequence of formulae $\psi_1, \ldots, \psi_k$ such that

- $\psi_k = \varphi$ and
- for all $i \in \{1, \ldots, k\}$:
  - $\psi_i \in$ KB, or
  - ψ_i is the result of applying an inference rule to some elements of $\{\psi_1, \ldots, \psi_{i-1}\}$.

Derivation example

Example

Given: KB $= \{p, p \rightarrow q, p \rightarrow r, q \wedge r \rightarrow s\}$
Objective: Give a derivation of $s \wedge r$ from KB.

1. p (KB)
2. $p \rightarrow q$ (KB)
3. q (1, 2, modus ponens)
4. $p \rightarrow r$ (KB)
5. r (1, 4, modus ponens)
6. $q \wedge r$ (3, 5, and introduction)
7. $q \wedge r \rightarrow s$ (KB)
8. s (6, 7, modus ponens)
9. $s \wedge r$ (8, 5, and introduction)

Soundness and completeness

Definition (KB $\vdash_{\mathbf{C}} \varphi$, soundness, completeness)

We write KB $\vdash_{\mathbf{C}} \varphi$ if there is a derivation of φ from KB in calculus **C**.
(We often omit **C** when it is clear from context.)

A calculus **C** is sound or correct if for all KB and φ,
we have that KB $\vdash_{\mathbf{C}} \varphi$ implies KB $\models \varphi$.

A calculus **C** is complete if for all KB and φ,
we have that KB $\models \varphi$ implies KB $\vdash_{\mathbf{C}} \varphi$.

Consider the calculus **C** given by the derivation rules shown previously.
Question: Is **C** sound?
Question: Is **C** complete?

Refutation-completeness

- Clearly we want sound calculi.
- Do we also need complete calculi?
- Recall the contradiction theorem:
  KB $\cup \{\varphi\}$ is unsatisfiable iff KB $\models \neg\varphi$
- This implies that KB $\models \varphi$ iff KB $\cup \{\neg\varphi\}$ is unsatisfiable,
  i. e., KB $\models \varphi$ iff KB $\cup \{\neg\varphi\} \models \bot$.
- Hence, we can reduce the general entailment problem to testing entailment of $\bot$.

Definition (refutation-complete)

A calculus **C** is refutation-complete if for all KB,
we have that KB $\models \bot$ implies KB $\vdash_{\mathbf{C}} \bot$.

Question: What is the relationship between completeness and refutation-completeness?

Resolution: idea

- Resolution is a refutation-complete calculus for knowledge bases in CNF.
- For knowledge bases that are not in CNF, we can convert them to equivalent formulae in CNF.
  - However, this conversion can take exponential time.
  - Alternatively, we can convert to a satisfiability-equivalent (but not logically equivalent) knowledge base in polynomial time.
- To test if KB $\models \varphi$, we test if KB $\cup \{\neg\varphi\} \vdash_{\mathbf{R}} \bot$,
  where **R** is the resolution calculus.
  (In the following, we simply write $\vdash$ instead of $\vdash_{\mathbf{R}}$.)
- In the worst case, resolution takes exponential time.
- However, this is probably true for all refutation complete proof methods, as we will see in the computational complexity part of the course.

Knowledge bases as clause sets

- Resolution requires that knowledge bases are given in CNF.
- In this case, we can simplify notation:
  - A formula in CNF can be equivalently seen as a set of clauses (due to commutativity, idempotence and associativity of $(\vee)$).
  - A set of formulae can then also be seen as a set of clauses.
  - A clause can be seen as a set of literals (due to commutativity, idempotence and associativity of $(\wedge)$).
  - So a knowledge base can be represented as a set of sets of literals.
- Example:
  - $\mathrm{KB} = \{(p \vee p), (\neg p \vee q) \wedge (\neg p \vee r) \wedge (\neg p \vee q) \wedge r, (\neg q \vee \neg r \vee s) \wedge p\}$
  - as clause set: $\{\{p\}, \{\neg p, q\}, \{\neg p, r\}, \{r\}, \{\neg q, \neg r, s\}\}$

Resolution: notation, empty clauses

- In the following, we use common logical notation for sets of literals (treating them as clauses) and sets of sets of literals (treating them as CNF formulae).
- Example:
  - Let $I = \{p \mapsto 1, q \mapsto 1, r \mapsto 1, s \mapsto 1\}$.
  - Let $\Delta = \{\{p\}, \{\neg p, q\}, \{\neg p, r\}, \{r\}, \{\neg q, \neg r, s\}\}$.
  - We can write $I \models \Delta$.
- One notation ambiguity:
  - Does the empty set mean an empty clause (equivalent to $\bot$) or an empty set of clauses (equivalent to $\top$)?
  - To resolve this ambiguity, the empty clause is written as $\square$, while the empty set of clauses is written as $\emptyset$.

The resolution rule

The resolution calculus consists of a single rule,
called the resolution rule:

$$\frac{C_1 \cup \{l\},\ C_2 \cup \{\neg l\}}{C_1 \cup C_2},$$

where C_1 and C_2 are (possibly empty) clauses, and
l is an atom (and hence l and $\neg l$ are complementary literals).

In the rule above,

- l and $\neg l$ are called the resolution literals,
- $C_1 \cup \{l\}$ and $C_2 \cup \{\neg l\}$ are called the parent clauses, and
- $C_1 \cup C_2$ is called the resolvent.

Resolution proofs

Definition (resolution proof)

Let Δ be a set of clauses. We define the resolvents of Δ as
$\mathbf{R}(\Delta) := \Delta \cup \{ C \mid C \text{ is a resolvent of two clauses from } \Delta \}$.

A resolution proof of a clause D from Δ, is a sequence of clauses $C_1, \ldots, C_n$ with

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

We say that D can be derived from Δ by resolution, written $\Delta \vdash_{\mathbf{R}} D$, if there exists a resolution proof of D from Δ.

Remarks: Resolution is a sound and refutation-complete,
but incomplete proof system.

Resolution proofs: example

Using resolution for testing entailment: example

Let $\text{KB} = \{p, p \rightarrow (q \wedge r)\}$.
We want to use resolution to show that show that $\text{KB} \models r \vee s$.
Three steps:

1. Reduce entailment to unsatisfiability.
2. Convert resulting knowledge base to clause form (CNF).
3. Derive empty clause by resolution.

Step 1: Reduce entailment to unsatisfiability.

$\text{KB} \models r \vee s$ iff $\text{KB} \cup \{\neg(r \vee s)\}$ is unsatisfiable.
Hence, consider $\text{KB}' = \text{KB} \cup \{\neg(r \vee s)\} = \{p, p \rightarrow (q \wedge r), \neg(r \vee s)\}$.

...

Resolution proofs: example (ctd.)

Using resolution for testing entailment: example (ctd.)

$\mathrm{KB}' = \mathrm{KB} \cup \{\neg(r \vee s)\} = \{p, p \to (q \wedge r), \neg(r \vee s)\}$.

Step 2: Convert resulting knowledge base to clause form (CNF).

p
$\rightsquigarrow$ clauses: $\{p\}$

$p \to (q \wedge r) \equiv \neg p \vee (q \wedge r) \equiv (\neg p \vee q) \wedge (\neg p \vee r)$
$\rightsquigarrow$ clauses: $\{\neg p, q\}, \{\neg p, r\}$

$\neg(r \vee s) \equiv \neg r \wedge \neg s$
$\rightsquigarrow$ clauses: $\{\neg r\}, \{\neg s\}$

$\Delta = \{\{p\}, \{\neg p, q\}, \{\neg p, r\}, \{\neg r\}, \{\neg s\}\}$

...

Resolution proofs: example (ctd.)

Using resolution for testing entailment: example (ctd.)

$\Delta = \{\{p\}, \{\neg p, q\}, \{\neg p, r\}, \{\neg r\}, \{\neg s\}\}$

Step 3: Derive empty clause by resolution.

- $C_1 = \{p\}$ (from Δ)
- $C_2 = \{\neg p, q\}$ (from Δ)
- $C_3 = \{\neg p, r\}$ (from Δ)
- $C_4 = \{\neg r\}$ (from Δ)
- $C_5 = \{\neg s\}$ (from Δ)
- $C_6 = \{q\}$ (from C_1 and C_2)
- $C_7 = \{\neg p\}$ (from C_3 and C_4)
- $C_8 = \square$ (from C_1 and C_7)

Note: Much shorter proofs exist. (For example?)

Another example

Another resolution example

We want to prove $\{p \to q, q \to r\} \models p \to r$.

Larger example: blood types

We know the following:

- If test T is positive, the person has blood type A or AB.
- If test S is positive, the person has blood type B or AB.
- If a person has blood type A, then test T will be positive.
- If a person has blood type B, then test S will be positive.
- If a person has blood type AB, both tests will be positive.
- A person has exactly one of the blood types A, B, AB, 0.
- Suppose T is true and S is false for a given person.

Prove that the person must have blood type A or 0.

Summary

- Logics are mathematical approaches for formalizing reasoning.
- Propositional logic is one logic which is of particular relevance to computer science.
- Three important components of all forms of logic include:
  - Syntax formalizes what statements can be expressed.
    $\rightsquigarrow$ atoms, connectives, formulae, ...
  - Semantics formalizes what these statements mean.
    $\rightsquigarrow$ interpretations, models, satisfiable, valid, ...
  - Calculi (proof systems) provide formal rules for deriving conclusions from a set of given statements.
    $\rightsquigarrow$ inference rules, derivations, sound, complete, refutation-complete, ...
- We had a closer look at the resolution calculus, which is a sound and refutation-complete proof system.

Further topics

There are many further topics we did not discuss:

- resolution strategies to make resolution as efficient as possible in practice
- other proof systems, for example tableaux proofs
- algorithms for model construction, for example the Davis-Putnam-Logemann-Loveland (DPLL) procedure

These topics are discussed in advanced courses, such as:

- Foundations of Artificial Intelligence (every summer semester)
- Principles of Knowledge Representation and Reasoning (no fixed schedule; roughly once in two years)
- Modal Logic (no fixed schedule; infrequently)