

# Formal Specification and Verification

1. Classical Logic

24.04.2014

Viorica Sofronie-Stokkermans

e-mail: [sofronie@uni-koblenz.de](mailto:sofronie@uni-koblenz.de)

# Mathematical foundations:

---

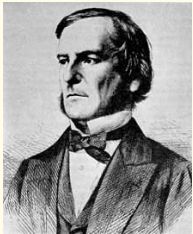
Classical first-order logic

# Mathematical foundations:

---

## Classical first-order logic

**George Boole** (1815-1864)



George Boole is best known as the author of “The Laws of Thought” (1854). He is the inventor of the prototype of what is now called Boolean logic. Because of this Boole is also regarded as a founder of the field of computer science.

# Mathematical foundations:

---

## Classical first-order logic

**David Hilbert** (1862-1943)



David Hilbert is recognized as one of the most influential and universal mathematicians of the 19th and early 20th centuries.

A famous example of his leadership in mathematics is his 1900 presentation of a collection of 23 problems that set the course for much of the mathematical research of the 20th century.

Hilbert is known as one of the founders of proof theory and mathematical logic.

# Mathematical foundations:

---

## Classical first-order logic

### Gottlob Frege (1848-1925)



Gottlob Frege is considered to be one of the founders of modern logic and made major contributions to the foundations of mathematics.

Frege invented axiomatic predicate logic, in large part thanks to the fact that he introduced and used quantified variables.

# Formal logics

---

A formal logic consists of:

- **Syntax:** a formal language (formula expressing facts)
- **Semantics:** to define the meaning of the language, that is which facts are valid)
- **Deductive system:** made of axioms and inference rules to formally derive theorems, that is facts that are provable

# Questions about formal logics

---

The main questions about a formal logic are:

- The soundness of the deductive system: no provable formula is invalid
- The completeness of the deductive system: all valid formulae are provable

# Part 1: Propositional classical logic

---

**Literature** (also for first-order logic)

Schöning: Logik für Informatiker, Spektrum

Fitting: First-Order Logic and Automated Theorem Proving, Springer



# 1.1 Syntax

---

- propositional variables
- logical symbols
  - ⇒ Boolean combinations

# Propositional Variables

---

Let  $\Pi$  be a set of **propositional variables**.

We use letters  $P, Q, R, S$ , to denote propositional variables.

# Propositional Formulas

---

$F_{\Pi}$  is the set of propositional formulas over  $\Pi$  defined as follows:

$F, G, H$	$::=$	$\perp$	(falsum)
		$\top$	(verum)
		$P, P \in \Pi$	(atomic formula)
		$\neg F$	(negation)
		$(F \wedge G)$	(conjunction)
		$(F \vee G)$	(disjunction)
		$(F \rightarrow G)$	(implication)
		$(F \leftrightarrow G)$	(equivalence)

# Notational Conventions

---

- We omit brackets according to the following rules:
  - $\neg > \wedge > \vee > \rightarrow > \leftrightarrow$  (binding precedences)
  - $\vee$  and  $\wedge$  are associative and commutative

## 1.2 Semantics

---

In **classical logic** (dating back to Aristoteles) there are “only” two truth values “true” and “false” which we shall denote, respectively, by 1 and 0.

There are **multi-valued logics** having more than two truth values.

# Valuations

---

A propositional variable has no intrinsic meaning. The meaning of a propositional variable has to be defined by a valuation.

A  $\Pi$ -valuation is a map

$$\mathcal{A} : \Pi \rightarrow \{0, 1\}.$$

where  $\{0, 1\}$  is the set of **truth values**.

# Truth Value of a Formula in $\mathcal{A}$

---

Given a  $\Pi$ -valuation  $\mathcal{A}$ , the function  $\mathcal{A}^* : \Sigma\text{-formulas} \rightarrow \{0, 1\}$  is defined inductively over the structure of  $F$  as follows:

$$\mathcal{A}^*(\perp) = 0$$

$$\mathcal{A}^*(\top) = 1$$

$$\mathcal{A}^*(P) = \mathcal{A}(P)$$

$$\mathcal{A}^*(\neg F) = 1 - \mathcal{A}^*(F)$$

$$\mathcal{A}^*(F \rho G) = B_\rho(\mathcal{A}^*(F), \mathcal{A}^*(G))$$

with  $B_\rho$  the Boolean function associated with  $\rho$

For simplicity, we write  $\mathcal{A}$  instead of  $\mathcal{A}^*$ .

# Truth Value of a Formula in $\mathcal{A}$

---

**Example:** Let's evaluate the formula

$$(P \rightarrow Q) \wedge (P \wedge Q \rightarrow R) \rightarrow (P \rightarrow R)$$

w.r.t. the valuation  $\mathcal{A}$  with

$$\mathcal{A}(P) = 1, \mathcal{A}(Q) = 0, \mathcal{A}(R) = 1$$

(On the blackboard)



## 1.3 Models, Validity, and Satisfiability

---

$F$  is **valid** in  $\mathcal{A}$  ( $\mathcal{A}$  is a **model** of  $F$ ;  $F$  holds under  $\mathcal{A}$ ):

$$\mathcal{A} \models F :\Leftrightarrow \mathcal{A}(F) = 1$$

$F$  is **valid** (or is a **tautology**):

$$\models F :\Leftrightarrow \mathcal{A} \models F \text{ for all } \Pi\text{-valuations } \mathcal{A}$$

$F$  is called **satisfiable** iff there exists an  $\mathcal{A}$  such that  $\mathcal{A} \models F$ .

Otherwise  $F$  is called **unsatisfiable** (or **contradictory**).

## 1.3 Models, Validity, and Satisfiability

---

### Examples:

$F \rightarrow F$  and  $F \vee \neg F$  are **valid** for all formulae  $F$ .

Obviously, every **valid** formula is also **satisfiable**

$F \wedge \neg F$  is **unsatisfiable**

The formula  $P$  is **satisfiable**, but not **valid**

# Entailment and Equivalence

---

$F$  entails (implies)  $G$  (or  $G$  is a consequence of  $F$ ), written  $F \models G$ , if for all  $\Pi$ -valuations  $\mathcal{A}$ , whenever  $\mathcal{A} \models F$  then  $\mathcal{A} \models G$ .

$F$  and  $G$  are called **equivalent** if for all  $\Pi$ -valuations  $\mathcal{A}$  we have  $\mathcal{A} \models F \Leftrightarrow \mathcal{A} \models G$ .

## Proposition 1.1:

$F$  entails  $G$  iff  $(F \rightarrow G)$  is valid

## Proposition 1.2:

$F$  and  $G$  are equivalent iff  $(F \leftrightarrow G)$  is valid.

# Entailment and Equivalence

---

Extension to sets of formulas  $N$  in the “natural way”, e.g.,  $N \models F$  if for all  $\Pi$ -valuations  $\mathcal{A}$ : if  $\mathcal{A} \models G$  for all  $G \in N$ , then  $\mathcal{A} \models F$ .

# Validity vs. Unsatisfiability

---

Validity and unsatisfiability are just two sides of the same medal as explained by the following proposition.

## Proposition 1.3:

$$F \text{ valid} \iff \neg F \text{ unsatisfiable}$$

$$N \models F \iff N \cup \neg F \text{ unsatisfiable}$$

Hence in order to design a theorem prover (validity/entailment checker) it is sufficient to design a checker for unsatisfiability.

# Checking Unsatisfiability

---

Every formula  $F$  contains only finitely many propositional variables. Obviously,  $\mathcal{A}(F)$  depends only on the values of those finitely many variables in  $F$  under  $\mathcal{A}$ .

If  $F$  contains  $n$  distinct propositional variables, then it is sufficient to check  $2^n$  valuations to see whether  $F$  is satisfiable or not.

$\Rightarrow$  truth table.

So the satisfiability problem is clearly decidable (but, by Cook's Theorem, NP-complete).

Nevertheless, in practice, there are (much) better methods than truth tables to check the satisfiability of a formula. (later more)

# Some Important Equivalences

---

The following equivalences are valid for all formulas  $F, G, H$ :

$$(F \wedge F) \leftrightarrow F$$

$$(F \vee F) \leftrightarrow F$$

(Idempotency)

$$(F \wedge G) \leftrightarrow (G \wedge F)$$

$$(F \vee G) \leftrightarrow (G \vee F)$$

(Commutativity)

$$(F \wedge (G \wedge H)) \leftrightarrow ((F \wedge G) \wedge H)$$

$$(F \vee (G \vee H)) \leftrightarrow ((F \vee G) \vee H)$$

(Associativity)

$$(F \wedge (G \vee H)) \leftrightarrow ((F \wedge G) \vee (F \wedge H))$$

$$(F \vee (G \wedge H)) \leftrightarrow ((F \vee G) \wedge (F \vee H))$$

(Distributivity)

# Some Important Equivalences

---

The following equivalences are valid for all formulas  $F, G, H$ :

$$(F \wedge (F \vee G)) \leftrightarrow F$$

$$(F \vee (F \wedge G)) \leftrightarrow F$$

(Absorption)

$$(\neg\neg F) \leftrightarrow F$$

(Double Negation)

$$\neg(F \wedge G) \leftrightarrow (\neg F \vee \neg G)$$

$$\neg(F \vee G) \leftrightarrow (\neg F \wedge \neg G)$$

(De Morgan's Laws)

$$(F \wedge G) \leftrightarrow F, \text{ if } G \text{ is a tautology}$$

$$(F \vee G) \leftrightarrow \top, \text{ if } G \text{ is a tautology}$$

(Tautology Laws)

$$(F \wedge G) \leftrightarrow \perp, \text{ if } G \text{ is unsatisfiable}$$

$$(F \vee G) \leftrightarrow F, \text{ if } G \text{ is unsatisfiable}$$

(Tautology Laws)



## 1.4 Normal Forms

---

We define **conjunctions** of formulas as follows:

$$\bigwedge_{i=1}^0 F_i = \top.$$

$$\bigwedge_{i=1}^1 F_i = F_1.$$

$$\bigwedge_{i=1}^{n+1} F_i = \bigwedge_{i=1}^n F_i \wedge F_{n+1}.$$

and analogously **disjunctions**:

$$\bigvee_{i=1}^0 F_i = \perp.$$

$$\bigvee_{i=1}^1 F_i = F_1.$$

$$\bigvee_{i=1}^{n+1} F_i = \bigvee_{i=1}^n F_i \vee F_{n+1}.$$

# Literals and Clauses

---

A **literal** is either a propositional variable  $P$  or a negated propositional variable  $\neg P$ .

A **clause** is a (possibly empty) disjunction of literals.

# Literals and Clauses

---

A **literal** is either a propositional variable  $P$  or a negated propositional variable  $\neg P$ .

A **clause** is a (possibly empty) disjunction of literals.

## Example of clauses:

$\perp$

$P$

$\neg P$

$P \vee Q \vee R$

$P \vee \neg Q \vee \neg R$

$P \vee P \vee \neg Q \vee \neg R \vee R$

the empty clause

positive unit clause

negative unit clause

positive clause

clause

allow repetitions/complementary literals

# CNF and DNF

---

A formula is in **conjunctive normal form (CNF, clause normal form)**, if it is a conjunction of disjunctions of literals (or in other words, a conjunction of clauses).

A formula is in **disjunctive normal form (DNF)**, if it is a disjunction of conjunctions of literals.

Warning: definitions in the literature differ:

- are complementary literals permitted?

- are duplicated literals permitted?

- are empty disjunctions/conjunctions permitted?

# Conversion to CNF/DNF

---

## Proposition 1.4:

For every formula there is an equivalent formula in CNF (and also an equivalent formula in DNF).

Proof:

We consider the case of CNF.

Apply the following rules as long as possible (modulo associativity and commutativity of  $\wedge$  and  $\vee$ ):

**Step 1:** Eliminate equivalences:

$$(F \leftrightarrow G) \Rightarrow_K (F \rightarrow G) \wedge (G \rightarrow F)$$

# Conversion to CNF/DNF

---

**Step 2:** Eliminate implications:

$$(F \rightarrow G) \Rightarrow_K (\neg F \vee G)$$

**Step 3:** Push negations downward:

$$\neg(F \vee G) \Rightarrow_K (\neg F \wedge \neg G)$$

$$\neg(F \wedge G) \Rightarrow_K (\neg F \vee \neg G)$$

**Step 4:** Eliminate multiple negations:

$$\neg\neg F \Rightarrow_K F$$

The formula obtained from a formula  $F$  after applying steps 1-4 is called the **negation normal form (NNF)** of  $F$

# Conversion to CNF/DNF

---

**Step 5:** Push disjunctions downward:

$$(F \wedge G) \vee H \Rightarrow_K (F \vee H) \wedge (G \vee H)$$

**Step 6:** Eliminate  $\top$  and  $\perp$ :

$$(F \wedge \top) \Rightarrow_K F$$

$$(F \wedge \perp) \Rightarrow_K \perp$$

$$(F \vee \top) \Rightarrow_K \top$$

$$(F \vee \perp) \Rightarrow_K F$$

$$\neg \perp \Rightarrow_K \top$$

$$\neg \top \Rightarrow_K \perp$$

# Conversion to CNF/DNF

---

Proving termination is easy for most of the steps; only step 3 and step 5 are a bit more complicated.

The resulting formula is equivalent to the original one and in CNF.

The conversion of a formula to DNF works in the same way, except that disjunctions have to be pushed downward in step 5.



# Complexity

---

Conversion to CNF (or DNF) may produce a formula whose size is **exponential** in the size of the original one.

# Satisfiability-preserving Transformations

---

The goal

“find a formula  $G$  in CNF such that  $\models F \leftrightarrow G$ ”

is unpractical.

But if we relax the requirement to

“find a formula  $G$  in CNF such that  $F \models \perp$  iff  $G \models \perp$ ”

we can get an efficient transformation.

# Satisfiability-preserving Transformations

---

## Idea:

A formula  $F[F']$  is satisfiable iff  $F[P] \wedge (P \leftrightarrow F')$  is satisfiable (where  $P$  new propositional variable that works as abbreviation for  $F'$ ).

We can use this rule recursively for all subformulas in the original formula (this introduces a linear number of new propositional variables).

Conversion of the resulting formula to CNF increases the size only by an additional factor (each formula  $P \leftrightarrow F'$  gives rise to at most one application of the distributivity law).

# Optimized Transformations

---

A further improvement is possible by taking the **polarity** of the subformula  $F$  into account.

Assume that  $F$  contains neither  $\rightarrow$  nor  $\leftrightarrow$ . A subformula  $F'$  of  $F$  has **positive polarity** in  $F$ , if it occurs below an even number of negation signs; it has **negative polarity** in  $F$ , if it occurs below an odd number of negation signs.

# Optimized Transformations

---

## Proposition 1.5:

Let  $F[F']$  be a formula containing neither  $\rightarrow$  nor  $\leftrightarrow$ ; let  $P$  be a propositional variable not occurring in  $F[F']$ .

If  $F'$  has positive polarity in  $F$ , then  $F[F']$  is satisfiable if and only if  $F[P] \wedge (P \rightarrow F')$  is satisfiable.

If  $F'$  has negative polarity in  $F$ , then  $F[F']$  is satisfiable if and only if  $F[P] \wedge (F' \rightarrow P)$  is satisfiable.

Proof:

Exercise.

This satisfiability-preserving transformation to clause form is also called **structure-preserving transformation to clause form**.

# Optimized Transformations

---

**Example:** Let  $F = (Q_1 \wedge Q_2) \vee (R_1 \wedge R_2)$ .

The following are equivalent:

- $F \models \perp$
- $P_F \wedge (P_F \leftrightarrow (P_{Q_1 \wedge Q_2} \vee P_{R_1 \wedge R_2})) \wedge (P_{Q_1 \wedge Q_2} \leftrightarrow (Q_1 \wedge Q_2))$   
 $\wedge (P_{R_1 \wedge R_2} \leftrightarrow (R_1 \wedge R_2)) \models \perp$
- $P_F \wedge (P_F \rightarrow (P_{Q_1 \wedge Q_2} \vee P_{R_1 \wedge R_2})) \wedge (P_{Q_1 \wedge Q_2} \rightarrow (Q_1 \wedge Q_2))$   
 $\wedge (P_{R_1 \wedge R_2} \rightarrow (R_1 \wedge R_2)) \models \perp$
- $P_F \wedge (\neg P_F \vee P_{Q_1 \wedge Q_2} \vee P_{R_1 \wedge R_2}) \wedge (\neg P_{Q_1 \wedge Q_2} \vee Q_1) \wedge (\neg P_{Q_1 \wedge Q_2} \vee Q_2)$   
 $\wedge (\neg P_{R_1 \wedge R_2} \vee R_1) \wedge (\neg P_{R_1 \wedge R_2} \vee R_2) \models \perp$

# Proof methods

---

- Simple Decision Procedures  
truth table method

- Deductive methods

- Forward reasoning

- Assumptions and axioms are logically combined by inference rules to reason towards the goal

- Backward reasoning

- Inference rules are directly applied to the goal, possibly generating new subgoals.

# Proof methods

---

- Simple Decision Procedures
  - truth table method
- Deductive methods
  - Inference Systems and Proofs (generalities)
  - Example: Hilbert Deductive System
  - The Resolution Procedure
  - Sequent calculi
  - The DPLL procedure



# 1.5 Inference Systems and Proofs

---

Inference systems  $\Gamma$  (proof calculi) are sets of tuples

$$(F_1, \dots, F_n, F_{n+1}), \quad n \geq 0,$$

called inferences or inference rules, and written

$$\frac{\overbrace{F_1 \dots F_n}^{\text{premises}}}{\underbrace{F_{n+1}}_{\text{conclusion}}} .$$

**Clausal inference system:** premises and conclusions are clauses. One also considers inference systems over other data structures.

# Proofs

---

A **proof** in  $\Gamma$  of a formula  $F$  from a set of formulas  $N$  (called **assumptions**) is a sequence  $F_1, \dots, F_k$  of formulas where

(i)  $F_k = F$ ,

(ii) for all  $1 \leq i \leq k$ :  $F_i \in N$ , or else there exists an inference  $(F_{i_1}, \dots, F_{i_{n_i}}, F_i)$  in  $\Gamma$ , such that  $0 \leq i_j < i$ , for  $1 \leq j \leq n_i$ .

# Soundness and Completeness

---

**Provability**  $\vdash_{\Gamma}$  of  $F$  from  $N$  in  $\Gamma$ :

$N \vdash_{\Gamma} F \iff$  there exists a proof  $\Gamma$  of  $F$  from  $N$ .

$\Gamma$  is called **sound**  $\iff$

$$\frac{F_1 \dots F_n}{F} \in \Gamma \Rightarrow F_1, \dots, F_n \models F$$

$\Gamma$  is called **complete**  $\iff$

$$N \models F \Rightarrow N \vdash_{\Gamma} F$$

$\Gamma$  is called **refutationally complete**  $\iff$

$$N \models \perp \Rightarrow N \vdash_{\Gamma} \perp$$

# A deductive system for Propositional logic

---

## Variant of the system of Hilbert-Ackermann

(Signature:  $\vee, \neg; \quad x \rightarrow y \equiv_{\text{Def}} \neg x \vee y$ )

**Axiom Schemata** (to be instantiated for all possible formulae)

$$(1) (p \vee p) \rightarrow p$$

$$(2) p \rightarrow (q \vee p)$$

$$(3) (p \vee q) \rightarrow (q \vee p)$$

$$(4) (p \rightarrow q) \rightarrow (r \vee p \rightarrow r \vee q)$$

## Inference rules

Modus Ponens: 
$$\frac{p, \quad p \rightarrow q}{q}$$

# Example of proof

---

**Prove**  $\phi \vee \neg\phi$

1.  $((\phi \vee \phi) \rightarrow \phi) \rightarrow (\neg\phi \vee (\phi \vee \phi) \rightarrow \neg\phi \vee \phi)$  [Instance of (4)]
2.  $\phi \vee \phi \rightarrow \phi$  [Instance of (1)]
3.  $\neg\phi \vee (\phi \vee \phi) \rightarrow (\neg\phi \vee \phi)$  [1., 2., and MP]
- 3'.  $= (\phi \rightarrow (\phi \vee \phi)) \rightarrow (\neg\phi \vee \phi)$  [3 and definition of  $\rightarrow$ ]
4.  $\phi \rightarrow \phi \vee \phi$  [Instance of (2)]
5.  $\neg\phi \vee \phi$  [3., 4. and MP]
6.  $(\neg\phi \vee \phi) \rightarrow (\phi \vee \neg\phi)$  [Instance of (3)]
7.  $\phi \vee \neg\phi$  [5., 6. and MP]

# Soundness

---

$\Gamma$  is called **sound**  $:\Leftrightarrow$

$$\frac{F_1 \dots F_n}{F} \in \Gamma \Rightarrow F_1, \dots, F_n \models F$$

$\Gamma$  sound iff If  $N \vdash_{\Gamma} F$  then  $N \models F$ .

**Theorem.** The Hilbert deductive system is sound.

**Proof:** The proof for propositional logic is by induction on the length of the formal proof of  $F$  from  $N$ .

Proof of length 0: show that all axioms are valid

Induction step  $n \mapsto n + 1$ : uses the definition of a proof.

It is sufficient to show that  $(\phi \wedge (\phi \rightarrow \phi')) \models \phi'$ .

# Completeness

---

$\Gamma$  is called **complete**  $:\Leftrightarrow$

$$N \models F \Rightarrow N \vdash_{\Gamma} F$$

**Theorem.** The Hilbert deductive system is complete.

The very first proof for propositional logic was given by Bernays (a student of Hilbert).

# Completeness: Sketch of Bernay's proof

---

Every formula is interderivable with its conjunctive normal form.

A conjunction is provable if and only if each of its conjuncts is provable.

A disjunction of propositional variables and negations of propositional variables is provable if and only if it contains a propositional variable and its negation. Conversely, every such disjunction is provable.

So, a formula is provable if and only if every conjunct in its conjunctive normal form contains a variable and its negation.

Now suppose that  $\phi$  is a valid but underivable formula.

Its conjunctive normal form  $CNF(\phi)$  is also underivable, so it must contain a conjunct  $\phi'$  where every propositional variable occurs only negated or unnegated but not both.

If  $\phi$  was added as a new axiom (so that  $\models \phi$  implies soundness of the new deductive system), then  $CNF(\phi)$  and  $\phi'$  would also be derivable.

By substituting  $X$  for every unnegated variable and  $\neg X$  for every negated variable in  $\phi'$ , we would obtain  $X$  as a derivable formula (after some simplification), and the system would be inconsistent, which is the desired contradiction.



# Sequent calculus for propositional logic

---

Sequent Calculus based on notion of sequent

$$\underbrace{\psi_1, \dots, \psi_m}_{\text{Antecedent}} \Rightarrow \underbrace{\phi_1, \dots, \phi_n}_{\text{Succedent}}$$

Has same semantics as

$$\models \psi_1 \wedge \dots \wedge \psi_m \rightarrow (\phi_1 \vee \dots \vee \phi_n)$$

$$\{\psi_1, \dots, \psi_m\} \models \phi_1 \vee \dots \vee \phi_n$$

# Notation for Sequents

---

$$\underbrace{\psi_1, \dots, \psi_m}_{\text{Antecedent}} \Rightarrow \underbrace{\phi_1, \dots, \phi_n}_{\text{Succedent}}$$

Consider antecedent/succedent as sets of formulae (may be empty)

# Notation for Sequents

---

$$\underbrace{\psi_1, \dots, \psi_m}_{\text{Antecedent}} \Rightarrow \underbrace{\phi_1, \dots, \phi_n}_{\text{Succedent}}$$

Consider antecedent/succedent as sets of formulae (may be empty)

## Conventions:

- empty antecedent = empty conjunction =  $\top$
- empty succedent = empty disjunction =  $\perp$

# Notation for Sequents

---

$$\underbrace{\psi_1, \dots, \psi_m}_{\text{Antecedent}} \Rightarrow \underbrace{\phi_1, \dots, \phi_n}_{\text{Succedent}}$$

Consider antecedent/succedent as sets of formulae (may be empty)

## Conventions:

- empty antecedent = empty conjunction =  $\top$
- empty succedent = empty disjunction =  $\perp$

## Alternative notation:

$$\psi_1, \dots, \psi_m \vdash \phi_1, \dots, \phi_n$$

Not used here because of the risk of potential confusion with the provability relation

# Notation for Sequents

---

$$\underbrace{\psi_1, \dots, \psi_m}_{\text{Antecedent}} \Rightarrow \underbrace{\phi_1, \dots, \phi_n}_{\text{Succedent}}$$

Consider antecedent/succedent as sets of formulas, may be empty

## Schema Variables:

$\phi, \psi, \dots$  match formulas,  $\Gamma, \Delta, \dots$  match sets of formulas

Characterize infinitely many sequents with a single schematic sequent:

**Example:**  $\Gamma \Rightarrow \Delta, \phi \wedge \psi$

Matches any sequent with occurrence of conjunction in succedent

We call  $\phi \wedge \psi$  **main formula** and  $\Gamma, \Delta$  **side formulae** of sequent.

# Sequent Calculus Rules of Propositional Logic

---

Write syntactic transformation schema for sequents that reflects semantics of connectives as closely as possible

$$\text{Rule Name } \frac{\overbrace{\Gamma_1 \Rightarrow \Delta_1 \dots \Gamma_n \Rightarrow \Delta_n}^{\text{premises}}}{\underbrace{\Gamma \Rightarrow \Delta}_{\text{conclusion}}} .$$

# Sequent Calculus Rules of Propositional Logic

---

Write syntactic transformation schema for sequents that reflects semantics of connectives as closely as possible

$$\text{Rule Name } \frac{\overbrace{\Gamma_1 \Rightarrow \Delta_1 \dots \Gamma_n \Rightarrow \Delta_n}^{\text{premises}}}{\underbrace{\Gamma \Rightarrow \Delta}_{\text{conclusion}}} .$$

**Example:**

$$\text{andRight } \frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \wedge \psi, \Delta} .$$

# Sequent Calculus Rules of Propositional Logic

Write syntactic transformation schema for sequents that reflects semantics of connectives as closely as possible

$$\text{Rule Name } \frac{\overbrace{\Gamma_1 \Rightarrow \Delta_1 \dots \Gamma_n \Rightarrow \Delta_n}^{\text{premises}}}{\underbrace{\Gamma \Rightarrow \Delta}_{\text{conclusion}}} .$$

## Example:

$$\text{andRight } \frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \wedge \psi, \Delta} .$$

Informal meaning:

In order to prove that  $\Gamma$  entails  $(\phi \wedge \psi) \vee \Delta$  we need to prove that:

$\Gamma$  entails  $\phi \vee \Delta$  and

$\Gamma$  entails  $\psi \vee \Delta$



# Sequent Calculus Rules of Propositional Logic

---

Write syntactic transformation schema for sequents that reflects semantics of connectives as closely as possible

$$\text{Rule Name } \frac{\overbrace{\Gamma_1 \Rightarrow \Delta_1 \dots \Gamma_n \Rightarrow \Delta_n}^{\text{premises}}}{\underbrace{\Gamma \Rightarrow \Delta}_{\text{conclusion}}} .$$

**Example:**

$$\text{andRight } \frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \wedge \psi, \Delta} .$$

**Sound rule (essential):** If  $\models (\Gamma_1 \rightarrow \Delta_1)$  and  $\dots \models (\Gamma_n \rightarrow \Delta_n)$  then  $\models (\Gamma \rightarrow \Delta)$

# Sequent Calculus Rules of Propositional Logic

Write syntactic transformation schema for sequents that reflects semantics of connectives as closely as possible

$$\text{Rule Name } \frac{\overbrace{\Gamma_1 \Rightarrow \Delta_1 \dots \Gamma_n \Rightarrow \Delta_n}^{\text{premises}}}{\underbrace{\Gamma \Rightarrow \Delta}_{\text{conclusion}}} .$$

**Example:**

$$\text{andRight } \frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \wedge \psi, \Delta} .$$

**Sound rule (essential):** If  $\models (\Gamma_1 \rightarrow \Delta_1)$  and ... and  $\models (\Gamma_n \rightarrow \Delta_n)$  then  $\models (\Gamma \rightarrow \Delta)$

**Complete rule (desirable):** If  $\models (\Gamma \rightarrow \Delta)$  then  $\models (\Gamma_1 \rightarrow \Delta_1), \dots, \models (\Gamma_n \rightarrow \Delta_n)$

# Rules of Propositional Sequent Calculus

---

main	left side (antecedent)	right side (succedent)
not	$\frac{\Gamma \Rightarrow \phi, \Delta}{\Gamma, \neg \phi \Rightarrow \Delta}$	$\frac{\Gamma, \phi \Rightarrow \Delta}{\Gamma \Rightarrow \neg \phi, \Delta}$

# Rules of Propositional Sequent Calculus

---

main	left side (antecedent)	right side (succedent)
not	$\frac{\Gamma \Rightarrow \phi, \Delta}{\Gamma, \neg \phi \Rightarrow \Delta}$	$\frac{\Gamma, \phi \Rightarrow \Delta}{\Gamma \Rightarrow \neg \phi, \Delta}$
and	$\frac{\Gamma, \phi, \psi \Rightarrow \Delta}{\Gamma, \phi \wedge \psi \Rightarrow \Delta}$	$\frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \wedge \psi, \Delta}$

# Rules of Propositional Sequent Calculus

---

main	left side (antecedent)	right side (succedent)
not	$\frac{\Gamma \Rightarrow \phi, \Delta}{\Gamma, \neg \phi \Rightarrow \Delta}$	$\frac{\Gamma, \phi \Rightarrow \Delta}{\Gamma \Rightarrow \neg \phi, \Delta}$
and	$\frac{\Gamma, \phi, \psi \Rightarrow \Delta}{\Gamma, \phi \wedge \psi \Rightarrow \Delta}$	$\frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \wedge \psi, \Delta}$
or	$\frac{\Gamma, \phi \Rightarrow \Delta \quad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \phi \vee \psi \Rightarrow \Delta}$	$\frac{\Gamma \Rightarrow \phi, \psi, \Delta}{\Gamma \Rightarrow \phi \vee \psi, \Delta}$

# Rules of Propositional Sequent Calculus

---

main	left side (antecedent)	right side (succedent)
not	$\frac{\Gamma \Rightarrow \phi, \Delta}{\Gamma, \neg \phi \Rightarrow \Delta}$	$\frac{\Gamma, \phi \Rightarrow \Delta}{\Gamma \Rightarrow \neg \phi, \Delta}$
and	$\frac{\Gamma, \phi, \psi \Rightarrow \Delta}{\Gamma, \phi \wedge \psi \Rightarrow \Delta}$	$\frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \wedge \psi, \Delta}$
or	$\frac{\Gamma, \phi \Rightarrow \Delta \quad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \phi \vee \psi \Rightarrow \Delta}$	$\frac{\Gamma \Rightarrow \phi, \psi, \Delta}{\Gamma \Rightarrow \phi \vee \psi, \Delta}$
imp	$\frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \phi \rightarrow \psi \Rightarrow \Delta}$	$\frac{\Gamma, \phi \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \rightarrow \psi, \Delta}$

# Rules of Propositional Sequent Calculus

main	left side (antecedent)	right side (succedent)
not	$\frac{\Gamma \Rightarrow \phi, \Delta}{\Gamma, \neg \phi \Rightarrow \Delta}$	$\frac{\Gamma, \phi \Rightarrow \Delta}{\Gamma \Rightarrow \neg \phi, \Delta}$
and	$\frac{\Gamma, \phi, \psi \Rightarrow \Delta}{\Gamma, \phi \wedge \psi \Rightarrow \Delta}$	$\frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \wedge \psi, \Delta}$
or	$\frac{\Gamma, \phi \Rightarrow \Delta \quad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \phi \vee \psi \Rightarrow \Delta}$	$\frac{\Gamma \Rightarrow \phi, \psi, \Delta}{\Gamma \Rightarrow \phi \vee \psi, \Delta}$
imp	$\frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \phi \rightarrow \psi \Rightarrow \Delta}$	$\frac{\Gamma, \phi \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \rightarrow \psi, \Delta}$

close  $\overline{\Gamma, \phi \Rightarrow \phi, \Delta}$     true  $\overline{\Gamma \Rightarrow \text{true}, \Delta}$     false  $\overline{\Gamma, \text{false} \Rightarrow \Delta}$

# Justification of Rules

---

Compute rules by applying semantic definitions



# Justification of Rules

---

Compute rules by applying semantic definitions

$$\text{orRight} \quad \frac{\Gamma \Rightarrow \phi, \psi, \Delta}{\Gamma \Rightarrow \phi \vee \psi, \Delta}$$

Follows directly from semantics of sequents

# Justification of Rules

---

Compute rules by applying semantic definitions

$$\text{orRight} \quad \frac{\Gamma \Rightarrow \phi, \psi, \Delta}{\Gamma \Rightarrow \phi \vee \psi, \Delta}$$

Follows directly from semantics of sequents

$$\text{andRight} \quad \frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \wedge \psi, \Delta}$$

$$\models \Gamma \rightarrow (\phi \wedge \psi) \vee \Delta \text{ iff } (\models \Gamma \rightarrow \phi \vee \Delta \text{ and } \models \Gamma \rightarrow \psi \vee \Delta)$$

# Sequent Calculus Proofs

---

Goal to prove:  $\mathcal{G} = (\psi_1, \dots, \psi_m \Rightarrow \phi_1, \dots, \phi_n)$

# Sequent Calculus Proofs

---

Goal to prove:  $\mathcal{G} = (\psi_1, \dots, \psi_m \Rightarrow \phi_1, \dots, \phi_n)$

- find rule  $R$  whose conclusion matches  $\mathcal{G}$
- instantiate  $R$  such that conclusion identical to  $\mathcal{G}$
- recursively find proofs for resulting premisses  $\mathcal{G}_1, \dots, \mathcal{G}_r$
- tree structure with goal as root
- close proof branch when rule without premisses encountered

# A Simple Proof

---

$$\Rightarrow (p \wedge (p \rightarrow q)) \rightarrow q$$

# A Simple Proof

---

$$\frac{p \wedge (p \rightarrow q) \Rightarrow q}{\Rightarrow (p \wedge (p \rightarrow q)) \rightarrow q} \quad (\text{imp}), \text{ right}$$

# A Simple Proof

---

$$\frac{\frac{p, (p \rightarrow q) \Rightarrow q}{p \wedge (p \rightarrow q) \Rightarrow q}}{\Rightarrow (p \wedge (p \rightarrow q)) \rightarrow q}$$

(and), left  
(imp), right

# A Simple Proof

---

$$\frac{\frac{\frac{p \Rightarrow q, p \quad p, q \Rightarrow q}{p, (p \rightarrow q) \Rightarrow q}}{p \wedge (p \rightarrow q) \Rightarrow q}}{\Rightarrow (p \wedge (p \rightarrow q)) \rightarrow q}$$

(imp), left  
(and), left  
(imp), right



# A Simple Proof

---

$\frac{\frac{}{p \Rightarrow q, p} \quad \frac{}{p, q \Rightarrow q}}{p, (p \rightarrow q) \Rightarrow q}$	close, close (imp), left
$\frac{p, (p \rightarrow q) \Rightarrow q}{p \wedge (p \rightarrow q) \Rightarrow q}$	(and), left
$\frac{p \wedge (p \rightarrow q) \Rightarrow q}{\Rightarrow (p \wedge (p \rightarrow q)) \rightarrow q}$	(imp), right

# A Simple Proof

---

$$\begin{array}{c} \text{close } * \\ \hline p \Rightarrow q, p \end{array} \quad \begin{array}{c} \text{close } * \\ \hline p, q \Rightarrow q \end{array}$$
$$\hline p, (p \rightarrow q) \Rightarrow q$$
$$\hline p \wedge (p \rightarrow q) \Rightarrow q$$
$$\hline \Rightarrow (p \wedge (p \rightarrow q)) \rightarrow q$$

A proof is closed iff all its branches are closed

# Soundness, Completeness, Termination

---

**Soundness and completeness** can be proved for every rule:

**Sound:** If  $\models (\Gamma_1 \rightarrow \Delta_1)$  and  $\dots$  and  $\models (\Gamma_n \rightarrow \Delta_n)$  then  $\models (\Gamma \rightarrow \Delta)$

**Complete:** If  $\models (\Gamma \rightarrow \Delta)$  then  $\models (\Gamma_1 \rightarrow \Delta_1), \dots, \models (\Gamma_n \rightarrow \Delta_n)$

# Soundness, Completeness

---

**Soundness and completeness** can be proved for every rule:

**Sound:** If  $\models (\Gamma_1 \rightarrow \Delta_1)$  and  $\dots$  and  $\models (\Gamma_n \rightarrow \Delta_n)$  then  $\models (\Gamma \rightarrow \Delta)$

**Complete:** If  $\models (\Gamma \rightarrow \Delta)$  then  $\models (\Gamma_1 \rightarrow \Delta_1), \dots, \models (\Gamma_n \rightarrow \Delta_n)$

**Consequence:** The following are equivalent:

(1)  $\Gamma \models \Delta$

(2) there exists a proof in the sequent calculus for  $\Gamma \Rightarrow \Delta$ .

# The Propositional Resolution Calculus

---

Resolution inference rule:

$$\frac{C \vee A \quad \neg A \vee D}{C \vee D}$$

Terminology:  $C \vee D$ : **resolvent**;  $A$ : **resolved atom**

(Positive) factorisation inference rule:

$$\frac{C \vee A \vee A}{C \vee A}$$

# The Resolution Calculus *Res*

---

These are **schematic inference rules**; for each substitution of the **schematic variables**  $C$ ,  $D$ , and  $A$ , respectively, by propositional clauses and atoms we obtain an inference rule.

As “ $\vee$ ” is considered associative and commutative, we assume that  $A$  and  $\neg A$  can occur anywhere in their respective clauses.

## Sample Refutation

---

1.  $\neg P \vee \neg P \vee Q$  (given)
2.  $P \vee Q$  (given)
3.  $\neg R \vee \neg Q$  (given)
4.  $R$  (given)
5.  $\neg P \vee Q \vee Q$  (Res. 2. into 1.)
6.  $\neg P \vee Q$  (Fact. 5.)
7.  $Q \vee Q$  (Res. 2. into 6.)
8.  $Q$  (Fact. 7.)
9.  $\neg R$  (Res. 8. into 3.)
10.  $\perp$  (Res. 4. into 9.)

# Resolution with Implicit Factorization *RIF*

---

$$\frac{C \vee A \vee \dots \vee A \quad \neg A \vee D}{C \vee D}$$

1.  $\neg P \vee \neg P \vee Q$  (given)
2.  $P \vee Q$  (given)
3.  $\neg R \vee \neg Q$  (given)
4.  $R$  (given)
5.  $\neg P \vee Q \vee Q$  (Res. 2. into 1.)
6.  $Q \vee Q \vee Q$  (Res. 2. into 5.)
7.  $\neg R$  (Res. 6. into 3.)
8.  $\perp$  (Res. 4. into 7.)



# Soundness and Completeness

---

**Theorem 1.6.** Propositional resolution is sound.

for both the resolution rule and the positive factorization rule  
the conclusion of the inference is entailed by the premises.

**Theorem 1.7.** Propositional resolution is refutationally complete.

If  $N \models \perp$  we can deduce  $\perp$  starting from  $N$  and using  
the inference rules of the propositional resolution calculus.