

Propositional Logic

Basics

Syntax of propositional logic

Definition

An **atomic formula** (or **atom**) has the form A_i where $i = 1, 2, 3, \dots$

Formulas are defined inductively:

- ▶ \perp (“False”) and \top (“True”) are formulas
- ▶ All atomic formulas are formulas
- ▶ For all formulas F , $\neg F$ is a formula.
- ▶ For all formulas F and G , $(F \circ G)$ is a formula, where $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$

\neg	is called	negation
\wedge	is called	conjunction
\vee	is called	disjunction
\rightarrow	is called	implication
\leftrightarrow	is called	bi-implication

Parentheses

Precedence of logical operators in decreasing order:

$$\neg \quad \wedge \quad \vee \quad \rightarrow \quad \leftrightarrow$$

Operators with higher precedence bind more strongly.

Example

Instead of $(A \rightarrow ((B \wedge \neg(C \vee D)) \vee E))$

we can write $A \rightarrow B \wedge \neg(C \vee D) \vee E$.

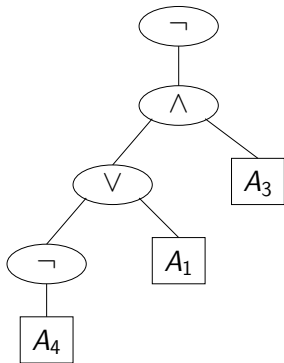
Outermost parentheses can be dropped.

Syntax tree of a formula

Every formula can be represented by a syntax tree.

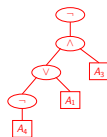
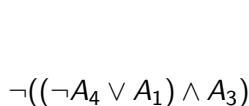
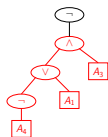
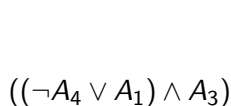
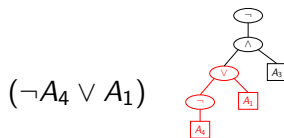
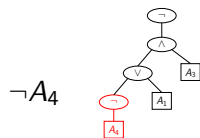
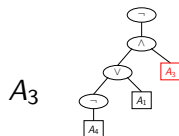
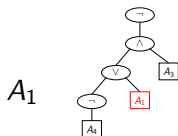
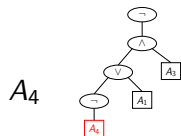
Example

$$F = \neg((\neg A_4 \vee A_1) \wedge A_3)$$



Subformulas

The **subformulas** of a formula are the formulas corresponding to the subtrees of its syntax tree.



Induction on formulas

Proof by induction on the structure of a formula:

In order to prove some property $\mathcal{P}(F)$ for all formulas F it suffices to prove the following:

- ▶ Base cases:
prove $\mathcal{P}(\perp)$, prove $\mathcal{P}(\top)$, and prove $\mathcal{P}(A_i)$ for all atoms A_i
- ▶ Induction step for \neg :
prove $\mathcal{P}(\neg F)$ under the induction hypothesis $\mathcal{P}(F)$
- ▶ Induction step for all $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$:
prove $\mathcal{P}(F \circ G)$ under the induction hypotheses $\mathcal{P}(F)$ and $\mathcal{P}(G)$

Operators that are merely abbreviations need not be considered!

Semantics of propositional logic (I)

The elements of the set $\{0, 1\}$ are called **truth values**.
(You may call 0 “false” and 1 “true”)

An **assignment** is a function $\mathcal{A} : Atoms \rightarrow \{0, 1\}$
where *Atoms* is the set of all atoms.

We extend \mathcal{A} to a function $\hat{\mathcal{A}} : Formulas \rightarrow \{0, 1\}$

Semantics of propositional logic (II)

$$\hat{\mathcal{A}}(A_i) = \mathcal{A}(A_i)$$

$$\hat{\mathcal{A}}(\neg F) = \begin{cases} 1 & \text{if } \hat{\mathcal{A}}(F) = 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\hat{\mathcal{A}}(F \wedge G) = \begin{cases} 1 & \text{if } \hat{\mathcal{A}}(F) = 1 \text{ and } \hat{\mathcal{A}}(G) = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\hat{\mathcal{A}}(F \vee G) = \begin{cases} 1 & \text{if } \hat{\mathcal{A}}(F) = 1 \text{ or } \hat{\mathcal{A}}(G) = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\hat{\mathcal{A}}(F \rightarrow G) = \begin{cases} 1 & \text{if } \hat{\mathcal{A}}(F) = 0 \text{ or } \hat{\mathcal{A}}(G) = 1 \\ 0 & \text{otherwise} \end{cases}$$

Instead of $\hat{\mathcal{A}}$ we simply write \mathcal{A}

Using arithmetic: $\mathcal{A}(F \wedge G) = \min(\mathcal{A}(F), \mathcal{A}(G))$

$\mathcal{A}(F \vee G) = \max(\mathcal{A}(F), \mathcal{A}(G))$

Abbreviations

$A, B, C,$
 $P, Q, R,$ or ... instead of $A_1, A_2, A_3 \dots$

$$F_1 \leftrightarrow F_2 \text{ abbreviates } (F_1 \wedge F_2) \vee (\neg F_1 \wedge \neg F_2)$$
$$\bigvee_{i=1}^n F_i \text{ abbreviates } (\dots ((F_1 \vee F_2) \vee F_3) \vee \dots \vee F_n)$$
$$\bigwedge_{i=1}^n F_i \text{ abbreviates } (\dots ((F_1 \wedge F_2) \wedge F_3) \wedge \dots \wedge F_n)$$

Special cases:

$$\bigvee_{i=1}^0 F_i = \bigvee \emptyset = \perp \qquad \bigwedge_{i=1}^0 F_i = \bigwedge \emptyset = \top$$

Truth tables (I)

We can compute \hat{A} with the help of **truth tables**.

\neg	A	A	\vee	B	A	\wedge	B
1	0	0	0	0	0	0	0
0	1	0	1	1	0	0	1
		1	1	0	1	0	0
		1	1	1	1	1	1

A	\rightarrow	B	A	\leftrightarrow	B
0	1	0	0	1	0
0	1	1	0	0	1
1	0	0	1	0	0
1	1	1	1	1	1

Coincidence Lemma

Lemma

Let \mathcal{A}_1 and \mathcal{A}_2 be two assignments.

*If $\mathcal{A}_1(A_i) = \mathcal{A}_2(A_i)$ for all atoms A_i in some formula F ,
then $\mathcal{A}_1(F) = \mathcal{A}_2(F)$.*

Proof.

Exercise.



Models

If $\mathcal{A}(F) = 1$ then we write $\mathcal{A} \models F$
and say F is true under \mathcal{A}
or \mathcal{A} is a model of F

If $\mathcal{A}(F) = 0$ then we write $\mathcal{A} \not\models F$
and say F is false under \mathcal{A}
or \mathcal{A} is not a model of F

Validity and satisfiability

Definition (Validity)

A formula F is **valid** (or a **tautology**) if every assignment is a model of F .

We write $\models F$ if F is valid, and $\not\models F$ otherwise.

Definition (Satisfiability)

A formula F is **satisfiable** if it has at least one model; otherwise F is **unsatisfiable**.

A (finite or infinite!) set of formulas S is **satisfiable** if there is an assignment that is a model of every formula in S .

Exercise

	Valid	Satisfiable	Unsatisfiable
A			
$A \vee B$			
$A \vee \neg A$			
$A \wedge \neg A$			
$A \rightarrow \neg A$			
$A \rightarrow (B \rightarrow A)$			
$A \rightarrow (A \rightarrow B)$			
$A \leftrightarrow \neg A$			

Exercise

Which of the following statements are true?

	Y	C.ex.
If F is valid then F is satisfiable		
If F is satisfiable then $\neg F$ is satisfiable		
If F is valid then $\neg F$ is unsatisfiable		
If F is unsatisfiable then $\neg F$ is unsatisfiable		

Mirroring principle

all propositional formulas

valid formulas G	satisfiable but not valid formulas F $\neg F$	unsatisfiable formulas $\neg G$
------------------------------	--	---

Consequence (aka entailment)

Definition

A formula G is a (semantic) consequence of a set of formulas M if every model \mathcal{A} of all $F \in M$ is also a model of G .

We also say that M entails G and write $M \models G$.

In a nutshell:

“Every model of M is a model of G .”

Example

$A \vee B, A \rightarrow B, B \wedge R \rightarrow \neg A, R \models (R \wedge \neg A) \wedge B$

Consequence

Example

$$\underbrace{A \vee B, A \rightarrow B, B \wedge R \rightarrow \neg A, R}_M \models (R \wedge \neg A) \wedge B$$

Proof:

Assume $\mathcal{A} \models F$ for all $F \in M$.

We need to prove $\mathcal{A} \models (R \wedge \neg A) \wedge B$.

It suffices to prove $\mathcal{A} \models R$, $\mathcal{A} \models \neg A$, and $\mathcal{A} \models B$

- ▶ $\mathcal{A} \models R$ is immediate.
- ▶ $\mathcal{A} \models B$ follows from $\mathcal{A} \models A \vee B$ and $\mathcal{A} \models A \rightarrow B$:

Proof by cases:

If $\mathcal{A}(A) = 0$ then $\mathcal{A}(B) = 1$ because $\mathcal{A} \models A \vee B$

If $\mathcal{A}(A) = 1$ then $\mathcal{A}(B) = 1$ because $\mathcal{A} \models A \rightarrow B$

- ▶ $\mathcal{A} \models \neg A$ follows from $\mathcal{A} \models B$ and $\mathcal{A} \models R$.

Exercise

M	F	$M \models F?$
A	$A \vee B$	
A	$A \wedge B$	
A, B	$A \vee B$	
A, B	$A \wedge B$	
$A \wedge B$	A	
$A \vee B$	A	
$A, A \rightarrow B$	B	

Consequence

Exercise

The following statements are equivalent:

1. $F_1, \dots, F_k \models G$
2. $\models (\bigwedge_{i=1}^k F_i) \rightarrow G$

Proof of “if $F_1, \dots, F_k \models G$ then $\models \underbrace{(\bigwedge_{i=1}^k F_i) \rightarrow G}_H$ ”.

Assume $F_1, \dots, F_k \models G$.

We need to prove $\models H$, i.e. $\mathcal{A}(H) = 1$ for all \mathcal{A} .

We pick an arbitrary \mathcal{A} and show $\mathcal{A}(H) = 1$.

Proof by cases: either $\mathcal{A}(\bigwedge F_i) = 0$ or $\mathcal{A}(\bigwedge F_i) = 1$.

- ▶ $\mathcal{A}(\bigwedge F_i) = 0$: Then $\mathcal{A}(H) = 1$ because $H = \bigwedge F_i \rightarrow G$.
- ▶ $\mathcal{A}(\bigwedge F_i) = 1$: Then $\mathcal{A}(F_i) = 1$ for all i .
Therefore \mathcal{A} is a model of F_1, \dots, F_k .
Therefore $\mathcal{A} \models G$ because $F_1, \dots, F_k \models G$.

Validity and satisfiability

Exercise

The following statements are equivalent:

1. $F \rightarrow G$ is valid.
2. $F \wedge \neg G$ is unsatisfiable.

Exercise

Let M be a set of formulas, and let F and G be formulas.
Which of the following statements hold?

	Y/N	C.ex.
If F satisfiable then $M \models F$.		
If F valid then $M \models F$.		
If $F \in M$ then $M \models F$.		
If $F \models G$ then $\neg F \models \neg G$.		

Notation

Warning: The symbol \models is overloaded:

$$\mathcal{A} \models F$$

$$\models F$$

$$M \models F$$

Convenient variations for set of formulas S :

$$\mathcal{A} \models S \text{ means that for all } F \in S, \mathcal{A} \models F$$

$$\models S \text{ means that for all } F \in S, \models F$$

$$M \models S \text{ means that for all } F \in S, M \models F$$

Propositional Logic Equivalences

Equivalence

Definition (Equivalence)

Two formulas F and G are (semantically) equivalent if $\mathcal{A}(F) = \mathcal{A}(G)$ for every assignment \mathcal{A} .

We write $F \equiv G$ to denote that F and G are equivalent.

Exercise

Which of the following equivalences hold?

$$(A \wedge (A \vee B)) \equiv A$$

$$(A \wedge (B \vee C)) \equiv ((A \wedge B) \vee C)$$

$$(A \rightarrow (B \rightarrow C)) \equiv ((A \rightarrow B) \rightarrow C)$$

$$(A \rightarrow (B \rightarrow C)) \equiv ((A \wedge B) \rightarrow C)$$

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

$$(A \rightarrow B) \equiv (\neg A \rightarrow \neg B)$$

$$(A \leftrightarrow (B \leftrightarrow C)) \equiv ((A \leftrightarrow B) \leftrightarrow C)$$

Observation

The following connections hold:

$$\begin{aligned} \models F \rightarrow G & \text{ iff } F \models G \\ \models F \leftrightarrow G & \text{ iff } F \equiv G \end{aligned}$$

NB: “iff” means “if and only if”

Reductions between problems (I)

- ▶ **Validity** to **Unsatisfiability**:

F valid iff $? \neg F$ unsatisfiable

- ▶ **Unsatisfiability** to **Validity**:

F unsatisfiable iff $? \neg F$ valid

- ▶ **Validity** to **Consequence**:

F valid iff $? \models ? \top \models F$

- ▶ **Consequence** to **Validity**:

$F \models G$ iff $? F \rightarrow G$ valid

- ▶ **Validity** to **Equivalence**:

F valid iff $? \equiv ? F \equiv \top$

- ▶ **Equivalence** to **Validity**:

$F \equiv G$ iff $? F \leftrightarrow G$ valid

Properties of semantic equivalence

- ▶ Semantic equivalence is an **equivalence relation** between formulas.
- ▶ Semantic equivalence is **closed under operators**:

If $F_1 \equiv F_2$ and $G_1 \equiv G_2$

then $\neg F_1 \equiv \neg F_2$ and

$(F_1 \circ G_1) \equiv (F_2 \circ G_2)$ for $\circ \in \{\vee, \wedge, \rightarrow, \leftrightarrow\}$

Equivalence relation + Closure under Operations

=

Congruence relation

Replacement theorem

Theorem

Let $F \equiv G$. Let H be a formula with an occurrence of F as a subformula. Let H' be the result of replacing an arbitrary occurrence of F in H by G . Then $H \equiv H'$.

Proof by induction on the structure of H .

We consider only the case $H = \neg H_0$.

Two cases: either $F = H$ or F is a subformula of H_0 .

- ▶ $F = H$: Then $H' = G$ and thus $H = F \equiv G = H'$.
- ▶ F is a subformula of H_0 .

Let H'_0 be the result of replacing F by G in H_0 .

IH: $H_0 \equiv H'_0$

Thus $H = \neg H_0 \equiv \neg H'_0 = H'$.

Equivalences (I)

Theorem

$$(F \wedge F) \equiv F$$

$$(F \vee F) \equiv F \quad (\text{Idempotence})$$

$$(F \wedge G) \equiv (G \wedge F)$$

$$(F \vee G) \equiv (G \vee F) \quad (\text{Commutativity})$$

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

$$((F \vee G) \vee H) \equiv (F \vee (G \vee H)) \quad (\text{Associativity})$$

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

$$(F \vee (F \wedge G)) \equiv F \quad (\text{Absorption})$$

Equivalences (II)

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

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

$$\neg\neg F \equiv F$$

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

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

$$\neg\top \equiv \perp$$

$$\neg\perp \equiv \top$$

$$(\top \vee G) \equiv \top$$

$$(\top \wedge G) \equiv G$$

$$(\perp \vee G) \equiv G$$

$$(\perp \wedge G) \equiv \perp$$

(Distributivity)

(Double negation)

(deMorgan's Laws)

Warning

The symbols \models and \equiv are **not** operators
in the language of propositional logic
but part of the meta-language for talking about logic.

Examples:

$\mathcal{A} \models F$ and $F \equiv G$ are not propositional formulas.
 $(\mathcal{A} \models F) \equiv G$ and $(F \equiv G) \leftrightarrow (G \equiv F)$ are nonsense.

Propositional Logic

Normal Forms

Abbreviations

Until further notice:

$F_1 \rightarrow F_2$ abbreviates $\neg F_1 \vee F_2$

$F_1 \leftrightarrow F_2$ abbreviates $(F_1 \wedge F_2) \vee (\neg F_1 \wedge \neg F_2)$

\top abbreviates $A_1 \vee \neg A_1$

\perp abbreviates $A_1 \wedge \neg A_1$

Literals

Definition

A **literal** is an atom or the negation of an atom.
In the former case the literal is **positive**,
in the latter case it is **negative**.

Negation Normal Form (NNF)

Definition

A formula is in **negation normal form (NNF)** if negation (\neg) occurs only directly in front of atoms.

Example

In NNF: $\neg A \wedge \neg B$

Not in NNF: $\neg(A \vee B)$

Transformation into NNF

Any formula can be transformed into an equivalent formula in NNF by pushing \neg inwards. Apply the following equivalences from left to right as long as possible:

$$\begin{aligned}\neg\neg F &\equiv F \\ \neg(F \wedge G) &\equiv (\neg F \vee \neg G) \\ \neg(F \vee G) &\equiv (\neg F \wedge \neg G)\end{aligned}$$

Example

$$\begin{aligned}(\neg(A \wedge \neg B) \wedge C) &\equiv ((\neg A \vee \neg\neg B) \wedge C) \equiv ((\neg A \vee B) \wedge C) \\ (\text{"}F \equiv G \equiv H\text{" is an abbreviation for "}F \equiv G \text{ and } G \equiv H\text{"})\end{aligned}$$

Does this process always terminate? Is the result unique?

CNF and DNF

Definition

A formula F is in **conjunctive normal form (CNF)** if it is a conjunction of disjunctions of literals:

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

where $L_{i,j} \in \{A_1, A_2, \dots\} \cup \{\neg A_1, \neg A_2, \dots\}$

Definition

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

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

where $L_{i,j} \in \{A_1, A_2, \dots\} \cup \{\neg A_1, \neg A_2, \dots\}$

Transformation into CNF and DNF

Any formula can be transformed into an equivalent formula in CNF or DNF in two steps:

1. Transform the initial formula into its NNF
2. Transform the NNF into CNF or DNF:
 - ▶ Transformation into CNF. Apply the following equivalences from left to right as long as possible:

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

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

- ▶ Transformation into DNF. Apply the following equivalences from left to right as long as possible:

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

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

Termination

Why does the transformation into NNF and CNF terminate?

Challenge Question: Find a weight function $w :: \text{formula} \rightarrow \mathbb{N}$ such that $w(l.h.s.) > w(r.h.s.)$ for the equivalences

$$\begin{aligned}\neg\neg F &\equiv F \\ \neg(F \wedge G) &\equiv (\neg F \vee \neg G) \\ \neg(F \vee G) &\equiv (\neg F \wedge \neg G) \\ (F \vee (G \wedge H)) &\equiv ((F \vee G) \wedge (F \vee H)) \\ ((F \wedge G) \vee H) &\equiv ((F \vee H) \wedge (G \vee H))\end{aligned}$$

Define w recursively:

$$w(A_i) = \dots$$

$$w(\neg F) = \dots w(F) \dots$$

$$w(F \wedge G) = \dots w(F) \dots w(G) \dots$$

$$w(F \vee G) = \dots w(F) \dots w(G) \dots$$

Complexity considerations

The CNF and DNF of a formula of size n can have size 2^n

Can we do better? Yes, if we do not insist on \equiv .

Definition

Two formulas F and G are **equisatisfiable** if F is satisfiable iff G is satisfiable.

Theorem

For every formula F of size n there is an equisatisfiable CNF formula G of size $O(n)$.

Propositional Logic
Definitional CNF
(Tseytin's transformation)

Definitional CNF

1. The **definitional CNF** of a formula is obtained in 2 steps:

Repeatedly replace a subformula G of the form $\neg A$, $A \wedge B$ or $A \vee B$ (A, B atoms!) by a new atom A' and conjoin $A' \leftrightarrow G$. (This replacement is not applied to the “definitions” $A' \leftrightarrow G$ but only to the (remains of the) original formula.)

2. Translate all the subformulas $A' \leftrightarrow G$ into CNF.

Example

$$\begin{aligned} & \neg(\boxed{A_1 \vee A_2}) \wedge A_3 \\ \rightsquigarrow & \boxed{\neg A_4} \wedge A_3 \wedge (A_4 \leftrightarrow (A_1 \vee A_2)) \\ \rightsquigarrow & \boxed{A_5 \wedge A_3} \wedge (A_4 \leftrightarrow (A_1 \vee A_2)) \wedge (A_5 \leftrightarrow \neg A_4) \\ \rightsquigarrow & A_6 \wedge (A_4 \leftrightarrow (A_1 \vee A_2)) \wedge (A_5 \leftrightarrow \neg A_4) \wedge (A_6 \leftrightarrow (A_5 \wedge A_3)) \\ \rightsquigarrow & A_6 \wedge \text{CNF}(A_4 \leftrightarrow (A_1 \vee A_2)) \wedge \text{CNF}(A_5 \leftrightarrow \neg A_4) \wedge \text{CNF}(A_6 \leftrightarrow (A_5 \wedge A_3)) \end{aligned}$$

Definitional CNF: Complexity

Let the initial formula have size n .

1. Each replacement step increases the size of the formula by a constant.
There are at most as many replacement steps as subformulas, linearly many.
2. The conversion of each $A \leftrightarrow G$ into CNF increases the size by a constant.
There are only linearly many such subformulas.

Thus: the definitional CNF has size $O(n)$, and can be constructed in $O(n)$ time.

Definitional CNF: Correctness — Notation

Definition

The notation $F[G/A]$ denotes the result of replacing **all** occurrences of the atom A in F by G .

We pronounce it as “ F with G for A ”.

Example

$$(A \wedge B)[(A \rightarrow B)/B] = (A \wedge (A \rightarrow B))$$

Definition

The notation $\mathcal{A}[v/A]$ denotes a modified version of \mathcal{A} that maps A to v and behaves like \mathcal{A} otherwise:

$$(\mathcal{A}[v/A])(A_i) = \begin{cases} v & \text{if } A_i = A \\ \mathcal{A}(A_i) & \text{otherwise} \end{cases}$$

Definitional CNF: Correctness — Substitution Lemma

Lemma

$\mathcal{A}(F[G/A]) = \mathcal{A}'(F)$ where $\mathcal{A}' = \mathcal{A}[\mathcal{A}(G)/A]$

Proof by structural induction on F .

► F is an atom:

If $F = A$: $\mathcal{A}(F[G/A]) = \mathcal{A}(G) = \mathcal{A}'(F)$

If $F \neq A$: $\mathcal{A}(F[G/A]) = \mathcal{A}(F) = \mathcal{A}'(F)$

► $F = F_1 \wedge F_2$:

$$\begin{aligned}\mathcal{A}((F_1 \wedge F_2)[G/A]) &= \mathcal{A}(F_1[G/A] \wedge F_2[G/A]) \\ &= \min(\mathcal{A}(F_1[G/A]), \mathcal{A}(F_2[G/A])) \\ &\stackrel{IH}{=} \min(\mathcal{A}'(F_1), \mathcal{A}'(F_2)) \\ &= \mathcal{A}'(F_1 \wedge F_2)\end{aligned}$$

Definitional CNF: Correctness

Each replacement step produces an equisatisfiable formula:

Lemma

Let A be an atom that does not occur in G .

Then $F[G/A]$ is equisatisfiable with $F \wedge (A \leftrightarrow G)$.

Proof Assume $\mathcal{A} \models F[G/A]$ for some assignment \mathcal{A} .

Let $\mathcal{A}' := \mathcal{A}[\mathcal{A}(G)/A]$. We prove $\mathcal{A}' \models F \wedge (A \leftrightarrow G)$.

$\mathcal{A}' \models F$: Substitution Lemma.

$\mathcal{A}' \models (A \leftrightarrow G)$: Because $\mathcal{A}'(A) = \mathcal{A}(G) = \mathcal{A}'(G)$
(by definition of \mathcal{A}' and because A does not occur in G).

Assume $\mathcal{A} \models F \wedge (A \leftrightarrow G)$ for some assignment \mathcal{A} .

We prove $\mathcal{A} \models F[G/A]$, that is, $\mathcal{A}(F[G/A]) = 1$.

We show $\mathcal{A}(F[G/A]) = \mathcal{A}'(F) = \mathcal{A}(F) = 1$ for $\mathcal{A}' := \mathcal{A}[\mathcal{A}(G)/A]$.

$\mathcal{A}(F[G/A]) = \mathcal{A}'(F)$: Substitution Lemma.

$\mathcal{A}'(F) = \mathcal{A}(F)$: From $\mathcal{A} \models (A \leftrightarrow G)$ follows $\mathcal{A}(A) = \mathcal{A}(G)$, and
so $\mathcal{A}' = \mathcal{A}$.

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

Definitional CNF: Correctness

Does $F \wedge (A \leftrightarrow G) \models F[G/A]$ hold?

Does $F[G/A] \models F \wedge (A \leftrightarrow G)$ hold?

Summary

Theorem

For every formula F of size n
there is an *equisatisfiable CNF* formula G of size $O(n)$.

Proof.

Repeated application of the Lemma. □

Similarly it can be shown:

Theorem

For every formula F of size n
there is an *equivalent DNF* formula G of size $O(n)$.

Validity of CNF

Validity of formulas in CNF can be checked in linear time.

A formula in CNF is valid iff all its disjunctions are valid.

A disjunction is valid iff it contains both an atomic A and $\neg A$ as literals.

Example

Valid: $(A \vee \neg A \vee B) \wedge (C \vee \neg C)$

Not valid: $(A \vee \neg A) \wedge (\neg A \vee C)$

Satisfiability of DNF

Satisfiability of formulas in DNF can be checked in linear time.

A formula in DNF is satisfiable iff at least one of its conjunctions is satisfiable. A conjunction is satisfiable iff it does not contain both an atomic A and $\neg A$ as literals.

Example

Satisfiable: $(\neg B \wedge A \wedge B) \vee (\neg A \wedge C)$

Unsatisfiable: $(A \wedge \neg A \wedge B) \vee (C \wedge \neg C)$

Satisfiability/validity of DNF and CNF

Theorem

Satisfiability of formulas in CNF is NP-complete.

Theorem

Validity of formulas in DNF is co-NP-complete.

Standard decision procedure for validity of F :

1. Transform $\neg F$ into an equisat. formula G in def. CNF
2. Apply efficient CNF-based SAT solver to G

Propositional Logic

Horn Formulas

Efficient satisfiability checks

In this and the next slide sets:

- ▶ A very efficient satisfiability check for a special class of formulas in CNF: **Horn formulas**,
- ▶ Efficient satisfiability checks for arbitrary formulas in CNF: **DPLL** and **resolution** (later).

Horn formulas

Definition

A formula F in CNF is a **Horn formula** if every disjunction in F contains at most one positive literal.

Every disjunct of a Horn formula can equivalently be viewed as an implication $K \rightarrow B$ where

- ▶ K is a conjunction of atoms or \top , and
- ▶ B is an atom or \perp .

A	\equiv	$(\top \rightarrow A)$	fact
$(\neg A \vee \neg B \vee C)$	\equiv	$(A \wedge B \rightarrow C)$	rule
$(\neg A \vee B)$	\equiv	$(A \rightarrow B)$	rule
$\neg A$	\equiv	$(A \rightarrow \perp)$	goal
$(\neg A \vee \neg B)$	\equiv	$(A \wedge B \rightarrow \perp)$	goal

Satisfiability check for Horn formulas

Input: a Horn formula F .

Output: Model \mathcal{M} of F or “unsatisfiable”

```
for all atoms  $A_i$  in  $F$  do  $\mathcal{M}(A_i) := 0$ ;  
while  $F$  has a conjunct  $K \rightarrow B$   
    such that  $\mathcal{M}(K) = 1$  and  $\mathcal{M}(B) = 0$   
do  
    if  $B = \perp$  then return “unsatisfiable”  
    else  $\mathcal{M}(B) := 1$   
return  $\mathcal{M}$ 
```

Maximal number of iterations of the while loop:
number of implications in F

Each iteration requires at most $O(|F|)$ steps.

Overall complexity: $O(|F|^2)$

[Algorithm can be improved to $O(|F|)$. See Schöning.]

Correctness of the model building algorithm

Theorem

The algorithm returns a model iff F is satisfiable.

Proof. Invariant: if $\mathcal{M}(A) = 1$, then $\mathcal{A}(A) = 1$ for every atom A and model \mathcal{A} of F .

(a) If “unsatisfiable” then unsatisfiable.

Assume F has model \mathcal{A} but algorithm answers “unsatisfiable”.

Let $(A_{i_1} \wedge \dots \wedge A_{i_k} \rightarrow \perp)$ be the subformula causing “unsatisfiable”.

Since $\mathcal{M}(A_{i_1}) = \dots = \mathcal{M}(A_{i_k}) = 1$, $\mathcal{A}(A_{i_1}) = \dots = \mathcal{A}(A_{i_k}) = 1$.

Then $\mathcal{A}(A_{i_1} \wedge \dots \wedge A_{i_k} \rightarrow \perp) = 0$ and so $\mathcal{A}(F) = 0$, contradiction.

(b) If “ \mathcal{M} ” then $\mathcal{M} \models F$.

After termination with “ \mathcal{M} ”, every conjunct $K \rightarrow B$ of F satisfies $\mathcal{M}(K) = 0$ or $\mathcal{M}(B) = 1$.

Therefore $\mathcal{M}(K \rightarrow B) = 1$ and thus $\mathcal{M} \models F$.

Correctness of the model building algorithm

Corollary

A satisfiable Horn formula has a unique model with a smallest number of true atoms.

Propositional Logic
DPLL: Davis-Putnam-
Logemann-Loveland

Davis–Putnam–Logemann–Loveland

DPLL algorithm:

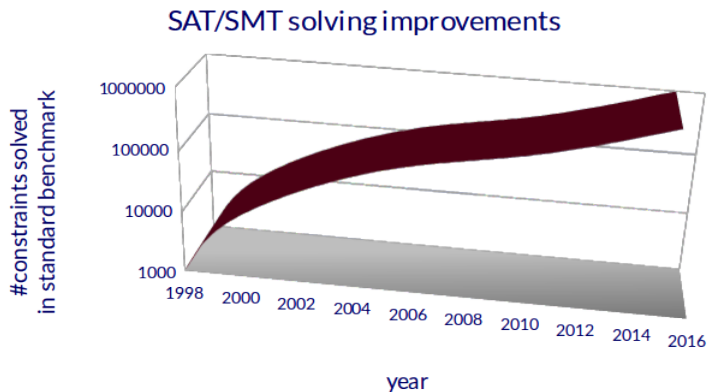
- ▶ combines search and deduction to decide satisfiability
- ▶ underlies most modern SAT solvers
- ▶ is over 50 years old



DPLL-based SAT solvers \geq 1990:

- ▶ clause learning
- ▶ non-chronological backtracking
- ▶ branching heuristics
- ▶ lazy evaluation

Performance increase of SAT solvers



Clause representation of CNF formulas

CNF: $(L_{1,1} \vee \dots \vee L_{1,n_1}) \wedge \dots \wedge (L_{k,1} \vee \dots \vee L_{1,n_k})$

Representation as set of sets of literals:

$$\underbrace{\{L_{1,1}, \dots, L_{1,n_1}\}}_{\text{clause}}, \dots, \{L_{k,1}, \dots, L_{1,n_k}\}$$

Clause = set of literals (disjunction).

Formula in CNF = set of clauses

Degenerate cases:

The empty clause stands for \perp .

The empty set of clauses stands for \top .

The joy of sets

We get “for free”:

▶ **Commutativity:**

$A \vee B \equiv B \vee A$, both represented by $\{A, B\}$

▶ **Associativity:**

$(A \vee B) \vee C \equiv A \vee (B \vee C)$, both represented by $\{A, B, C\}$

▶ **Idempotence:**

$(A \vee A) \equiv A$, both represented by $\{A\}$

Sets are a convenient representation of conjunctions and disjunctions with built in associativity, commutativity and idempotence

CNF-SAT: Input: Set of clauses F

Question: Is F unsatisfiable?

DPLL — The simplest algorithm for CNF-SAT

Simplest algorithm: Construct the truth table.

Best-case runtime is $\Theta(m \cdot 2^n)$ for a formula of length m over n variables.

DPLL — A first improvement: Partial Evaluation

Improvement: **partial evaluation** using Boole-Shannon expansion

Lemma (Boole-Shannon Expansion)

For every formula F and atom A :

$$F \equiv (A \wedge F[\top/A]) \vee (\neg A \wedge F[\perp/A]).$$

Proof By structural induction on F (exercise).

Corollary

F is satisfiable iff $F[\perp/A]$ or $F[\top/A]$ are satisfiable.

DPLL — First step: partial evaluation

$F[\perp/A]$ and $F[\top/A]$ easy to compute in clause normal form:

$F[\top/A] \equiv$ take F , remove all clauses with A , remove all $\neg A$.

$F[\perp/A] \equiv$ take F , remove all clauses with $\neg A$, remove all A .

Partial evaluation algorithm:

Given formula F , total order on the variables \prec :

If $\{\} \in F$ return **unsatisfiable**.

If $F = \emptyset$ return **satisfiable**.

Otherwise:

Fix the first variable A in F according to \prec .

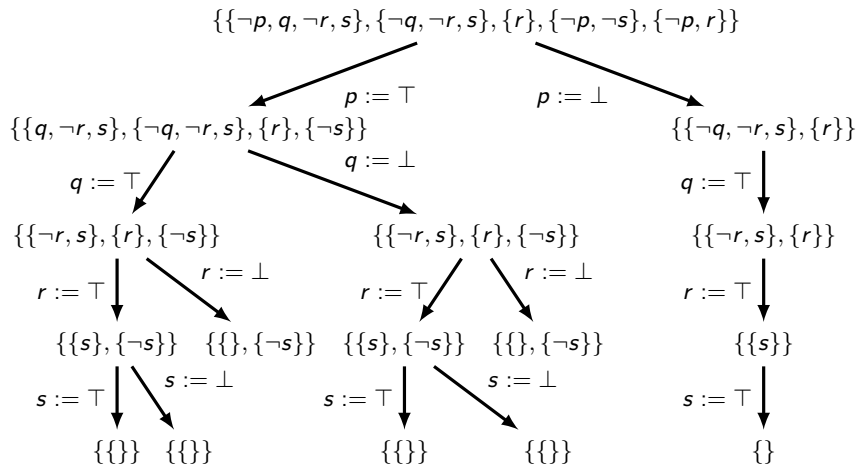
Recursively check if $F[\perp/A]$ is satisfiable;

if yes, return **satisfiable**.

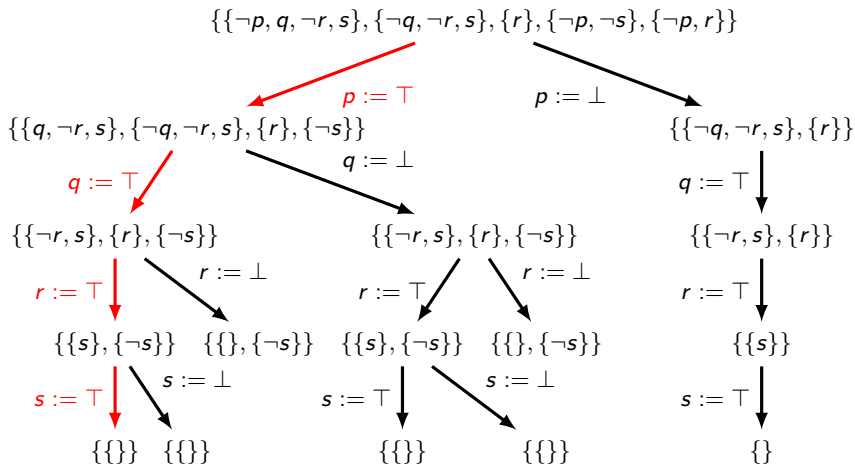
Recursively check if $F[\top/A]$ is satisfiable;

if yes, return **satisfiable**, otherwise **unsatisfiable**.

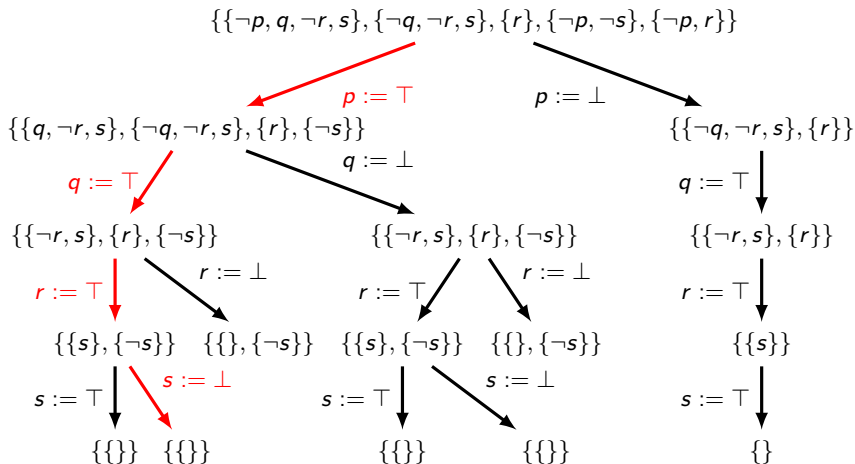
DPLL: Davis-Putnam-Logemann-Loveland



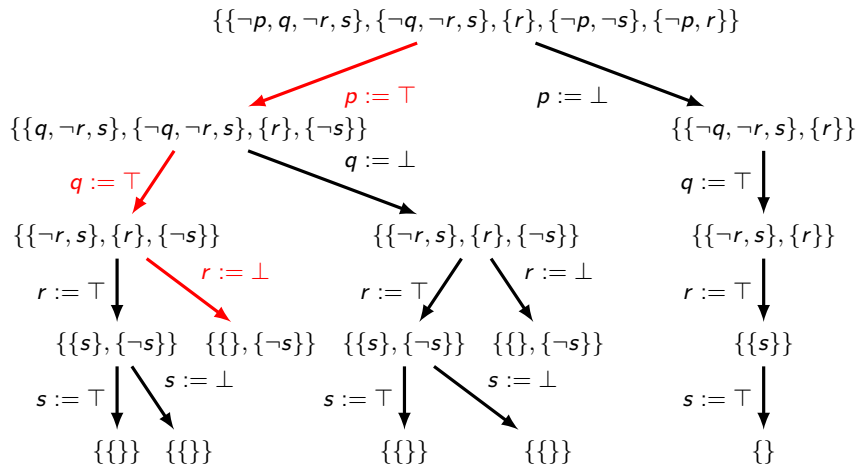
DPLL: Davis-Putnam-Logemann-Loveland



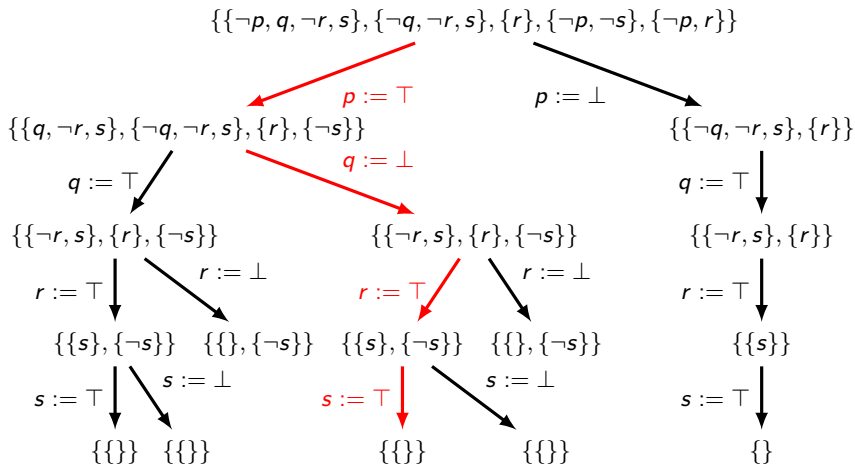
DPLL: Davis-Putnam-Logemann-Loveland



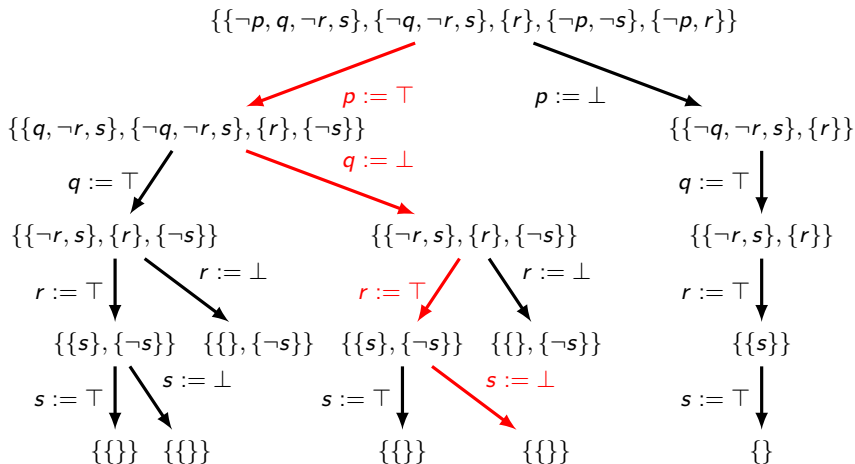
DPLL: Davis-Putnam-Logemann-Loveland



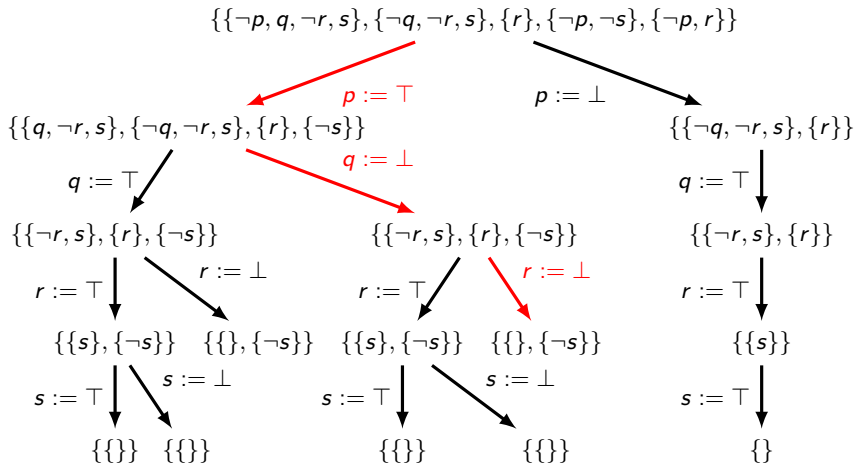
DPLL: Davis-Putnam-Logemann-Loveland



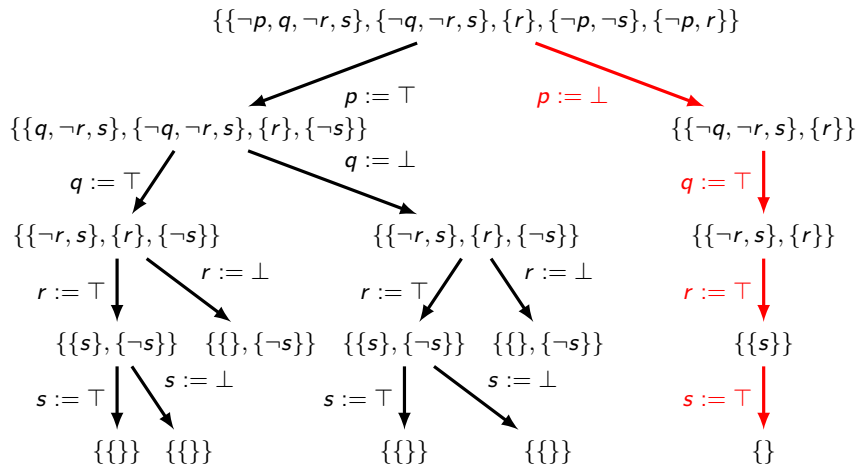
DPLL: Davis-Putnam-Logemann-Loveland



DPLL: Davis-Putnam-Logemann-Loveland



DPLL: Davis-Putnam-Logemann-Loveland



DPLL: Davis-Putnam-Logemann-Loveland

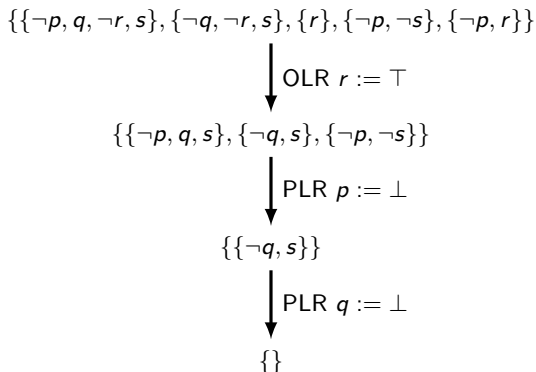
Instead of fixing an order on variables, choose the next variable **dynamically**.

- ▶ **OLR**: one-literal rule If $\{L\} \in F$ ($\{L\}$ is called **unit clause**), then every satisfying assignment sets L to true. So it suffices to check satisfiability of $F[\top/L]$.
- ▶ **PLR**: pure-literal rule
If L appears in F and \bar{L} does not, then it also suffices to check satisfiability of $F[\top/L]$ (**Why?**).

DPLL algorithm: Partial evaluation that gives priority to a variable satisfying **OLR**, then to a variable satisfying **PLR**, and otherwise picks the first unpicked variable of \prec .

Applying **OLR** can generate further unit clauses (**unit propagation**). Same for **PLR**, but DPLL often implemented with only **OLR** for efficiency.

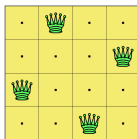
DPLL: Davis-Putnam-Logemann-Loveland



In this example PLR and OLR allow us to avoid all case splits.

Example: 4 queens

Problem: place 4 non-attacking queens on a 4x4 chess board



Variable p_{ij} models: there is a queen in square (i, j)

- ▶ ≥ 1 in each row: $\bigwedge_{i=1}^4 \bigvee_{j=1}^4 p_{ij}$
- ▶ ≤ 1 in each row: $\bigwedge_{i=1}^4 \bigwedge_{j \neq j'=1}^4 \neg p_{ij} \vee \neg p_{ij'}$
- ▶ ≤ 1 in each column: $\bigwedge_{j=1}^4 \bigwedge_{i \neq i'=1}^4 \neg p_{ij} \vee \neg p_{i'j}$
- ▶ ≤ 1 on each diagonal: $\bigwedge_{i,j=1}^4 \bigvee_k \neg p_{i-k,i+k} \vee \neg p_{i+k,j+k}$

Total number of clauses: $4 + 24 + 24 + 28 = 80$

DPLL: 4 queens

Running the DPLL algorithm:

- ▶ Start with $p_{11} \mapsto 1$
delete $\{p_{11}, p_{12}, p_{13}, p_{14}\}$, delete $\neg p_{11}$: 9 new unit clauses
unit propagation: deletes 65 clauses!
- ▶ Set $p_{23} \mapsto 1$
4 new unit clauses: $\{\neg p_{24}\}, \{\neg p_{43}\}, \{\neg p_{32}\}, \{\neg p_{34}\}$
unit propagation of $\{\neg p_{34}\}$: UNSAT
fixing only two literals collapsed from 80 clauses to 1
ruled out 2^{14} of 2^{16} possible assignments!
- ▶ Backtrack: $p_{11} \mapsto 0, p_{12} \mapsto 1$
delete $\{\neg p_{12}\}$: 9 new unit clauses
unit propagation: leaves only 1 clause $\{p_{43}\}$!
- ▶ Answer: $p_{12}, p_{24}, p_{31}, p_{43} \mapsto 1$

DPLL: Evaluation

Oriented towards satisfiability:

- ▶ $2^{O(n)}$ time for satisfiable formulas, but $2^{\Theta(n)}$ for unsatisfiable ones.
- ▶ DPLL computes a satisfying assignment, if there is one.
- ▶ The satisfying assignment is a **certificate** of satisfiability.
- ▶ Satisfiable formulas have short certificates: satisfying assignment never larger than the formula.

Coming next: **resolution**, a procedure oriented towards unsatisfiability.

- ▶ $2^{O(n)}$ time for unsatisfiable formulas, but $2^{\Theta(n)}$ for satisfiable ones.
- ▶ Resolution computes a certificate of unsatisfiability.
- ▶ However, the certificate is **exponentially longer** than the formula in the worst case.
- ▶ Polynomial certificates for satisfiability implies $NP = coNP$.

Propositional Logic

Compactness

Compactness Theorem

Theorem

*A set S of formulas is satisfiable
iff every finite subset of S is satisfiable.*

Equivalent formulation:

*A set S of formulas is unsatisfiable
iff some finite subset of S is unsatisfiable.*

An application: Graph Coloring

Definition

A **4-coloring** of a graph (V, E) is a map $c : V \rightarrow \{1, 2, 3, 4\}$ such that $(x, y) \in E$ implies $c(x) \neq c(y)$.

Theorem (4CT)

A finite planar graph has a 4-coloring.

Theorem

A planar graph $G = (V, E)$ with countably many vertices $V = \{v_1, v_2, \dots\}$ has a 4-coloring.

Proof $G \rightsquigarrow$ set of formulas S s.t. S is sat. iff G is 4-col.

G is planar

\Rightarrow every finite subgraph of G is planar and 4-col. (by 4CT)

\Rightarrow every finite subset of S is sat.

$\Rightarrow S$ is sat. (by Compactness)

$\Rightarrow G$ is 4-col.

Proof details

$G \rightsquigarrow S$:

For simplicity:

atoms are of the form A_i^c where $c \in \{1, \dots, 4\}$ and $i \in \mathbb{N}$

$$S := \{A_i^1 \vee A_i^2 \vee A_i^3 \vee A_i^4 \mid i \in \mathbb{N}\} \cup \\ \{A_i^c \rightarrow \neg A_i^d \mid i \in \mathbb{N}, c, d \in \{1, \dots, 4\}, c \neq d\} \cup \\ \{\neg(A_i^c \wedge A_j^c) \mid (v_i, v_j) \in E, c \in \{1, \dots, 4\}\}$$

Subgraph corresponding to some $T \subseteq S$:

$$V_T := \{v_i \mid A_i^c \text{ occurs in } T \text{ (for some } c)\}$$

$$E_T := \{(v_i, v_j) \mid \neg(A_i^c \wedge A_j^c) \in T \text{ (for some } c)\}$$

Proof of Compactness

Theorem

*A set S of formulas is satisfiable
iff every finite subset of S is satisfiable.*

Proof

\Rightarrow : If S is satisfiable then every finite subset of S is satisfiable.

Trivial.

\Leftarrow : If every finite subset of S is satisfiable then S is satisfiable.

We prove that S has a model.

Proof of Compactness

Definition

Let $b_1 \cdots b_n \in \{0, 1\}^*$ with $n \geq 0$ and let T be a set of formulas. An assignment \mathcal{A} is a $b_1 \cdots b_n$ -model of T if $\mathcal{A}(A_i) = b_i$ for every $i = 1, \dots, n$ and $\mathcal{A} \models T$.

In particular: every model is a ε -model.

Assume every finite $T \subseteq S$ is satisfiable. We prove:

1. There is an infinite sequence $b_1 b_2 \cdots \in \{0, 1\}^\omega$ such that for every $n \geq 1$ all finite $T \subseteq S$ have a $b_1 \cdots b_n$ -model.
2. The assignment \mathcal{B} given by $\mathcal{B}(A_i) := b_i$ for all $i \geq 1$ is a model of S .

Proof of Compactness: Part (1)

To prove: There is an infinite sequence $b_1 b_2 \cdots \in \{0, 1\}^\omega$ such that for every $n \geq 1$ all finite $T \subseteq S$ have a $b_1 \cdots b_n$ -model.

It suffices to show:

- (a) Every finite $T \subseteq S$ has an ε -model.
- (b) For every sequence $\sigma \in \{0, 1\}^*$: if every finite $T \subseteq S$ has a σ -model then there exists $b \in \{0, 1\}$ such that every finite $T \subseteq S$ has a σb -model.

Proof of (a): By assumption every $T \subseteq S$ has an ε -model.

Proof of (b): Next slide.

Proof of Compactness: Part (1)

Proof of (b): By contradiction.

Assume that for some $\sigma \in \{0, 1\}$: every finite $T \subseteq S$ has a σ -model, but

- (0) some finite $T_0 \subseteq S$ has no $\sigma 0$ -model; and
- (1) some finite $T_1 \subseteq S$ has no $\sigma 1$ -model.

Consider the finite set $T_0 \cup T_1$.

By assumption, $T_0 \cup T_1$ has some σ -model \mathcal{A} . Let $n := |\sigma|$.

Two possible cases:

- ▶ $\mathcal{A}(A_{n+1}) = 0$. Then \mathcal{A} is a $\sigma 0$ -model of T_0 , contradicting (0).
- ▶ $\mathcal{A}(A_{n+1}) = 1$. Then \mathcal{A} is a $\sigma 1$ -model of T_1 , contradicting (1).

Proof of Compactness: Part (2)

To prove: The assignment \mathcal{B} given by $\mathcal{B}(A_i) := b_i$ for all $i \geq 1$, where $b_1 b_2 \dots$ is the infinite sequence of (1), is a model of S .

We show $\mathcal{B} \models F$ for all $F \in S$.

Let m be the maximal index of all atoms in F .

By (1), $\{F\}$ has a $b_1 \dots b_m$ -model \mathcal{A} .

Hence $\mathcal{B} \models F$, because \mathcal{A} and \mathcal{B} agree on all atoms in F .

Corollary

Corollary

If $S \models F$ then there is a finite subset $M \subseteq S$ such that $M \models F$.

Propositional Logic Resolution

Resolution — The idea

Input: Set of clauses F

Question: Is F unsatisfiable?

Algorithm:

Keep on “resolving” two clauses from F and adding the result to F until the empty clause is found

Correctness:

If the empty clause is found, the initial F is unsatisfiable

Completeness:

If the initial F is unsatisfiable, the empty clause can be found.

Correctness/Completeness of **syntactic** procedure (**resolution**)
w.r.t. **semantic** property (**unsatisfiability**)

Resolvent

Definition

Let L be a literal. Then \bar{L} is defined as follows:

$$\bar{L} = \begin{cases} \neg A_i & \text{if } L = A_i \\ A_i & \text{if } L = \neg A_i \end{cases}$$

Definition

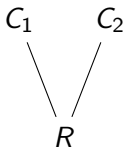
Let C_1, C_2 be clauses and let L be a literal such that $L \in C_1$ and $\bar{L} \in C_2$. Then the clause

$$(C_1 - \{L\}) \cup (C_2 - \{\bar{L}\})$$

is a **resolvent** of C_1 and C_2 .

The process of deriving the resolvent is called a **resolution step**.

Graphical representation of resolvent:



If $C_1 = \{L\}$ and $C_2 = \{\bar{L}\}$ then the empty clause is a resolvent of C_1 and C_2 . The special symbol \square denotes the empty clause.

Recall: \square represents \perp .

Resolution proof

Definition

A **resolution proof** of a clause C from a set of clauses F is a sequence of clauses C_0, \dots, C_n such that

- ▶ $C_i \in F$ or C_i is a resolvent of two clauses C_a and C_b , $a, b < i$,
- ▶ $C_n = C$

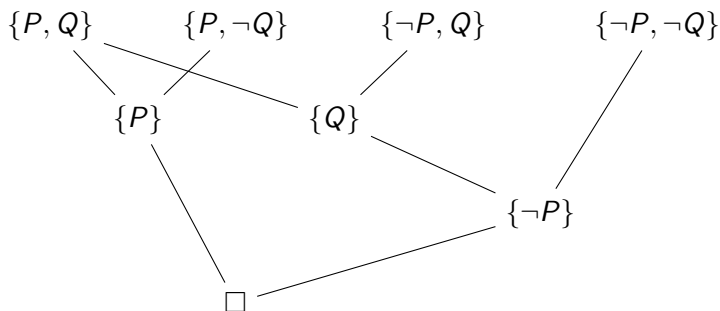
Then we can write $F \vdash_{Res} C$.

Note: F can be finite or infinite!

Resolution proof as DAG

A resolution proof can be shown as a DAG with the clauses in F as the leaves and C as the root:

Example



A linear resolution proof

0: $\{P, Q\}$

1: $\{P, \neg Q\}$

2: $\{\neg P, Q\}$

3: $\{\neg P, \neg Q\}$

4: $\{P\}$ (0, 1)

5: $\{Q\}$ (0, 2)

6: $\{\neg P\}$ (3, 5)

7: \square (4, 6)

Correctness of resolution

Lemma (Resolution Lemma)

Let R be a resolvent of two clauses C_1 and C_2 . Then $C_1, C_2 \models R$.

Proof By definition $R = (C_1 - \{L\}) \cup (C_2 - \{\bar{L}\})$ (for some L).

Assume $\mathcal{A} \models C_1$ and $\mathcal{A} \models C_2$. We show $\mathcal{A} \models R$.

There are two cases:

- ▶ $\mathcal{A} \models L$. Then $\mathcal{A} \models C_2 - \{\bar{L}\}$ (because $\mathcal{A} \models C_2$), thus $\mathcal{A} \models R$.
- ▶ $\mathcal{A} \not\models L$. Then $\mathcal{A} \models C_1 - \{L\}$ (because $\mathcal{A} \models C_1$), thus $\mathcal{A} \models R$.

Correctness of resolution

Theorem (Correctness of resolution)

Let F be a set of clauses. *If $F \vdash_{Res} C$ then $F \models C$.*

Proof Assume there is a resolution proof $C_0, \dots, C_n = C$. We show $F \models C_i$ by induction on i . IH: $F \models C_j$ for all $j < i$. There are two cases:

- ▶ $C_i \in F$.
Then $F \models C_i$ by definition.
- ▶ C_i is a resolvent of C_a and C_b for $a, b < i$.
Then $F \models C_a$ and $F \models C_b$ by IH, and $C_a, C_b \models C_i$ by the resolution lemma. Thus $F \models C_i$.

Corollary

Let F be a set of clauses. *If $F \vdash_{Res} \square$ then F is unsatisfiable.*

Completeness of resolution

Theorem

Let F be a finite set of clauses. If F is unsatisfiable then $F \vdash_{Res} \square$.

Theorem (Completeness of resolution)

Let F be a set of clauses. If F is unsatisfiable then $F \vdash_{Res} \square$.

Proof If F is infinite, there must be a finite unsatisfiable subset of F (by the Compactness Theorem); in that case let F be that finite subset and apply the previous theorem.

Corollary

A set of clauses F is unsatisfiable iff $F \vdash_{Res} \square$.

Completeness proof

Corollary

(of the Boole-Shannon expansion) F is unsatisfiable iff $F[\perp/A]$ and $F[\top/A]$ are unsatisfiable.

Idea for completeness proof:

If A is an atom of F , then both $F[\perp/A]$ and $F[\top/A]$ have fewer atoms than F .

Use Boole-Shannon to prove completeness by induction on the number of atoms of the unsatisfiable formula F :

- ▶ construct inductively resolution proofs for $F[\perp/A]$ and $F[\top/A]$, and
- ▶ “combine” them into a resolution proof for F .

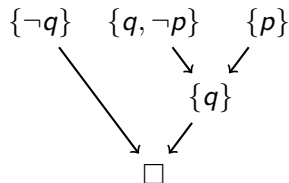
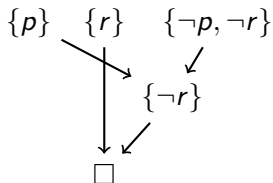
Inductive construction of resolution proofs

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

- Compute inductively proofs for $F[\top/s]$ and $F[\perp/s]$.

$$F[\top/s] \equiv \{ \{p\}, \{r\}, \{\neg p, \neg r\} \}$$

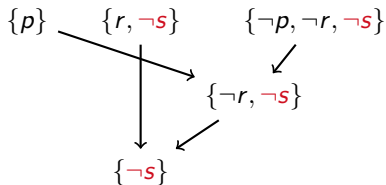
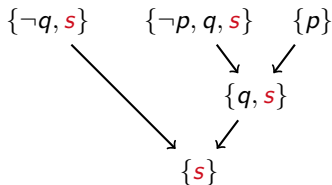
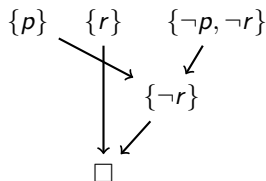
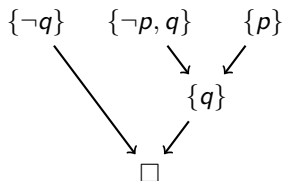
$$F[\perp/s] \equiv \{ \{\neg q\}, \{\neg p, q\}, \{p\} \}$$



Inductive construction of resolution proofs

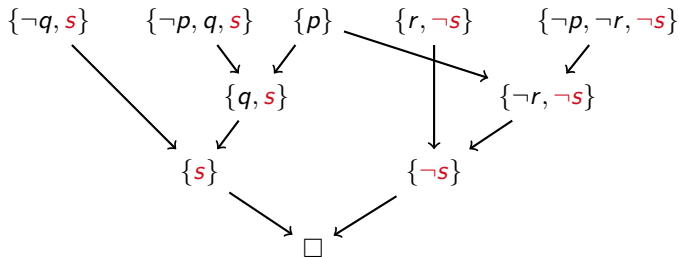
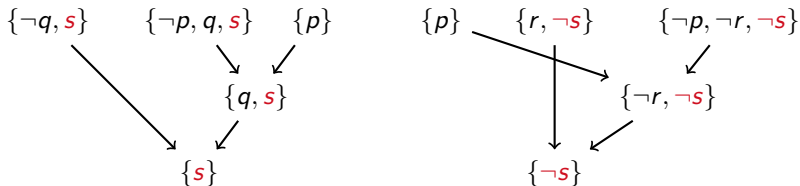
- ▶ Reintroduce s and $\neg s$.

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



Inductive construction of resolution proofs

- ▶ Combine the graphs for $\{s\}$ and $\{\neg s\}$.



Completeness proof

Theorem

Let F be a finite set of clauses. *If F is unsatisfiable then $F \vdash_{Res} \square$.*

Proof By induction on the number n of distinct atoms in F .

Basis: If $n = 0$ then $F = \{\}$ (but F is unsat.) or $F = \{\square\}$.

Step:

IH: For every unsat. set of clauses F with n dist. atoms, $F \vdash_{Res} \square$.

Let F contain $n + 1$ distinct atoms. Pick some atom A in F .

$F[\top/A] \equiv$ take F , remove all clauses with A , remove all $\neg A$.

$F[\perp/A] \equiv$ take F , remove all clauses with $\neg A$, remove all A .

Completeness proof

By IH: there are res. proofs $C_0, \dots, C_m = \square$ from $F[\perp/A]$ and $D_0, \dots, D_n = \square$ from $F[\top/A]$.

Now transform C_0, \dots, C_m into a proof C'_0, \dots, C'_m from F by adding A back into the clauses it was removed from. Then:

- ▶ either $C'_m = \{A\}$
- ▶ or $C'_m = \square$ (and we are done).

Similarly we transform D_0, \dots, D_n into a proof D'_0, \dots, D'_n from F by adding $\neg A$ back in. Then:

- ▶ either $D'_n = \{\neg A\}$
- ▶ or $D'_n = \square$ (and we are done).

If $C'_m = \{A\}$ and $D'_n = \{\neg A\}$ then $F \vdash_{Res} A$ and $F \vdash_{Res} \neg A$ and thus $F \vdash_{Res} \square$.

Resolution is only refutation complete

Not everything that is a consequence of a set of clauses
can be derived by resolution.

Exercise

Find F and C such that $F \models C$ but not $F \vdash_{Res} C$.

How to prove $F \models C$ by resolution?

Prove $F \cup \{\neg C\} \vdash_{Res} \square$

A resolution algorithm

Input: A CNF formula F , i.e. a finite set of clauses

while there are clauses $C_a, C_b \in F$ and resolvent R of C_a and C_b
such that $R \notin F$
do $F := F \cup \{R\}$

Lemma

The algorithm terminates.

Proof There are only finitely many clauses over a finite set of atoms.

Theorem

The initial F is unsatisfiable iff \square is in the final F

Proof F_{init} is unsat. iff $F_{init} \vdash_{Res} \square$ iff $\square \in F_{final}$ because the algorithm enumerates all R such that $F_{init} \vdash_{Res} R$.

The algorithm is a decision procedure for unsat. of CNF formulas.

Propositional Logic
CDCL: Conflict Driven Clause
Learning

CDCL: goal and idea

Goal: Combine DPLL and resolution into an algorithm oriented towards both satisfiability and unsatisfiability.

Idea: At every unsuccessful leaf of DPLL (called **conflict**), compute a **conflict clause**, and add it to the formula we are deciding about.

Conflict clauses “cache” previous search results, so we “learn from previous mistakes”.

Conflict clauses also determine backtracking.

We present a particular way of computing a conflict clause using resolution. There are other ways.

DPLL + CDCL algorithm

Given formula F and **partial assignment** \mathcal{A} :

$F|_{\mathcal{A}}$ denotes the result of deleting any clause containing a true literal, and deleting all false literals from each remaining clause.

Input: CNF formula F .

1. Initialise \mathcal{A} to the empty assignment
2. While there is unit clause $\{L\}$ or pure literal L in $F|_{\mathcal{A}}$, update $\mathcal{A} \mapsto \mathcal{A}[\top/L]$
3. If $F|_{\mathcal{A}} = \emptyset$, stop and output \mathcal{A} .
4. If $F|_{\mathcal{A}} \ni \square$, add new clause C to F by **learning procedure**.
If $C = \square$, stop and output UNSAT; otherwise backtrack to highest level where C is unit clause.
Go to line 2.
5. Apply **decision strategy** to update \mathcal{A} .
Go to line 2.

Terminology

- ▶ **State** of algorithm is pair (F, \mathcal{A}) , where F is CNF formula and \mathcal{A} is partial assignment.
Successful state when $\mathcal{A} \models F$. **Conflict state** when $\mathcal{A} \not\models F$.
(Note: conflict state if $F|_{\mathcal{A}} \ni \square$, successful state if $F|_{\mathcal{A}} = \emptyset$)
- ▶ Each assignment $A_i \mapsto b_i$ classifies as **decision assignment** or **implied assignment**.
- ▶ $A_i \mapsto b_i$ denotes decision assignment with **decision variable** A_i .
- ▶ $A_i \xrightarrow{C} b_i$ denotes an implied assignment arising through **unit propagation** on clause C .
- ▶ **Decision level** of assignment $A_i \mapsto b_i$ in a given state (F, \mathcal{A}) is number of decision assignments in \mathcal{A} that precede $A_i \mapsto b_i$.

Example: start with set of clauses $F = \{C_1, \dots, C_5\}$, where

$$C_1 = \{\neg A_1, \neg A_4, A_5\}$$

$$C_2 = \{\neg A_1, A_6, \neg A_5\}$$

$$C_3 = \{\neg A_1, \neg A_6, A_7\}$$

$$C_4 = \{\neg A_1, \neg A_7, \neg A_5\}$$

$$C_5 = \{A_1, A_4, A_6\}$$

Say current assignment is $(A_1 \mapsto 1, A_2 \mapsto 0, A_3 \mapsto 0, A_4 \mapsto 1)$.

Notice $F|_{\mathcal{A}}$ contains unit clause $\{A_5\}$.

Unit propagation further generates $(A_5 \xrightarrow{C_1} 1, A_6 \xrightarrow{C_2} 1, A_7 \xrightarrow{C_3} 1)$.

This leads to a conflict, with C_4 being made false.

Conflict analysis

After unit propagation:

- ▶ If not in conflict nor successful, make decision (line 5)
- ▶ If in conflict, **learned clause** is added (line 4)

Learned clause desiderata: If unit propagation from state (F, \mathcal{A}) leads to conflict, clause C is learned such that:

1. $F \equiv F \cup \{C\}$
2. C is **conflict clause**: each literal of C is made false by \mathcal{A}
3. C mentions only decision variables in \mathcal{A}

Clause learning using resolution

Suppose $\mathcal{A} = (A_1 \mapsto b_1, \dots, A_k \mapsto b_k)$ leads to conflict.

Find associated clauses D_1, \dots, D_{k+1} by backward induction:

1. $D_{k+1} :=$ any conflict clause of F under \mathcal{A} .
2. If $A_i \mapsto b_i$ is decision assignment or A_i not mentioned in D_{i+1} , set $D_i := D_{i+1}$.
3. If $A_i \stackrel{C_i}{\mapsto} b_i$ is implied assignment and A_i mentioned in D_{i+1} , define D_i to be resolvent of D_{i+1} and C_i with respect to A_i .

$C := A_1$, that is, the final clause A_1 is the **learned clause** .

Clause learning: example

Conflict of example above:

$$\begin{array}{ll} C_1 = \{\neg A_1, \neg A_4, A_5\} & D_8 := \{\neg A_1, \neg A_7, \neg A_5\} \quad (\text{clause } C_4) \\ C_2 = \{\neg A_1, A_6, \neg A_5\} & D_7 := \{\neg A_1, \neg A_5, \neg A_6\} \quad (\text{resolve } D_8, C_3) \\ C_3 = \{\neg A_1, \neg A_6, A_7\} & D_6 := \{\neg A_1, \neg A_5\} \quad (\text{resolve } D_7, C_2) \\ C_4 = \{\neg A_1, \neg A_7, \neg A_5\} & D_5 := \{\neg A_1, \neg A_4\} \quad (\text{resolve } D_6, C_1) \\ C_5 = \{A_1, A_4, A_6\} & D_4 := \{\neg A_1, \neg A_4\} \\ A_1 \mapsto 1, A_2 \mapsto 0, & D_3 := \{\neg A_1, \neg A_4\} \\ A_3 \mapsto 0, A_4 \mapsto 1, & D_2 := \{\neg A_1, \neg A_4\} \\ A_5 \stackrel{C_1}{\mapsto} 1, A_6 \stackrel{C_2}{\mapsto} 1, & D_1 := \{\neg A_1, \neg A_4\} \\ A_7 \stackrel{C_3}{\mapsto} 1 & \end{array}$$

Learned clause D_1 is conflict clause with only decision variables, including top-level one A_1 .

Clause learning: example

Intuitively:

- ▶ D_1 records that conflict due to decision to make A_1, A_4 true.
- ▶ Adding D_1 ensures search does not explore assignments with $A_1 \mapsto 1, A_4 \mapsto 1$.
- ▶ DPLL backtracks to highest level where D_1 is unit clause (after $A_1 \mapsto 1$), unit propagation leads to $A_4 \mapsto 0$.

Clause learning

Proposition: The clause learning procedure satisfies the three desiderata.

Proof sketch: **Observation:** If $A_i \stackrel{C_i}{\mapsto} b_i$, then the only literal of C_i true under \mathcal{A} is the literal for A_i (that is, C_i contains either A_i or $\neg A_i$, and b_i is chosen to make the literal true).

1. $F \equiv F \cup \{C\}$

Because C is obtained from clauses of F through resolution steps.

2. C is **conflict clause**: each literal is made false by \mathcal{A} .

We show by induction that $D_{k+1}, D_k, \dots, D_1 = C$ are conflict clauses.

D_{k+1} is conflict clause by definition.

If D_{i+1} is conflict clause and $D_i = D_{i+1}$, then so is D_i .

If D_{i+1} is conflict clause and $D_i \neq D_{i+1}$, then D_i is the result of resolving D_{i+1} and C_i . By the **observation**, all literals of D_i are made false by \mathcal{A} .

3. C mentions only decision variables in \mathcal{A} .

Because every other variable, say A_i , disappears after resolving with D_{i+1} w.r.t. A_i .

Indeed, since \mathcal{A} makes D_{i+1} false, by the **observation** A_i has opposite signs in D_{i+1} and C_i .

Example (without PLR)

$$\{\neg A_1\} \{A_1, A_3, A_4\} \{\neg A_2, \neg A_5\} \{A_3, \neg A_4, A_5, \neg A_6\} \{A_1, \neg A_2, \neg A_4, A_6\}$$

$$\text{OLR: } A_1 \mapsto 0 \quad \{A_3, A_4\} \quad \{\neg A_2, \neg A_5\} \quad \{A_3, \neg A_4, A_5, \neg A_6\} \quad \{\neg A_2, \neg A_4, A_6\}$$

$$\text{DE: } A_2 \mapsto 1 \quad \{A_3, A_4\} \quad \{\neg A_5\} \quad \{A_3, \neg A_4, A_5, \neg A_6\} \quad \{\neg A_4, A_6\}$$

$$\text{OLR: } A_5 \mapsto 0 \quad \{A_3, A_4\} \quad \{A_3, \neg A_4, \neg A_6\} \quad \{\neg A_4, A_6\}$$

$$\text{DE: } A_3 \mapsto 0 \quad \{A_4\} \quad \{\neg A_4, \neg A_6\} \quad \{\neg A_4, A_6\}$$

$$\text{OLR: } A_4 \mapsto 1 \quad \{\neg A_6\} \quad \{A_6\}$$

$$\text{OLR: } A_6 \mapsto 1 \quad \{\}$$

$$D_7 := \{A_3, \neg A_4, A_5, \neg A_6\} \quad (\text{conflict clause})$$

$$D_6 := \{A_1, \neg A_2, A_3, \neg A_4, A_5\} \quad (\text{resolve } D_7, \{A_1, \neg A_2, \neg A_4, A_6\})$$

$$D_5 := \{A_1, \neg A_2, A_3, A_5\} \quad (\text{resolve } D_6, \{A_1, A_3, A_4\})$$

$$D_4 := \{A_1, \neg A_2, A_3, A_5\}$$

$$D_3 := \{A_1, \neg A_2, A_3\} \quad (\text{resolve } D_4, \{\neg A_2, \neg A_5\})$$

$$D_2 := \{A_1, \neg A_2, A_3\}$$

$$D_1 := \{\neg A_2, A_3\} \quad (\text{resolve } D_2, \{\neg A_1\})$$

Backtracking to $\{A_1 \mapsto 0, A_2 \mapsto 1\}$. Unit propagation: $A_3 \mapsto 1$.

Basic Proof Theory

Propositional Logic

(See the book by Troelstra and Schwichtenberg)

Proof rules and proof systems

Proof systems are defined by (proof or **inference**) rules of the form

$$\frac{T_1 \quad \dots \quad T_n}{T} \text{ rule-name}$$

where T_1, \dots, T_n (**premises**) and T (**conclusion**) are syntactic objects (eg formulas).

Intuitive reading: If T_1, \dots, T_n are provable, then T is provable.

Degenerate case: If $n = 0$ the rule is called an **axiom** and the horizontal line is sometimes omitted.

If some U is provable, we write $\vdash U$.

Proof trees

Proofs (also: **derivations**) are drawn as trees of nested proof rules.

Example:

$$\frac{\frac{\overline{T_1} \quad \overline{U}}{\overline{T_2}} \quad \overline{T_3}}{\frac{S_1 \quad S_2}{R}}$$

We sometimes omit the names of proof rules in a proof tree if they are obvious or for space reasons. **You should always show them!**

Every fragment

$$\frac{T_1 \quad \dots \quad T_n}{T}$$

of a proof tree must be (an instance of) a proof rule.

All proofs must start with axioms.

The **depth** of a proof tree is the number of rules on the longest branch of the tree. Thus ≥ 1

Abbreviations

Until further notice:

\perp , \neg , \wedge , \vee , \rightarrow are primitives.

\top abbreviates $\neg\perp$

A possible simplification:

$\neg F$ abbreviates $F \rightarrow \perp$

We now consider three important proof systems:

- ▶ Sequent Calculus
- ▶ Natural Deduction
- ▶ Hilbert Systems

Sequent Calculus

Propositional Logic

Sequent Calculus

Invented by Gerhard Gentzen in 1935. Birth of proof theory.

Proof rules

$$\frac{S_1 \quad \dots \quad S_n}{S}$$

where S_1, \dots, S_n and S are **sequents**: expressions of the form

$$\Gamma \Rightarrow \Delta$$

with Γ and Δ finite **multisets** of formulas.

Multiset = set with possibly repeated elements; using sets possible but less elegant.

Notice: \Rightarrow is just a—suggestive—separator

Intention of the calculus:

$$\Gamma \Rightarrow \Delta \text{ is provable (derivable) iff } \bigwedge \Gamma \models \bigvee \Delta \quad (\bigwedge \Gamma \rightarrow \bigvee \Delta \text{ valid})$$

Sequents: Notation

- ▶ We use set notation for multisets, e.g. $\{A, B \rightarrow C, A\}$
- ▶ Drop $\{\}$: $F_1, \dots, F_m \Rightarrow G_1, \dots, G_n$
- ▶ F, Γ abbreviates $\{F\} \cup \Gamma$ (similarly for Δ)
- ▶ Γ_1, Γ_2 abbreviates $\Gamma_1 \cup \Gamma_2$ (similarly for Δ)

Sequent Calculus rules

$$\frac{}{\perp, \Gamma \Rightarrow \Delta} \quad \perp L$$

$$\frac{\Gamma \Rightarrow F, \Delta}{\neg F, \Gamma \Rightarrow \Delta} \quad \neg L$$

$$\frac{F, G, \Gamma \Rightarrow \Delta}{F \wedge G, \Gamma \Rightarrow \Delta} \quad \wedge L$$

$$\frac{F, \Gamma \Rightarrow \Delta \quad G, \Gamma \Rightarrow \Delta}{F \vee G, \Gamma \Rightarrow \Delta} \quad \vee L$$

$$\frac{\Gamma \Rightarrow F, \Delta \quad G, \Gamma \Rightarrow \Delta}{F \rightarrow G, \Gamma \Rightarrow \Delta} \quad \rightarrow L$$

$$\frac{}{A, \Gamma \Rightarrow A, \Delta} \quad Ax$$

$$\frac{F, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \neg F, \Delta} \quad \neg R$$

$$\frac{\Gamma \Rightarrow F, \Delta \quad \Gamma \Rightarrow G, \Delta}{\Gamma \Rightarrow F \wedge G, \Delta} \quad \wedge R$$

$$\frac{\Gamma \Rightarrow F, G, \Delta}{\Gamma \Rightarrow F \vee G, \Delta} \quad \vee R$$

$$\frac{F, \Gamma \Rightarrow G, \Delta}{\Gamma \Rightarrow F \rightarrow G, \Delta} \quad \rightarrow R$$

Sequent Calculus rules

Intuition: read backwards as proof search rules

$$\frac{}{\perp, \Gamma \Rightarrow \Delta} \quad \perp L$$

$$\frac{\Gamma \Rightarrow F, \Delta}{\neg F, \Gamma \Rightarrow \Delta} \quad \neg L$$

$$\frac{F, G, \Gamma \Rightarrow \Delta}{F \wedge G, \Gamma \Rightarrow \Delta} \quad \wedge L$$

$$\frac{F, \Gamma \Rightarrow \Delta \quad G, \Gamma \Rightarrow \Delta}{F \vee G, \Gamma \Rightarrow \Delta} \quad \vee L$$

$$\frac{\Gamma \Rightarrow F, \Delta \quad G, \Gamma \Rightarrow \Delta}{F \rightarrow G, \Gamma \Rightarrow \Delta} \quad \rightarrow L$$

$$\frac{}{A, \Gamma \Rightarrow A, \Delta} \quad Ax$$

$$\frac{F, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \neg F, \Delta} \quad \neg R$$

$$\frac{\Gamma \Rightarrow F, \Delta \quad \Gamma \Rightarrow G, \Delta}{\Gamma \Rightarrow F \wedge G, \Delta} \quad \wedge R$$

$$\frac{\Gamma \Rightarrow F, G, \Delta}{\Gamma \Rightarrow F \vee G, \Delta} \quad \vee R$$

$$\frac{F, \Gamma \Rightarrow G, \Delta}{\Gamma \Rightarrow F \rightarrow G, \Delta} \quad \rightarrow R$$

Every rule decomposes its principal formula

$$\overline{\Rightarrow (P \vee R) \wedge (Q \vee \neg R) \rightarrow P \vee Q}$$

$$\frac{}{\Rightarrow (P \vee R) \wedge (Q \vee \neg R) \rightarrow P \vee Q} \rightarrow R$$

$$\frac{F, \Gamma \Rightarrow G, \Delta}{\Gamma \Rightarrow F \rightarrow G, \Delta} \rightarrow R$$

$$\frac{\overline{(P \vee R) \wedge (Q \vee \neg R) \Rightarrow P \vee Q}}{\Rightarrow (P \vee R) \wedge (Q \vee \neg R) \rightarrow P \vee Q} \rightarrow R$$

$$\frac{F, \Gamma \Rightarrow G, \Delta}{\Gamma \Rightarrow F \rightarrow G, \Delta} \rightarrow R$$

$$\frac{\overline{(P \vee R) \wedge (Q \vee \neg R) \Rightarrow P \vee Q} \wedge L}{\Rightarrow (P \vee R) \wedge (Q \vee \neg R) \rightarrow P \vee Q} \rightarrow R$$

$$\frac{F, G, \Gamma \Rightarrow \Delta}{F \wedge G, \Gamma \Rightarrow \Delta} \wedge L$$

$$\frac{\frac{\overline{P \vee R, Q \vee \neg R \Rightarrow P \vee Q}}{(P \vee R) \wedge (Q \vee \neg R) \Rightarrow P \vee Q} \wedge L}{\Rightarrow (P \vee R) \wedge (Q \vee \neg R) \rightarrow P \vee Q} \rightarrow R$$

$$\frac{F, G, \Gamma \Rightarrow \Delta}{F \wedge G, \Gamma \Rightarrow \Delta} \wedge L$$

$$\frac{\frac{\overline{P \vee R, Q \vee \neg R \Rightarrow P \vee Q} \vee R}{(P \vee R) \wedge (Q \vee \neg R) \Rightarrow P \vee Q} \wedge L}{\Rightarrow (P \vee R) \wedge (Q \vee \neg R) \rightarrow P \vee Q} \rightarrow R$$

$$\frac{\Gamma \Rightarrow F, G, \Delta}{\Gamma \Rightarrow F \vee G, \Delta} \vee R$$

$$\frac{\frac{\frac{P \vee R, Q \vee \neg R \Rightarrow P, Q}{P \vee R, Q \vee \neg R \Rightarrow P \vee Q} \vee R}{(P \vee R) \wedge (Q \vee \neg R) \Rightarrow P \vee Q} \wedge L}{\Rightarrow (P \vee R) \wedge (Q \vee \neg R) \rightarrow P \vee Q} \rightarrow R$$

$$\frac{\Gamma \Rightarrow F, G, \Delta}{\Gamma \Rightarrow F \vee G, \Delta} \vee R$$

$$\begin{array}{c}
 \frac{P \vee R, Q \vee \neg R \Rightarrow P, Q}{P \vee R, Q \vee \neg R \Rightarrow P \vee Q} \vee R \\
 \frac{(P \vee R) \wedge (Q \vee \neg R) \Rightarrow P \vee Q}{\Rightarrow (P \vee R) \wedge (Q \vee \neg R) \rightarrow P \vee Q} \wedge L \\
 \Rightarrow (P \vee R) \wedge (Q \vee \neg R) \rightarrow P \vee Q \rightarrow R
 \end{array}$$

$$\frac{F, \Gamma \Rightarrow \Delta \quad G, \Gamma \Rightarrow \Delta}{F \vee G, \Gamma \Rightarrow \Delta} \vee L$$

$$\begin{array}{c}
\frac{P, Q \vee \neg R \Rightarrow P, Q}{\frac{P \vee R, Q \vee \neg R \Rightarrow P, Q}{\frac{P \vee R, Q \vee \neg R \Rightarrow P \vee Q} \vee R} \vee L} \\
\frac{\frac{(P \vee R) \wedge (Q \vee \neg R) \Rightarrow P \vee Q}{\Rightarrow (P \vee R) \wedge (Q \vee \neg R) \rightarrow P \vee Q} \wedge L} \rightarrow R} \\
\frac{F, \Gamma \Rightarrow \Delta \quad G, \Gamma \Rightarrow \Delta}{F \vee G, \Gamma \Rightarrow \Delta} \vee L
\end{array}$$

$$\frac{\frac{\frac{\overline{P, Q \vee \neg R \Rightarrow P, Q} \text{ Ax} \quad \frac{\overline{R, Q \vee \neg R \Rightarrow P, Q}}{\vee L}}{\frac{P \vee R, Q \vee \neg R \Rightarrow P, Q}{\vee R}}}{\frac{(P \vee R) \wedge (Q \vee \neg R) \Rightarrow P \vee Q}^{\wedge L}}{\Rightarrow (P \vee R) \wedge (Q \vee \neg R) \rightarrow P \vee Q} \rightarrow R}$$

$$\frac{\overline{A, \Gamma \Rightarrow A, \Delta} \text{ Ax}}$$

$$\begin{array}{c}
\frac{P, Q \vee \neg R \Rightarrow P, Q \quad Ax}{\frac{P \vee R, Q \vee \neg R \Rightarrow P, Q}{P \vee R, Q \vee \neg R \Rightarrow P \vee Q} \vee R} \vee L \quad \frac{R, Q \vee \neg R \Rightarrow P, Q}{\frac{(P \vee R) \wedge (Q \vee \neg R) \Rightarrow P \vee Q} \wedge L} \rightarrow R} \vee L \\
\frac{F, \Gamma \Rightarrow \Delta \quad G, \Gamma \Rightarrow \Delta}{F \vee G, \Gamma \Rightarrow \Delta} \vee L
\end{array}$$

$$\begin{array}{c}
\frac{P, Q \vee \neg R \Rightarrow P, Q \quad Ax \quad \frac{\frac{R, Q \Rightarrow P, Q \quad R, \neg R \Rightarrow P, Q}{R, Q \vee \neg R \Rightarrow P, Q} \vee L}{P \vee R, Q \vee \neg R \Rightarrow P, Q} \vee L}{\frac{P \vee R, Q \vee \neg R \Rightarrow P, Q}{P \vee R, Q \vee \neg R \Rightarrow P \vee Q} \vee R}{\frac{(P \vee R) \wedge (Q \vee \neg R) \Rightarrow P \vee Q}{\Rightarrow (P \vee R) \wedge (Q \vee \neg R) \rightarrow P \vee Q} \wedge L} \rightarrow R \\
\\
\frac{F, \Gamma \Rightarrow \Delta \quad G, \Gamma \Rightarrow \Delta}{F \vee G, \Gamma \Rightarrow \Delta} \vee L
\end{array}$$

$$\begin{array}{c}
\frac{\overline{P, Q \vee \neg R \Rightarrow P, Q} \text{ Ax} \quad \frac{\overline{R, Q \Rightarrow P, Q} \text{ Ax} \quad \overline{R, \neg R \Rightarrow P, Q}}{R, Q \vee \neg R \Rightarrow P, Q} \vee L}{R, Q \vee \neg R \Rightarrow P, Q} \vee L \\
\frac{\overline{P \vee R, Q \vee \neg R \Rightarrow P, Q}}{P \vee R, Q \vee \neg R \Rightarrow P \vee Q} \vee R \\
\frac{\overline{(P \vee R) \wedge (Q \vee \neg R) \Rightarrow P \vee Q} \wedge L}{\Rightarrow (P \vee R) \wedge (Q \vee \neg R) \rightarrow P \vee Q} \rightarrow R
\end{array}$$

$$\overline{A, \Gamma \Rightarrow A, \Delta} \quad \text{Ax}$$

Proof search properties

- ▶ For every logical operator (\neg etc) there is one left and one right rule
- ▶ Every formula in the premise of a rule is a subformula of the conclusion of the rule.
This is called the **subformula property**.
 \Rightarrow no need to guess anything when applying a rule backward
- ▶ Backward rule application terminates because one operator is removed in each step.

Instances of rules

Definition

An **instance** of a rule is the result of replacing Γ and Δ by multisets of concrete formulas and F and G by concrete formulas.

Example

$$\frac{\Rightarrow P \wedge Q, A, B}{\neg(P \wedge Q) \Rightarrow A, B}$$

is an instance of

$$\frac{\Gamma \Rightarrow F, \Delta}{\neg F, \Gamma \Rightarrow \Delta}$$

setting $F := P \wedge Q$, $\Gamma := \emptyset$, $\Delta := \{A, B\}$

Proof trees

Definition (Proof tree)

A **proof tree** is a tree whose nodes are sequents and where each parent-children fragment

$$\frac{S_1 \quad \dots \quad S_n}{S}$$

is an instance of a proof rule.

(\Rightarrow all leaves must be instances of axioms)

A sequent S is **provable** (or **derivable**) if there is a proof tree with root S .

We write $\vdash_G S$ to denote that S is derivable.

Proof trees

An alternative inductive definition of proof trees:

Definition (Proof tree)

If

$$\frac{S_1 \quad \dots \quad S_n}{S}$$

is an instance of a proof rule and

there are proof trees T_1, \dots, T_n with roots S_1, \dots, S_n

then

$$\frac{T_1 \quad \dots \quad T_n}{S}$$

is a proof tree (with root S).

What does $\Gamma \Rightarrow \Delta$ “mean”?

Definition

$$|\Gamma \Rightarrow \Delta| = \left(\bigwedge \Gamma \rightarrow \bigvee \Delta \right)$$

Example: $|\{A, B\} \Rightarrow \{P, Q\}| = (A \wedge B \rightarrow P \vee Q)$

Remember: $\bigwedge \emptyset = \top$ and $\bigvee \emptyset = \perp$

In the following slides we prove: $\vdash_G S$ iff $\models |S|$

Soundness

Lemma (Rule Equivalence)

For every rule $\frac{S_1 \dots S_n}{S}$

- ▶ $|S| \equiv |S_1| \wedge \dots \wedge |S_n|$
- ▶ $|S|$ is a tautology iff all $|S_i|$ are tautologies

Proof: Exercise.

Theorem (Soundness of \vdash_G)

If $\vdash_G S$ then $\models |S|$.

Proof by induction on the height of the proof tree for $\vdash_G S$.

Tree must end in rule instance

$$\frac{S_1 \dots S_n}{S}$$

If $n = 0$ then we vacuously have $\models |S_i|$ for all i .

If $n > 0$ then by IH we also have $\models |S_i|$ for all i .

So $\models |S_i|$ for all i , hence $\models |S|$ by Rule Equivalence.

Proof search = growing a proof tree from the root

To prove completeness we first examine the properties of the **proof search procedure**:

- ▶ Start from an initial sequent S_0
- ▶ At each stage we have some potentially *partial* proof tree with unproved leaves
- ▶ In each step, pick some unproved leaf S and some rule instance

$$\frac{S_1 \quad \dots \quad S_n}{S}$$

and extend the tree with that rule instance
(creating new unproved leaves S_1, \dots, S_n)

Proof search terminates if ...

- ▶ there are no more unproved leaves — **success**
- ▶ there is some unproved leaf where no rule applies — **failure**
By the rules, **that leaf is of the form**

$$P_1, \dots, P_k \Rightarrow Q_1, \dots, Q_l$$

where all P_i and Q_j are atoms, no $P_i = Q_j$, and no $P_i = \perp$.

Example (failed proof)

$$\frac{\frac{\overline{P \Rightarrow P} \quad Ax \quad Q \Rightarrow P}{P \vee Q \Rightarrow P} \vee L \quad \frac{P \Rightarrow Q \quad \overline{Q \Rightarrow Q} \quad Ax}{P \vee Q \Rightarrow Q} \vee L}{P \vee Q \Rightarrow P \wedge Q} \wedge R$$

Falsifying assignments?

Proof search always terminates

Lemma (Termination)

Proof search terminates from any initial sequent S_0 .

Proof

In every step, one logical operator is removed.

- ⇒ Size of sequent decreases by 1
- ⇒ Depth of proof tree is bounded by size of S_0
- ⇒ Construction of proof tree terminates. □

Observe: Breadth only bounded by $2^{\text{size of } S_0}$.

Proof search preserves equivalence

Lemma (Search Equivalence)

At each stage of the search process,

if S_1, \dots, S_k are the unproved leaves, then $|S_0| \equiv |S_1| \wedge \dots \wedge |S_k|$

Proof by induction on the number of search steps.

Initially trivially true (base case).

When applying a rule instance

$$\frac{U_1 \quad \dots \quad U_n}{S_i}$$

we have

$$|S_0| \equiv |S_1| \wedge \dots \wedge |S_i| \wedge \dots \wedge |S_k|$$

(by IH)

$$\equiv |S_1| \wedge \dots \wedge |S_{i-1}| \wedge |U_1| \wedge \dots \wedge |U_n| \wedge |S_{i+1}| \wedge \dots \wedge |S_k|$$

(by Lemma Rule Equivalence)

Completeness

Lemma

If proof search fails, $|S_0|$ is not a tautology.

Proof If proof search fails, there is some unproved leaf

$$S = P_1, \dots, P_k \Rightarrow Q_1, \dots, Q_l$$

where all P_i, Q_j atoms, no $P_i = Q_j$ and no $P_i = \perp$.

Any assignment \mathcal{A} with $\mathcal{A}(P_i) = 1$ (for all i)

and $\mathcal{A}(Q_j) = 0$ (for all j) satisfies $\mathcal{A}(|S|) = 0$.

Thus $\mathcal{A}(|S_0|) = 0$ by Lemma Search Equivalence. □

Because of soundness of \vdash_G :

Corollary

Starting with some fixed S_0 , proof search cannot both fail (for some choices) and succeed (for other choices).

\Rightarrow no need for backtracking upon failure!

Completeness

Theorem (Completeness)

If $\models S$ then $\vdash_G S$.

Proof by contraposition: if not $\vdash_G S$ then proof search must fail.
Therefore $\not\models S$.

Corollary

Proof search is a decision procedure: it always terminates and it succeeds iff $\models S$.

Multisets versus sets

Termination only because of multisets.

With sets, the principal formula may get duplicated:

$$\frac{\Gamma \Rightarrow F, \Delta}{\neg F, \Gamma \Rightarrow \Delta} \quad \neg L \quad \Gamma := \{\neg F\} \rightsquigarrow \frac{\neg F \Rightarrow F, \Delta}{\neg F \Rightarrow \Delta}$$

An alternative formulation of the set version:

$$\frac{\Gamma \setminus \{\neg F\} \Rightarrow F, \Delta}{\neg F, \Gamma \Rightarrow \Delta}$$

Gentzen used sequences (hence “sequent calculus”)

Admissible Rules and Cut Elimination

Admissible rules

Definition

A rule

$$\frac{S_1 \quad \dots \quad S_n}{S}$$

is **admissible** if $\vdash_G S_1, \dots, \vdash_G S_n$ together imply $\vdash_G S$.

\Rightarrow Admissible rules can be used in proofs like normal rules

Admissibility of

$$\frac{S_1 \quad \dots \quad S_n}{S}$$

can be shown semantically (using \vdash_G iff \models)

by proving that $\models |S_1|, \dots, \models |S_n|$ together imply $\models |S|$.

Proof theory is interested in **syntactic proofs** that show **how** to eliminate admissible rules.

Cut elimination rule

Theorem (Gentzen's *Hauptsatz*)

The cut elimination rule

$$\frac{\Gamma \Rightarrow F, \Delta \quad \Gamma, F \Rightarrow \Delta}{\Gamma \Rightarrow \Delta} \quad \textit{cut}$$

is admissible.

Proof Omitted.

Proofs with cut elimination can be much shorter than proofs without!

But: applying the rule needs creativity! (find the right F)

Intuitively: Proof of Gentzen's theorem shows how to replace creativity by calculation.

Many applications.

Tableaux Calculus

Propositional Logic

A compact version of sequent calculus

The idea

What's "wrong" with sequent calculus:

Why do we have to copy(?) Γ and Δ
with every rule application?

The answer: tableaux calculus.

The idea:

Describe *backward* sequent calculus rule application
but leave Γ and Δ implicit/shared

Comparison:

Sequent Proof is a tree labeled by sequents,
trees grow upwards

Tableaux Proof is a tree labeled by formulas,
trees grow downwards

Terminology: **tableau** = tableaux calculus proof tree

Tableaux rules (examples)

Notation: $+F \approx F$ occurs on the right of \Rightarrow
 $-F \approx F$ occurs on the left of \Rightarrow

<i>S.C.</i>	\rightsquigarrow	<i>Tab.</i>	\rightsquigarrow	<i>Effect</i>
$\frac{F, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \neg F, \Delta}$	\rightsquigarrow	$\frac{+\neg F}{-F}$	\rightsquigarrow	$\begin{array}{c} +\neg F \\ \\ -F \end{array}$
$\frac{\Gamma \Rightarrow F, G, \Delta}{\Gamma \Rightarrow F \vee G, \Delta}$	\rightsquigarrow	$\frac{+F \vee G}{+F}$ $+G$	\rightsquigarrow	$\begin{array}{c} +F \vee G \\ \\ +F \\ \\ +G \end{array}$
$\frac{\Gamma \Rightarrow F, \Delta \quad \Gamma \Rightarrow G, \Delta}{\Gamma \Rightarrow F \wedge G, \Delta}$	\rightsquigarrow	$\frac{+F \wedge G}{+F \mid +G}$	\rightsquigarrow	$\begin{array}{c} +F \wedge G \\ / \quad \backslash \\ +F \quad +G \end{array}$

Interpretation of tableaux rule

$$\frac{F}{FGH}$$

if F matches the formula at some node in the tableau
extend the end of some branch starting at that node
according to FGH .

Example

$- A \rightarrow B$

$- B \rightarrow C$

$- A$

$+ C$

$A \rightarrow B, B \rightarrow C, A \Rightarrow C$

From tableau to sequents:

- ▶ Every path from the root to a leaf in a tableau represents a sequent
- ▶ The set of all such sequents represents the set of leaves of the corresponding sequent calculus proof

⇒

- ▶ A branch is **closed** (proved) if both $+F$ and $-F$ occur on it or $-\perp$ occurs on it
- ▶ The root sequent is proved if all branches are closed

Algorithm to prove $F_1, \dots \Rightarrow G_1, \dots$:

1. Start with the tableau $-F_1, \dots, +G_1, \dots$
2. while there is an open branch do
 - pick some non-atomic formula on that branch,
 - extend the branch according to the matching rule

Termination

No formula needs to be used twice on the same branch.
But possibly on *different* branches:

$$\begin{array}{l} +\neg A \wedge \neg B \\ +A \vee B \end{array}$$

A formula occurrence in a tableau can be deleted
if it has been used in every unclosed branch
starting from that occurrence

Tableaux rules

$$\frac{-\neg F}{+F}$$

$$\frac{+\neg F}{-F}$$

$$\frac{-F \wedge G}{-F \quad -G}$$

$$\frac{+F \wedge G}{+F \mid +G}$$

$$\frac{-F \vee G}{-F \mid -G}$$

$$\frac{+F \vee G}{+F \quad +G}$$

$$\frac{-F \rightarrow G}{+F \mid -G}$$

$$\frac{+F \rightarrow G}{-F \quad +G}$$

Natural Deduction

Propositional Logic

(See the book by Troelstra and Schwichtenberg)

Natural deduction (Gentzen 1935) aims at *natural* proofs.

It formalizes good mathematical practice.

Resolution, but also sequent calculus, aim at proof search.

Main principles: Introduction and elimination rules

1. For every logical operator \oplus there are two kinds of rules:

▶ **Introduction rules:** How to prove $F \oplus G$

$$\frac{\dots}{F \oplus G}$$

▶ **Elimination rules:** What can be proved from $F \oplus G$

$$\frac{F \oplus G \quad \dots}{\dots}$$

Examples

$$\frac{A \quad B}{A \wedge B}$$

$\wedge I$

$$\frac{F \wedge G}{F}$$

$\wedge E_1$

$$\frac{F \wedge G}{G}$$

$\wedge E_2$

Main principles: Local assumptions

2. Proof can contain subproofs with *local/closed* assumptions

Example

Inference rule formalizing “if from the local assumption F we can prove G then we can prove $F \rightarrow G$ ”:

$$\frac{\begin{array}{c} [F] \\ \vdots \\ G \end{array}}{F \rightarrow G} \rightarrow I$$

A proof tree:

$$\frac{\frac{\begin{array}{c} [P] \quad Q \\ \hline P \wedge Q \end{array}}{P \rightarrow P \wedge Q} \rightarrow I}{\quad} \wedge I$$

“From the (open) assumption Q we can prove $P \rightarrow P \wedge Q$.”

In symbols: $Q \vdash_N P \rightarrow P \wedge Q$

Main principles: Growing the proof tree

Upwards:

Main principles: Growing the proof tree

Upwards:

$$\overline{P \rightarrow P \wedge Q}$$

Main principles: Growing the proof tree

Upwards:

$$\overline{P \rightarrow P \wedge Q} \quad \rightarrow I$$

Main principles: Growing the proof tree

Upwards:

$$\frac{\overline{P \wedge Q}}{P \rightarrow P \wedge Q} \rightarrow I$$

Main principles: Growing the proof tree

Upwards:

$$\frac{\overline{P \wedge Q}}{P \rightarrow P \wedge Q} \quad \begin{array}{l} \wedge I \\ \rightarrow I \end{array}$$

Main principles: Growing the proof tree

Upwards:

$$\frac{\frac{P \quad Q}{P \wedge Q}}{P \rightarrow P \wedge Q} \quad \begin{array}{l} \wedge I \\ \rightarrow I \end{array}$$

Main principles: Growing the proof tree

Upwards:

$$\frac{\frac{[P] \quad Q}{P \wedge Q}}{P \rightarrow P \wedge Q} \quad \begin{array}{l} \wedge I \\ \rightarrow I \end{array}$$

Main principles: Growing the proof tree

Upwards:

$$\frac{\frac{[P] \quad Q}{P \wedge Q}}{P \rightarrow P \wedge Q} \quad \begin{array}{l} \wedge I \\ \rightarrow I \end{array}$$

Downwards:

Main principles: Growing the proof tree

Upwards:

$$\frac{\frac{[P] \quad Q}{P \wedge Q}}{P \rightarrow P \wedge Q} \quad \begin{array}{l} \wedge I \\ \rightarrow I \end{array}$$

Downwards:

Main principles: Growing the proof tree

Upwards:

$$\frac{\frac{[P] \quad Q}{P \wedge Q}}{P \rightarrow P \wedge Q} \quad \wedge I \quad \rightarrow I$$

Downwards:

$$\frac{P \quad Q}{\quad}$$

Main principles: Growing the proof tree

Upwards:

$$\frac{\frac{[P] \quad Q}{P \wedge Q}}{P \rightarrow P \wedge Q} \quad \wedge I \rightarrow I$$

Downwards:

$$\frac{P \quad Q}{\quad} \quad \wedge I$$

Main principles: Growing the proof tree

Upwards:

$$\frac{\frac{[P] \quad Q}{P \wedge Q}}{P \rightarrow P \wedge Q} \quad \wedge I \rightarrow I$$

Downwards:

$$\frac{P \quad Q}{P \wedge Q} \quad \wedge I$$

Main principles: Growing the proof tree

Upwards:

$$\frac{\frac{[P] \quad Q}{P \wedge Q} \quad \wedge I}{P \rightarrow P \wedge Q} \rightarrow I$$

Downwards:

$$\frac{P \quad Q}{P \wedge Q} \quad \wedge I$$

Main principles: Growing the proof tree

Upwards:

$$\frac{\frac{[P] \quad Q}{P \wedge Q}}{P \rightarrow P \wedge Q} \quad \wedge I \rightarrow I$$

Downwards:

$$\frac{P \quad Q}{\frac{P \wedge Q}{P \rightarrow P \wedge Q}} \quad \wedge I \rightarrow I$$

Main principles: Growing the proof tree

Upwards:

$$\frac{\frac{[P] \quad Q}{P \wedge Q}}{P \rightarrow P \wedge Q} \quad \wedge I \rightarrow I$$

Downwards:

$$\frac{\frac{[P] \quad Q}{P \wedge Q}}{P \rightarrow P \wedge Q} \quad \wedge I \rightarrow I$$

ND proof trees

- ▶ The nodes of a ND proof tree are (labeled by) formulas.
- ▶ Leaf nodes are called **assumptions**.
- ▶ The root is called the **conclusion**.

- ▶ Assumptions can be **open** or **closed**.
- ▶ Closed assumptions are written **[F]**.
- ▶ $\Gamma \vdash_N F$ denotes that there is a proof tree with conclusion F whose open assumptions belong to the set of formulas Γ .
(Reading: F is provable (derivable) from Γ .)

Intuition:

- ▶ A proof tree shows that the **conjunction** of the **open** assumptions entails the conclusion.
- ▶ Closed assumptions are **auxiliary local** assumptions in a subproof that have been closed (“discharged”) by some proof rule like $\rightarrow I$.

ND proof trees

ND proof trees are defined inductively:

- ▶ Every formula F is a ND proof tree with open assumption F and conclusion F .
(Intuition: From F we can prove F .)
- ▶ Larger proof trees are constructed using the rules of ND:
 - Introduction and Elimination rules for $\wedge, \vee, \rightarrow, \neg$, plus
 - a rule for \perp .

The application of a rule (backwards or forwards) adds new nodes to the tree and possibly closes some assumptions:

Natural Deduction rules

$$\frac{F \quad G}{F \wedge G} \wedge I$$

$$\frac{F \wedge G}{F} \wedge E_1 \quad \frac{F \wedge G}{G} \wedge E_2$$

$$\frac{\begin{array}{c} [F] \\ \vdots \\ G \end{array}}{F \rightarrow G} \rightarrow I$$

$$\frac{F \rightarrow G \quad F}{G} \rightarrow E$$

$$\frac{F}{F \vee G} \vee I_1 \quad \frac{G}{F \vee G} \vee I_2$$

$$\frac{F \vee G \quad \begin{array}{c} [F] \\ \vdots \\ H \end{array} \quad \begin{array}{c} [G] \\ \vdots \\ H \end{array}}{H} \vee E$$

$$\frac{\begin{array}{c} [F] \\ \vdots \\ \perp \end{array}}{\neg F} \neg I$$

$$\frac{\neg F \quad F}{\perp} \neg E$$

$$\frac{\begin{array}{c} [\neg F] \\ \vdots \\ \perp \end{array}}{F} \perp$$

Natural Deduction rules

How to read a rule

$$\frac{\dots \quad \begin{array}{c} [F] \\ \vdots \\ G \end{array} \quad \dots}{\dots} \quad r$$

Forward:

When applying rule r , in the proof of G we can close all (or some) of the assumptions F .

Backward:

In the subproof of G we can use the local assumption $[F]$.

We can use labels to show which rule application closed which assumptions (the slides won't but you must!).

Examples of proofs

$P \rightarrow Q \vdash_N \neg Q \rightarrow \neg P$:

$$\frac{\frac{[P]^2 \quad P \rightarrow Q}{Q} \rightarrow E:4 \quad [\neg Q]^1}{\perp} \neg E:3}{\neg P} \neg I:2}{\neg Q \rightarrow \neg P} \rightarrow I:1$$

$\neg(P \vee Q) \vdash_N \neg P \wedge \neg Q$:

$$\frac{\frac{[P]^2}{P \vee Q} \vee I:4 \quad \neg(P \vee Q)}{\perp} \neg E:3}{\neg P} \neg I:2}{\neg P \wedge \neg Q} \wedge I:1 \quad \frac{\frac{[Q]^5}{P \vee Q} \vee I:7 \quad \neg(P \vee Q)}{\perp} \neg E:6}{\neg Q} \neg I:5}{\neg P \wedge \neg Q} \wedge I:1$$

Soundness

Lemma (Soundness)

If $\Gamma \vdash_N F$ then $\Gamma \models F$

Proof by induction on the depth of the proof tree for $\Gamma \vdash_N F$.

Base: The tree has only one node F and $F \in \Gamma$.

Step: Case analysis of first rule applied (upwards).

Case: first rule is $\frac{G \rightarrow F \quad G}{F} \rightarrow E$

Let \mathcal{A} arbitrary such that $\mathcal{A}(\Gamma) = 1$. We prove $\mathcal{A}(F) = 1$

IH: $\Gamma \models G \rightarrow F$ and $\Gamma \models G$

IH and $\mathcal{A}(\Gamma) = 1$ yields $\mathcal{A}(G \rightarrow F) = 1$ and $\mathcal{A}(G) = 1$.

So $\mathcal{A}(F) = 1$

Soundness

Case: first rule is

$$\frac{\begin{array}{c} [G] \\ \vdots \\ F \end{array}}{G \rightarrow F} \rightarrow I$$

To show: $\Gamma \models G \rightarrow F$

IH: $\Gamma, G \models F$

$\Gamma \models G \rightarrow F$

iff for all \mathcal{A} : $\mathcal{A} \models \Gamma \Rightarrow \mathcal{A} \models G \rightarrow F$

iff for all \mathcal{A} : $\mathcal{A} \models \Gamma \Rightarrow (\mathcal{A} \models G \Rightarrow \mathcal{A} \models F)$

iff for all \mathcal{A} : $(\mathcal{A} \models \Gamma \text{ and } \mathcal{A} \models G) \Rightarrow \mathcal{A} \models F$

iff IH

Towards completeness: ND can simulate truth tables

Lemma (Tertium non datur)

$\vdash_N F \vee \neg F$

Proof:

$$\frac{\frac{\frac{[\neg(F \vee \neg F)]^1}{\perp} \quad \frac{[\neg F]^4}{F \vee \neg F} \quad \vee I_2:6}{\perp} \quad \neg E:5}{\frac{\perp}{F} \quad \perp:4}{F \vee \neg F} \quad \vee I_1:3}{\frac{[\neg(F \vee \neg F)]^1}{\perp} \quad \neg E:2}{F \vee \neg F} \quad \perp:1}$$

Towards completeness: ND can simulate truth tables

Definition

$$F^{\mathcal{A}} := \begin{cases} F & \text{if } \mathcal{A}(F) = 1 \\ \neg F & \text{if } \mathcal{A}(F) = 0 \end{cases}$$

Lemma (1)

If $\text{atoms}(F) \subseteq \{A_1, \dots, A_n\}$ then $A_1^{\mathcal{A}}, \dots, A_n^{\mathcal{A}} \vdash_N F^{\mathcal{A}}$ for every \mathcal{A} .

Proof By induction on F .

Only the case $F = G \rightarrow H$. Three subcases:

$\mathcal{A}(H) = 1$. Then $H^{\mathcal{A}} = H$, $(G \rightarrow H)^{\mathcal{A}} = G \rightarrow H$.

To prove: $A_1^{\mathcal{A}}, \dots, A_n^{\mathcal{A}} \vdash_N G \rightarrow H$. By IH: $A_1^{\mathcal{A}}, \dots, A_n^{\mathcal{A}} \vdash_N H$.

$$\frac{\overline{H} \quad IH}{G \rightarrow H} \rightarrow I$$

Towards completeness: ND can simulate truth tables

$\mathcal{A}(G) = 0$. Then $G^{\mathcal{A}} = \neg G$, $(G \rightarrow H)^{\mathcal{A}} = G \rightarrow H$.

To prove: $A_1^{\mathcal{A}}, \dots, A_n^{\mathcal{A}} \vdash_N G \rightarrow H$. By IH: $A_1^{\mathcal{A}}, \dots, A_n^{\mathcal{A}} \vdash_N \neg G$.

$$\frac{[G] \quad \overline{\neg G} \quad \text{IH}}{\perp} \quad \neg E$$

$$\frac{\perp \quad \perp}{H} \quad \perp$$

$$\frac{H}{G \rightarrow H} \quad \rightarrow I$$

$\mathcal{A}(G) = 1$ and $\mathcal{A}(H) = 0$. Then $G^{\mathcal{A}} = G$, $H^{\mathcal{A}} = \neg H$,
 $(G \rightarrow H)^{\mathcal{A}} = \neg(G \rightarrow H)$.

To prove: $A_1^{\mathcal{A}}, \dots, A_n^{\mathcal{A}} \vdash_N \neg(G \rightarrow H)$.

By IH: $A_1^{\mathcal{A}}, \dots, A_n^{\mathcal{A}} \vdash_N G$ and $A_1^{\mathcal{A}}, \dots, A_n^{\mathcal{A}} \vdash_N \neg H$.

$$\frac{[G \rightarrow H] \quad \overline{G} \quad \text{IH}}{H} \quad \rightarrow E$$

$$\frac{H \quad \overline{\neg H} \quad \text{IH}}{\perp} \quad \perp$$

$$\frac{\perp}{\neg(G \rightarrow H)} \quad \rightarrow I$$

Towards completeness: ND can simulate truth tables

Corollary (Cases)

If $F, \Gamma \vdash_N G$ and $\neg F, \Gamma \vdash_N G$ then $\Gamma \vdash_N G$.

Proof: By Lemma (Tertium non datur) $F, \Gamma \vdash_N F \vee \neg F$.

Apply

$$\frac{F \vee \neg F \quad \begin{array}{c} [F] \\ \vdots \\ G \end{array} \quad \begin{array}{c} [\neg F] \\ \vdots \\ G \end{array}}{G} \quad \vee E$$

Completeness

Lemma (2)

If $\text{atoms}(F) = \{A_1, \dots, A_n\}$ and $\models F$ then $A_1^{\mathcal{A}}, \dots, A_k^{\mathcal{A}} \vdash_N F$ for every \mathcal{A} and for all $k \leq n$.

Proof by (downward) induction on $k = n, \dots, 0$.

$k = n$. $A_1^{\mathcal{A}}, \dots, A_n^{\mathcal{A}} \vdash_N F$ holds by Lemma (1) and $F^{\mathcal{A}} = F$ because F is valid.

$k < n$. By IH $A_1^{\mathcal{A}}, \dots, A_k^{\mathcal{A}} \vdash_N F$ for every \mathcal{A} .

To prove: $A_1^{\mathcal{A}}, \dots, A_{k-1}^{\mathcal{A}} \vdash_N F$ for every \mathcal{A} .

Let \mathcal{A} arbitrary. Define $\bar{\mathcal{A}}$ by $\bar{\mathcal{A}}(A_i) = \mathcal{A}(A_i)$ for every $i < k$ and $\bar{\mathcal{A}}(A_k) = 1 - \mathcal{A}(A_k)$.

Assume w.l.o.g. $\mathcal{A}(A_k) = 1$ (otherwise swap \mathcal{A} and $\bar{\mathcal{A}}$).

Then $A_i^{\bar{\mathcal{A}}} = A_i^{\mathcal{A}}$ for every $i < k$, and $A_k^{\bar{\mathcal{A}}} = \neg A_k^{\mathcal{A}}$.

Taking $F := \mathcal{A}_k^{\mathcal{A}}$, $\Gamma := A_1^{\mathcal{A}}, \dots, A_{k-1}^{\mathcal{A}}$, and $G := F$ in Corollary (Cases) yields $A_1^{\mathcal{A}}, \dots, A_{k-1}^{\mathcal{A}} \vdash_N F$.

Completeness

Theorem (Completeness)

If $\Gamma \models F$ then $\Gamma \vdash_N F$.

Proof Only for $\Gamma := G$ (general case left as exercise).

Assume $G \models F$.

We have $\models G \rightarrow F$ and so $\vdash_N G \rightarrow F$ by Lemma (2) with $n = 0$.

Applying

$$\frac{G \rightarrow F \quad G}{F} \rightarrow E$$

yields $G \vdash_N F$.

Hilbert Systems

Propositional Logic

(See the book by Troelstra and Schwichtenberg)

Easy to define, hard to use.
No context management.

Hilbert systems and proof trees

A Hilbert system for propositional logic consists of

- ▶ a set of axioms (formulas over formula variables)
- ▶ and a single inference rule, $\rightarrow E$ or modus ponens:

$$\frac{F \rightarrow G \quad F}{G} \rightarrow E$$

Nodes of a proof tree are (labeled with) formulas.

Proof trees are defined inductively:

- ▶ Every formula F is a proof tree.
- ▶ Larger proof trees are constructed using $\rightarrow E$ (and $\rightarrow E$ only).

$\Gamma \vdash_H F$ denotes that there is a proof tree with root F whose leaves are either instances of axioms or elements of Γ (assumptions).

Alternative presentation

Proofs in Hilbert systems are frequently shown as lists of lines

$$\begin{array}{l} 1: F_1 \quad \textit{justification}_1 \\ 2: F_2 \quad \textit{justification}_2 \\ \vdots \\ i: F_i \quad \textit{justification}_i \\ \vdots \\ n: F_n \quad \textit{justification}_n \end{array}$$

where $\textit{justification}_i$ is either $\left\{ \begin{array}{l} \textit{assumption}, \\ \textit{axiom}, \\ \rightarrow E (j, k), \text{ with } j, k < i. \end{array} \right.$

Notational convention:

$$F \rightarrow G \rightarrow H \text{ means } F \rightarrow (G \rightarrow H)$$

Note: $F \rightarrow G \rightarrow H \equiv F \wedge G \rightarrow H$
 $F \rightarrow G \rightarrow H \not\equiv (F \rightarrow G) \rightarrow H$

A simple Hilbert system

Axioms: $F \rightarrow G \rightarrow F$ (A1)

$(F \rightarrow G \rightarrow H) \rightarrow (F \rightarrow G) \rightarrow F \rightarrow H$ (A2)

A proof of $A \rightarrow A$:

A simple Hilbert system

Axioms: $F \rightarrow G \rightarrow F$ (A1)

$(F \rightarrow G \rightarrow H) \rightarrow (F \rightarrow G) \rightarrow F \rightarrow H$ (A2)

A proof of $A \rightarrow A$:

A simple Hilbert system

Axioms: $F \rightarrow G \rightarrow F$ (A1)

$(F \rightarrow G \rightarrow H) \rightarrow (F \rightarrow G) \rightarrow F \rightarrow H$ (A2)

A proof of $A \rightarrow A$:

A simple Hilbert system

Axioms: $F \rightarrow G \rightarrow F$ (A1)

$(F \rightarrow G \rightarrow H) \rightarrow (F \rightarrow G) \rightarrow F \rightarrow H$ (A2)

A proof of $A \rightarrow A$:

1 :

2 :

3 :

4 :

5 : $A \rightarrow A$

A simple Hilbert system

Axioms: $F \rightarrow G \rightarrow F$ (A1)

$(F \rightarrow G \rightarrow H) \rightarrow (F \rightarrow G) \rightarrow F \rightarrow H$ (A2)

A proof of $A \rightarrow A$:

1 :

2 :

3 : _____ $\rightarrow A \rightarrow A$

4 : _____

5 : $A \rightarrow A$ $\rightarrow E : 3, 4$

A simple Hilbert system

Axioms: $F \rightarrow G \rightarrow F$ (A1)

$(F \rightarrow G \rightarrow H) \rightarrow (F \rightarrow G) \rightarrow F \rightarrow H$ (A2)

A proof of $A \rightarrow A$:

1 :

2 :

3 : $(A \rightarrow A \rightarrow A) \rightarrow A \rightarrow A$

4 : $A \rightarrow A \rightarrow A$

5 : $A \rightarrow A$

$\rightarrow E : 3, 4$

A simple Hilbert system

Axioms: $F \rightarrow G \rightarrow F$ (A1)

$(F \rightarrow G \rightarrow H) \rightarrow (F \rightarrow G) \rightarrow F \rightarrow H$ (A2)

A proof of $A \rightarrow A$:

1 :

2 :

3 : $(A \rightarrow A \rightarrow A) \rightarrow A \rightarrow A$

4 : $A \rightarrow A \rightarrow A$

A1

5 : $A \rightarrow A$

$\rightarrow E : 3, 4$

A simple Hilbert system

Axioms: $F \rightarrow G \rightarrow F$ (A1)

$(F \rightarrow G \rightarrow H) \rightarrow (F \rightarrow G) \rightarrow F \rightarrow H$ (A2)

A proof of $A \rightarrow A$:

1: _____ $\rightarrow (A \rightarrow A \rightarrow A) \rightarrow A \rightarrow A$

2: _____

3: $(A \rightarrow A \rightarrow A) \rightarrow A \rightarrow A$ $\rightarrow E : 2, 1$

4: $A \rightarrow A \rightarrow A$ A1

5: $A \rightarrow A$ $\rightarrow E : 3, 4$

A simple Hilbert system

Axioms: $F \rightarrow G \rightarrow F$ (A1)

$(F \rightarrow G \rightarrow H) \rightarrow (F \rightarrow G) \rightarrow F \rightarrow H$ (A2)

A proof of $A \rightarrow A$:

1 : $(A \rightarrow (A \rightarrow A) \rightarrow A) \rightarrow (A \rightarrow A \rightarrow A) \rightarrow A \rightarrow A$

2 : $A \rightarrow (A \rightarrow A) \rightarrow A$

3 : $(A \rightarrow A \rightarrow A) \rightarrow A \rightarrow A$ $\rightarrow E : 2, 1$

4 : $A \rightarrow A \rightarrow A$ A1

5 : $A \rightarrow A$ $\rightarrow E : 3, 4$

A simple Hilbert system

Axioms: $F \rightarrow G \rightarrow F$ (A1)

$(F \rightarrow G \rightarrow H) \rightarrow (F \rightarrow G) \rightarrow F \rightarrow H$ (A2)

A proof of $A \rightarrow A$:

1 : $(A \rightarrow (A \rightarrow A) \rightarrow A) \rightarrow (A \rightarrow A \rightarrow A) \rightarrow A \rightarrow A$

2 : $A \rightarrow (A \rightarrow A) \rightarrow A$ A1

3 : $(A \rightarrow A \rightarrow A) \rightarrow A \rightarrow A$ $\rightarrow E : 2, 1$

4 : $A \rightarrow A \rightarrow A$ A1

5 : $A \rightarrow A$ $\rightarrow E : 3, 4$

A simple Hilbert system

Axioms: $F \rightarrow G \rightarrow F$ (A1)

$(F \rightarrow G \rightarrow H) \rightarrow (F \rightarrow G) \rightarrow F \rightarrow H$ (A2)

A proof of $A \rightarrow A$:

1 : $(A \rightarrow (A \rightarrow A) \rightarrow A) \rightarrow (A \rightarrow A \rightarrow A) \rightarrow A \rightarrow A$ A2

2 : $A \rightarrow (A \rightarrow A) \rightarrow A$ A1

3 : $(A \rightarrow A \rightarrow A) \rightarrow A \rightarrow A$ $\rightarrow E : 2, 1$

4 : $A \rightarrow A \rightarrow A$ A1

5 : $A \rightarrow A$ $\rightarrow E : 3, 4$

A simple Hilbert system

Axioms: $F \rightarrow G \rightarrow F$ (A1)

$(F \rightarrow G \rightarrow H) \rightarrow (F \rightarrow G) \rightarrow F \rightarrow H$ (A2)

A proof of $A \rightarrow A$:

1 : $(A \rightarrow (A \rightarrow A) \rightarrow A) \rightarrow (A \rightarrow A \rightarrow A) \rightarrow A \rightarrow A$ A2

2 : $A \rightarrow (A \rightarrow A) \rightarrow A$ A1

3 : $(A \rightarrow A \rightarrow A) \rightarrow A \rightarrow A$ $\rightarrow E : 2, 1$

4 : $A \rightarrow A \rightarrow A$ A1

5 : $A \rightarrow A$ $\rightarrow E : 3, 4$

$\Rightarrow \vdash_H A \rightarrow A$

A simple Hilbert system

Axioms: $F \rightarrow G \rightarrow F$ (A1)

$(F \rightarrow G \rightarrow H) \rightarrow (F \rightarrow G) \rightarrow F \rightarrow H$ (A2)

A proof of $A \rightarrow A$:

1 : $(A \rightarrow (A \rightarrow A) \rightarrow A) \rightarrow (A \rightarrow A \rightarrow A) \rightarrow A \rightarrow A$ A2

2 : $A \rightarrow (A \rightarrow A) \rightarrow A$ A1

3 : $(A \rightarrow A \rightarrow A) \rightarrow A \rightarrow A$ $\rightarrow E : 2, 1$

4 : $A \rightarrow A \rightarrow A$ A1

5 : $A \rightarrow A$ $\rightarrow E : 3, 4$

$\Rightarrow \vdash_H A \rightarrow A$

Observe: The same proof can be used to derive $F \rightarrow F$ for any formula F .

Theorem (Deduction Theorem)

In any Hilbert-system that contains the axioms A1 and A2:

$$F, \Gamma \vdash_H G \quad \text{iff} \quad \Gamma \vdash_H F \rightarrow G$$

Proof “ \Leftarrow ”: Assume $\Gamma \vdash_H F \rightarrow G$. We prove $F, \Gamma \vdash_H G$.

$$\begin{aligned} & \Gamma \vdash_H F \rightarrow G \\ \Rightarrow & F, \Gamma \vdash_H F \rightarrow G \\ \Rightarrow & F, \Gamma \vdash_H G \quad \text{by } \rightarrow E \text{ because } F, \Gamma \vdash_H F \end{aligned}$$

Theorem (Deduction Theorem)

In any Hilbert-system that contains the axioms A1 and A2:

$$F, \Gamma \vdash_H G \quad \text{iff} \quad \Gamma \vdash_H F \rightarrow G$$

Proof “ \Rightarrow ”: Assume $F, \Gamma \vdash_H G$ with a proof of length n .
We prove $\Gamma \vdash_H F \rightarrow G$ by induction on n .

Base: $n = 1$. Then either $G \in \Gamma \cup \{F\}$ or G is instance of axiom.

- ▶ $G = F$. To prove: $\Gamma \vdash_H F \rightarrow F$. Done earlier.
- ▶ $G \in \Gamma$ or instance of axiom.

1 : G

2 : $G \rightarrow F \rightarrow G$

A1

3 : $F \rightarrow G$

$\rightarrow E : 1, 2$

Theorem (Deduction Theorem)

In any Hilbert-system that contains the axioms A1 and A2:

$$F, \Gamma \vdash_H G \quad \text{iff} \quad \Gamma \vdash_H F \rightarrow G$$

Proof " \Rightarrow ": Assume $F, \Gamma \vdash_H G$ with a proof of length n .
We prove $\Gamma \vdash_H F \rightarrow G$ by induction on n .

Step: $n > 1$. Assume last $\rightarrow E$ gives G from $H \rightarrow G$ and H .

IH: $\Gamma \vdash_H F \rightarrow H$ and $\Gamma \vdash_H F \rightarrow H \rightarrow G$.

To prove: $\Gamma \vdash_H F \rightarrow G$.

$$\begin{array}{c} \text{A2: } (F \rightarrow H \rightarrow G) \rightarrow (F \rightarrow H) \rightarrow F \rightarrow G \quad F \rightarrow H \rightarrow G \\ \hline (F \rightarrow H) \rightarrow F \rightarrow G \quad F \rightarrow H \\ \hline F \rightarrow G \end{array}$$

Hilbert System

From now on \vdash_H refers to the following set of axioms:

$$F \rightarrow G \rightarrow F \quad (\text{A1})$$

$$(F \rightarrow G \rightarrow H) \rightarrow (F \rightarrow G) \rightarrow F \rightarrow H \quad (\text{A2})$$

$$F \rightarrow G \rightarrow F \wedge G \quad (\text{A3})$$

$$F \wedge G \rightarrow F \quad (\text{A4})$$

$$F \wedge G \rightarrow G \quad (\text{A5})$$

$$F \rightarrow F \vee G \quad (\text{A6})$$

$$G \rightarrow F \vee G \quad (\text{A7})$$

$$F \vee G \rightarrow (F \rightarrow H) \rightarrow (G \rightarrow H) \rightarrow H \quad (\text{A8})$$

$$(\neg F \rightarrow \perp) \rightarrow F \quad (\text{A9})$$

We prove soundness and completeness.

Relating Hilbert and Natural Deduction

Theorem (Hilbert can simulate ND)

If $\Gamma \vdash_N F$ then $\Gamma \vdash_H F$

Proof. Translation in two steps:

$$\vdash_N \overset{(1)}{\rightsquigarrow} \vdash_H + \rightarrow I \overset{(2)}{\rightsquigarrow} \vdash_H$$

1. Transform a ND-proof tree into a proof tree containing Hilbert axioms, $\rightarrow E$, and $\rightarrow I$ by replacing all other ND rules by Hilbert proofs with $\rightarrow I$

Principle: ND rule \rightsquigarrow 1 axiom + $\rightarrow I/E$

2. Eliminate $\rightarrow I$ rules using the Deduction Theorem

Theorem (ND can simulate Hilbert)

If $\Gamma \vdash_H F$ then $\Gamma \vdash_N F$

Proof by induction on the length of the Hilbert proof of F .

- ▶ Every Hilbert axiom is provable in ND (Exercise!).
- ▶ $\rightarrow E$ is also available in ND.

Corollary

$\Gamma \vdash_H F$ iff $\Gamma \vdash_N F$

Corollary (Soundness and completeness)

$\Gamma \vdash_H F$ iff $\Gamma \models F$

First-Order Predicate Logic Basics

Syntax of predicate logic: terms

A **variable** is a symbol of the form x_i where $i = 1, 2, 3, \dots$

A **function symbol** is of the form f_i^k where $i = 1, 2, 3, \dots$ and $k = 0, 1, 2, \dots$

A **predicate symbol** is of the form P_i^k where $i = 1, 2, 3, \dots$ and $k = 0, 1, 2, \dots$

We call i the **index** and k the **arity** of the symbol. Function

symbols of arity 0 are called **constant symbols**.

Instead of $f_i^0()$ we write f_i^0 .

Terms are inductively defined as follows:

1. Variables are terms.
2. If f is a function symbol of arity k and t_1, \dots, t_k are terms then $f(t_1, \dots, t_k)$ is a term.

Syntax of predicate logic: formulas

If P is a predicate symbol of arity k and t_1, \dots, t_k are terms then $P(t_1, \dots, t_k)$ is an **atomic formula**.

If $k = 0$ we write P instead of $P()$.

Formulas (of predicate logic) are inductively defined as follows:

- ▶ Every atomic formula is a formula.
- ▶ If F is a formula, then $\neg F$ is also a formula.
- ▶ If F and G are formulas, then $F \wedge G$, $F \vee G$ and $F \rightarrow G$ are also formulas.
- ▶ If x is a variable and F is a formula, then $\forall x F$ and $\exists x F$ are also formulas.
The symbols \forall and \exists are called the **universal** and the **existential quantifier**.

Syntax trees and subformulas

Syntax trees are defined as before, extended with the following trees for $\forall xF$ and $\exists xF$:

$$\begin{array}{cc} \forall x & \exists x \\ | & | \\ F & F \end{array}$$

Subformulas again correspond to subtrees.

Structural induction of formulas

Like for propositional logic but

- ▶ Different base case: $\mathcal{P}(P(t_1, \dots, t_k))$

- ▶ Two new induction steps:

prove $\mathcal{P}(\forall x F)$ under the induction hypothesis $\mathcal{P}(F)$

prove $\mathcal{P}(\exists x F)$ under the induction hypothesis $\mathcal{P}(F)$

Naming conventions

x, y, z, \dots instead of x_1, x_2, x_3, \dots

a, b, c, \dots for constant symbols

f, g, h, \dots for function symbols of arity > 0

P, Q, R, \dots instead of P_i^k

Precedence of quantifiers

Quantifiers have the same precedence as \neg

Example

$\forall x P(x) \wedge Q(x)$ abbreviates $(\forall x P(x)) \wedge Q(x)$
not $\forall x (P(x) \wedge Q(x))$

Similarly for \exists etc.

[This convention is not universal]

Free and bound variables, closed formulas

A variable x **occurs** in a formula F if it occurs in some atomic subformula of F .

An occurrence of a variable in a formula is either **free** or **bound**.

An occurrence of x in F is **bound** if it occurs in some subformula of F of the form $\exists xG$ or $\forall xG$; the smallest such subformula is the **scope** of the occurrence.

Otherwise the occurrence is **free**.

A formula without any free occurrence of any variable is **closed**.

Example

$$\forall x P(x) \rightarrow \exists y Q(a, x, y)$$

Exercise

	Closed?
$\forall x P(a)$	
$\forall x \exists y (Q(x, y) \vee R(x, y))$	Y
$\forall x Q(x, x) \rightarrow \exists x Q(x, y)$	N
$\forall x P(x) \vee \forall x Q(x, x)$	Y
$\forall x (P(y) \wedge \forall y P(x))$	N
$P(x) \rightarrow \exists x Q(x, f(x))$	N

	Formula?
$\exists x P(f(x))$	
$\exists f P(f(x))$	

Semantics of predicate logic: structures

A **structure** is a pair $\mathcal{A} = (U_{\mathcal{A}}, I_{\mathcal{A}})$

where $U_{\mathcal{A}}$ is an arbitrary, **nonempty** set called the **universe** of \mathcal{A} , and the **interpretation** $I_{\mathcal{A}}$ is a partial function that maps

- ▶ variables to elements of the universe $U_{\mathcal{A}}$,
- ▶ function symbols of arity k to functions of type $U_{\mathcal{A}}^k \rightarrow U_{\mathcal{A}}$,
- ▶ predicate symbols of arity k to functions of type $U_{\mathcal{A}}^k \rightarrow \{0, 1\}$ (predicates) [or equivalently to subsets of $U_{\mathcal{A}}^k$ (relations)]

$I_{\mathcal{A}}$ maps syntax (variables, functions and predicate symbols) to their meaning (elements, functions and predicates)

The special case of arity 0 can be written more simply:

- ▶ constant symbols are mapped to elements of $U_{\mathcal{A}}$,
- ▶ predicate symbols of arity 0 are mapped to $\{0, 1\}$.

Abbreviations:

$x^{\mathcal{A}}$ abbreviates $I_{\mathcal{A}}(x)$

$f^{\mathcal{A}}$ abbreviates $I_{\mathcal{A}}(f)$

$P^{\mathcal{A}}$ abbreviates $I_{\mathcal{A}}(P)$

Example

$$U_{\mathcal{A}} = \mathbb{N}$$

$$I_{\mathcal{A}}(P) = P^{\mathcal{A}} = \{(m, n) \mid m, n \in \mathbb{N} \text{ and } m < n\}$$

$$I_{\mathcal{A}}(Q) = Q^{\mathcal{A}} = \{m \mid m \in \mathbb{N} \text{ and } m \text{ is prime}\}$$

$$I_{\mathcal{A}}(f) \text{ is the successor function: } f^{\mathcal{A}}(n) = n + 1$$

$$I_{\mathcal{A}}(g) \text{ is the addition function: } g^{\mathcal{A}}(m, n) = m + n$$

$$I_{\mathcal{A}}(a) = a^{\mathcal{A}} = 2$$

$$I_{\mathcal{A}}(z) = z^{\mathcal{A}} = 3$$

Intuition: is $\forall x P(x, f(x)) \wedge Q(g(a, z))$ true in this structure?

Evaluation of a term in a structure

Definition

Let t be a term and let $\mathcal{A} = (U_{\mathcal{A}}, I_{\mathcal{A}})$ be a structure. \mathcal{A} is **suitable** for t if $I_{\mathcal{A}}$ is defined for all variables and function symbols occurring in t . The **value** of a term t in a suitable structure \mathcal{A} , denoted by $\mathcal{A}(t)$, is defined recursively:

$$\mathcal{A}(x) = x^{\mathcal{A}}$$

$$\mathcal{A}(c) = c^{\mathcal{A}}$$

$$\mathcal{A}(f(t_1, \dots, t_k)) = f^{\mathcal{A}}(\mathcal{A}(t_1), \dots, \mathcal{A}(t_k))$$

Definition

Let F be a formula and let $\mathcal{A} = (U_{\mathcal{A}}, I_{\mathcal{A}})$ be a structure. \mathcal{A} is **suitable** for F if $I_{\mathcal{A}}$ is defined for all predicate and function symbols occurring in F and for all variables occurring free in F .

Evaluation of a formula in a structure

Suitable structures for $\forall x(P(x) \rightarrow \exists yQ(x, y))$

$$\mathcal{A}_1 = (U_1, I_1)$$

$$U_1 = \mathbb{N}$$

$$P^{\mathcal{A}_1} = 2\mathbb{N}$$

$$Q^{\mathcal{A}_1} = \{(n, m) \mid n + m = 5\}$$

$$\mathcal{A}_3 = (U_3, I_3)$$

$$U_3 = \{a, b\}$$

$$P^{\mathcal{A}_3} = \{a\}$$

$$Q^{\mathcal{A}_3} = \{(a, b), (b, a), (b, b)\}$$

$$\mathcal{A}_2 = (U_2, I_2)$$

$$U_2 = \{0, 1, 2\}$$

$$P^{\mathcal{A}_2} = \{0\}$$

$$Q^{\mathcal{A}_2} = \{(n, m) \mid n \leq m\}$$

$$\mathcal{A}_4 = (U_4, I_4)$$

$$U_4 = \mathbb{N}$$

$$P^{\mathcal{A}_4} = \emptyset$$

$$Q^{\mathcal{A}_4} = \emptyset$$

Evaluation of a formula in a structure

Let \mathcal{A} be suitable for F . The (truth) value of F in \mathcal{A} , denoted by $\mathcal{A}(F)$, is defined recursively:

$$\mathcal{A}(\neg F), \mathcal{A}(F \wedge G), \mathcal{A}(F \vee G), \mathcal{A}(F \rightarrow G)$$

as for propositional logic.

$$\mathcal{A}(P(t_1, \dots, t_k)) = \begin{cases} 1 & \text{if } (\mathcal{A}(t_1), \dots, \mathcal{A}(t_k)) \in P^{\mathcal{A}} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathcal{A}(\forall x F) = \begin{cases} 1 & \text{if for every } d \in U_{\mathcal{A}}, (\mathcal{A}[d/x])(F) = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\mathcal{A}(\exists x F) = \begin{cases} 1 & \text{if for some } d \in U_{\mathcal{A}}, (\mathcal{A}[d/x])(F) = 1 \\ 0 & \text{otherwise} \end{cases}$$

Recall: $\mathcal{A}[d/x]$ coincides with \mathcal{A} except $x^{\mathcal{A}[d/x]} = d$.

Notes

- ▶ During the evaluation of a formulas in a structure, the structure stays unchanged except for the interpretation of the variables.
- ▶ If the formula is closed, the initial interpretation of the variables is irrelevant.

Coincidence Lemma

Lemma

Let \mathcal{A} and \mathcal{A}' be two structures that coincide on all free variables, on all function symbols and all predicate symbols that occur in F . Then $\mathcal{A}(F) = \mathcal{A}'(F)$.

Proof.

Exercise.



Relation to propositional logic

- ▶ Every propositional formula can be seen as a formula of predicate logic where the atom A_i is replaced by the atom P_i^0 .
- ▶ Conversely, every formula of predicate logic that does not contain quantifiers and variables can be seen as a formula of propositional logic by replacing atomic formulas by propositional atoms.

Example

$$F = Q(a) \vee \neg P(f(b), b) \wedge P(b, f(b))$$

can be viewed as the propositional formula

$$F' = A_1 \vee \neg A_2 \wedge A_3 .$$

Predicate logic with equality

Predicate logic
+
distinguished predicate symbol “=” of arity 2

Semantics: A structure \mathcal{A} of predicate logic with equality always maps the predicate symbol = to the identity relation:

$$\mathcal{A}(=) = \{(d, d) \mid d \in U_{\mathcal{A}}\}$$

Formalizing statements

“Formalizing” a statement in predicate logic means to give a formula F **and** a structure \mathcal{A} such that “the meaning of F in \mathcal{A} corresponds to the statement.” More precisely: every structure that extends \mathcal{A} is a model of F .

Intuitively, the symbols interpreted in \mathcal{A} are those that the formalizer assumes are known by whoever is going to read the formula. F may contain other symbols, but then F must define what they mean (see next slides).

Typically, the formalizer chooses names for the symbols that suggest their meaning. The structure is often omitted, because it is assumed to be known (**danger!**).

We give different formalizations of the statement

There are infinitely many prime numbers

Formalization I

If the meanings of “prime” and “greater-than” are known, then we can take:

$$F_1: \forall x \exists y (Pr(y) \wedge y > x)$$

$$\mathcal{A}_1: U^{\mathcal{A}_1} = \mathbb{N}$$

$$Pr^{\mathcal{A}_1} = \{n \in \mathbb{N} \mid n \text{ is prime}\}$$

$$>^{\mathcal{A}_1} = \{(n, m) \in \mathbb{N} \mid n > m\}$$

What if the meaning of “prime” is not known?

Formalization II

If the meaning of “divides” and “one” are known, then we can take:

$$F_2: \quad \forall x (Pr(x) \leftrightarrow \forall y (Dv(y, x) \rightarrow (y = x \vee y = one))) \\ \rightarrow \quad \forall x \exists y (Pr(y) \wedge y > x)$$

$$\mathcal{A}_2: \quad U^{A_2} = \mathbb{N} \\ Dv^{A_2} = \{(n, m) \in \mathbb{N} \mid n \text{ divides } m\} \\ >^{A_2} = \{(n, m) \in \mathbb{N} \mid n > m\} \\ one^{A_2} = 1$$

We are now stating “if we define prime numbers as ... then there are infinitely many prime numbers”.

What if the meaning of “divides” is not known?

Formalization III

If the meaning of “product” is known , then we can take

$$\begin{aligned} F_3: \quad & \forall x \forall y (Dv(x, y) \leftrightarrow \exists z \text{prod}(x, z) = y) \\ & \wedge \quad \forall x (Pr(x) \leftrightarrow (\forall y Dv(y, x) \rightarrow (y = x \vee y = \text{one}))) \\ & \rightarrow \quad \forall x \exists y (Pr(y) \wedge y > x) \end{aligned}$$

(the conjunction of the first two formulas implies the third)

$$\begin{aligned} \mathcal{A}_3: \quad & U^{\mathcal{A}_3} = \mathbb{N} \\ & >^{\mathcal{A}_3} = \{(n, m) \in \mathbb{N} \mid n > m\} \\ & \text{one}^{\mathcal{A}_3} = 1 \\ & \text{prod}^{\mathcal{A}_3}(n, m) = n \cdot m \end{aligned}$$

What if the meaning of “product” is not known ?

Formalization IV

If the meaning of “sum”, “successor”, “one” and “zero” is known, then we can take

$$\begin{aligned} F_4: \quad & \forall x \text{ prod}(x, \text{zero}) = \text{zero} \\ & \wedge \quad \forall x \forall y \text{ prod}(x, \text{succ}(y)) = \text{sum}(\text{prod}(x, y), y) \\ & \wedge \quad \forall x \forall y (Dv(x, y) \leftrightarrow \exists z \text{ prod}(x, z) = y) \\ & \wedge \quad \forall x (\text{Pr}(x) \leftrightarrow (\forall y Dv(y, x) \rightarrow (y = x \vee y = \text{one}))) \\ & \rightarrow \quad \forall x \exists y (\text{Pr}(y) \wedge y > x) \end{aligned}$$

\mathcal{A}_4 only defines $>$, *sum*, *succ*, *one*, *zero*.

Observe: *prod* is defined *inductively*. The definition is no longer a macro, in the sense that we cannot produce an “equivalent” formula without the symbol *prod*.

What if the meaning of “sum” is not known?

Formalization V

$$\begin{aligned} F_5: & \quad \forall x \text{ sum}(x, \text{zero}) = x \\ & \quad \wedge \quad \forall x \forall y \text{ sum}(x, \text{succ}(y)) = \text{succ}(\text{sum}(x, y)) \\ & \quad \wedge \quad \forall x \text{ prod}(x, \text{zero}) = \text{zero} \\ & \quad \wedge \quad \forall x \forall y \text{ prod}(x, \text{succ}(y)) = \text{sum}(\text{prod}(x, y), y) \\ & \quad \wedge \quad \forall x \forall y (\text{Div}(x, y) \leftrightarrow \exists z \text{ prod}(x, z) = y) \\ & \quad \wedge \quad \forall x (\text{Pr}(x) \leftrightarrow (\forall y \text{ Div}(y, x) \rightarrow (y = x \vee y = \text{one}))) \\ & \quad \rightarrow \quad \forall x \exists y (\text{Pr}(y) \wedge y > x) \end{aligned}$$

\mathcal{A}_5 only defines $>$, *succ*, *one*, *zero*.

What if the meaning of 'greater than' and "one" is not known?

Formalization VI

$$\begin{aligned} F_6: \quad & one = succ(zero) \\ & \wedge \quad \forall x \forall y (x > y \leftrightarrow \exists z \neg(z = zero) \wedge sum(y, z) = x) \\ & \wedge \quad \forall x sum(x, zero) = x \\ & \wedge \quad \forall x \forall y sum(x, succ(y)) = succ(sum(x, y)) \\ & \wedge \quad \forall x prod(x, zero) = zero \\ & \wedge \quad \forall x \forall y prod(x, succ(y)) = sum(prod(x, y), y) \\ & \wedge \quad \forall x \forall y (Div(x, y) \leftrightarrow \exists z prod(x, z) = y) \\ & \wedge \quad \forall x (Pr(x) \leftrightarrow (\forall y Div(y, x) \rightarrow (y = x \vee y = one))) \\ & \rightarrow \quad \forall x \exists y (Pr(y) \wedge y > x) \end{aligned}$$

\mathcal{A}_6 only defines *succ*, *zero*.

Model, validity, satisfiability

Like in propositional logic

Definition

We write $\mathcal{A} \models F$ to denote that the structure \mathcal{A} is suitable for the formula F and that $\mathcal{A}(F) = 1$.

Then we say that F is **true** in \mathcal{A} or that \mathcal{A} is a **model** of F .

If every structure suitable for F is a model of F ,
then we write $\models F$ and say that F is **valid**.

If F has at least one model then we say that F is **satisfiable**.

Exercise

V: valid S: satisfiable, but not valid U: unsatisfiable

	V	S	U
$\forall x P(a)$			
$\exists x (\neg P(x) \vee P(a))$			
$P(a) \rightarrow \exists x P(x)$			
$P(x) \rightarrow \exists x P(x)$			
$\forall x P(x) \rightarrow \exists x P(x)$			
$\forall x P(x) \wedge \neg \forall y P(y)$			

Consequence and equivalence

Like in propositional logic

Definition

A formula G is a **consequence** of a set of formulas M if every structure that is a model of all $F \in M$ and suitable for G is also a model of G . Then we write $M \models G$.

Two formulas F and G are (**semantically**) **equivalent** if every structure \mathcal{A} suitable for both F and G satisfies $\mathcal{A}(F) = \mathcal{A}(G)$. Then we write $F \equiv G$.

Exercise

1. $\forall x P(x) \vee \forall x Q(x, x)$
2. $\forall x (P(x) \vee Q(x, x))$
3. $\forall x (\forall z P(z) \vee \forall y Q(x, y))$

	Y	N
1 \models 2		
2 \models 3		
3 \models 1		

Exercise

1. $\exists y \forall x P(x, y)$
2. $\forall x \exists y P(x, y)$

	Y	N
1 \models 2		
2 \models 1		

Exercise

	Y	N
$\forall x \forall y F \equiv \forall y \forall x F$		
$\forall x \exists y F \equiv \exists x \forall y F$		
$\exists x \exists y F \equiv \exists y \exists x F$		
$\forall x F \vee \forall x G \equiv \forall x (F \vee G)$		
$\forall x F \wedge \forall x G \equiv \forall x (F \wedge G)$		
$\exists x F \vee \exists x G \equiv \exists x (F \vee G)$		
$\exists x F \wedge \exists x G \equiv \exists x (F \wedge G)$		

Exercise

	Y	N
$P(x) \equiv \exists x P(x)$		
$P(x) \equiv \forall x P(x)$		
$P(a) \equiv P(x)$		
$P(x) \equiv P(y)$		
$\forall x \forall y P(y) \equiv \forall x P(x)$		
$\exists x \forall y P(y) \equiv \forall x P(x)$		

Equivalences

Theorem

- $\neg\forall x F \equiv \exists x \neg F$
 $\neg\exists x F \equiv \forall x \neg F$
- If x does not occur free in G then:*
 $\forall x F \wedge G \equiv \forall x (F \wedge G)$
 $\forall x F \vee G \equiv \forall x (F \vee G)$
 $\exists x F \wedge G \equiv \exists x (F \wedge G)$
 $\exists x F \vee G \equiv \exists x (F \vee G)$
- $\forall x F \wedge \forall x G \equiv \forall x (F \wedge G)$
 $\exists x F \vee \exists x G \equiv \exists x (F \vee G)$
- $\forall x \forall y F \equiv \forall y \forall x F$
 $\exists x \exists y F \equiv \exists y \exists x F$

Replacement theorem

Just like for propositional logic it can be proved:

Theorem

Let $F \equiv G$. Let H be a formula with an occurrence of F as a subformula. Then $H \equiv H'$, where H' is the result of replacing an arbitrary occurrence of F in H by G .

First-Order Logic Normal Forms

Abbreviations

We return to the abbreviations used in connection with resolution:

$F_1 \rightarrow F_2$ abbreviates $\neg F_1 \vee F_2$

\top abbreviates $P_1^0 \vee \neg P_1^0$

\perp abbreviates $P_1^0 \wedge \neg P_1^0$

Substitution

- ▶ Substitutions are mappings from variables to terms.
- ▶ By $[t/x]$ we denote the substitution that replaces x by t .
- ▶ The notation $F[t/x]$ (“ F with t for x ”) denotes the result of replacing all **FREE** free occurrences of x in F by t .

Example: $(\forall x P(x) \wedge Q(x))[f(y)/x] = \forall x P(x) \wedge Q(f(y))$

- ▶ Similarly for substitutions in terms:
 $u[t/x]$ is the result of replacing x by t in term u .

Example: $(f(x))[g(x)/x] = f(g(x))$

Variable capture

If a term t of F contains a bound occurrence of a variable, substitution may lead to **variable capture**:

$$(\forall x P(x, y))[f(x)/y] = \forall x P(x, f(x))$$

Variable capture must be avoided

Substitution lemmas

Lemma

$$\mathcal{A}(u[t/x]) = (\mathcal{A}[\mathcal{A}(t)/x])(u).$$

Proof by structural induction on u .

Lemma (Substitution Lemma)

If t contains no variable bound in F then

$$\mathcal{A}(F[t/x]) = (\mathcal{A}[\mathcal{A}(t)/x])(F).$$

Proof by structural induction on F with the help of the lemma on terms.

Warning

The notation $.[./.]$ is heavily overloaded:

Substitution in syntactic objects

$F[G/A]$ in propositional logic

$F[t/x]$

$u[t/x]$ where u is a term

Function update

$\mathcal{A}[v/A]$ where \mathcal{A} is a propositional assignment

$\mathcal{A}[d/x]$ where \mathcal{A} is a structure and $d \in U_{\mathcal{A}}$

Overall goal

Transform any formula F of length m into a closed formula

$$\forall x_1 \dots \forall x_n G \quad \text{where } G \text{ is quantifier-free,}$$

of length $O(m)$ that is equisatisfiable with F .

Rectified Formulas

Definition

A formula is **rectified** if no variable occurs both bound and free and all quantifiers in the formula bind different variables.

Lemma

Let $F = Qx G$ be a formula where $Q \in \{\forall, \exists\}$.

Let y be a variable that does not occur in G .

Then $F \equiv Qy G[y/x]$.

Lemma

Every formula is equivalent to a rectified formula.

Example

$$\forall x P(x, y) \wedge \exists x \exists y Q(x, y) \equiv \forall x' P(x', y) \wedge \exists x \exists y' Q(x, y')$$

Prenex form

Definition

A formula is in **prenex form** if it has the form

$$Q_1 y_1 \dots Q_n y_n F$$

where $Q_i \in \{\exists, \forall\}$, $n \geq 0$, and F is quantifier-free.

Prenex form

Theorem

Every formula is equivalent to a rectified formula in prenex form (a formula in **RPF**).

Proof First construct an equivalent rectified formula.

Then pull the quantifiers to the front using the following equivalences from left to right as long as possible:

$$\neg\forall x F \equiv \exists x \neg F$$

$$\neg\exists x F \equiv \forall x \neg F$$

$$Qx F \wedge G \equiv Qx (F \wedge G)$$

$$F \wedge Qx G \equiv Qx (F \wedge G)$$

$$Qx F \vee G \equiv Qx (F \vee G)$$

$$F \vee Qx G \equiv Qx (F \vee G)$$

For the last four rules note that the formula is rectified!

Skolem form

The **Skolem form** of a formula F in RPF is the result of applying the following algorithm to F :

while F contains an existential quantifier **do**

Let $F = \forall y_1 \forall y_2 \dots \forall y_n \exists z G$

(the block of universal quantifiers may be empty)

Let f be a **fresh** function symbol of arity n
that does not occur in F .

$F := \forall y_1 \forall y_2 \dots \forall y_n G[f(y_1, y_2, \dots, y_n)/z]$

i.e. remove the outermost existential quantifier in F and
replace every occurrence of z in G by $f(y_1, y_2, \dots, y_n)$

Example

$\exists x \forall y \exists z \forall u \exists v P(x, y, z, u, v) \equiv$

Exercise

Which formulas are rectified, in prenex, or Skolem form?

	R	P	S
$\forall x (T(x) \vee C(x) \vee D(x))$			
$\exists x \exists y (C(y) \vee B(x, y))$			
$\neg \exists x C(x) \leftrightarrow \forall x \neg C(x)$			
$\forall x (C(x) \rightarrow S(x)) \rightarrow \forall y (\neg C(y) \rightarrow \neg S(y))$			

Skolem form

Theorem

A formula in RPF and its Skolem form are equisatisfiable.

Proof Show: Every iteration produces an equisatisfiable formula.
Let (for simplicity) $F = \forall y \exists z G$ and $F' = \forall y G[f(y)/z]$.

1. $F' \models F$, that is, every model of F' is a model of F .

Assume \mathcal{A} is suitable for F' and $\mathcal{A}(F') = 1$.

\Rightarrow for all $u \in U_{\mathcal{A}}$, $\mathcal{A}[u/y](G[f(y)/z]) = 1$

\Rightarrow for all $u \in U_{\mathcal{A}}$, $\mathcal{A}[u/y][f^{\mathcal{A}}(u)/z](G) = 1$

\Rightarrow for all $u \in U_{\mathcal{A}}$ there is a $v \in U_{\mathcal{A}}$ s.t. $\mathcal{A}[u/y][v/z](G) = 1$

\Rightarrow for all $u \in U_{\mathcal{A}}$, $\mathcal{A}(\exists z G) = 1$

$\Rightarrow \mathcal{A}(\forall y \exists z G) = 1$

Skolem form

Theorem

A formula in RPF and its Skolem form are equisatisfiable.

Proof Show: Every iteration produces an equisatisfiable formula.

Let (for simplicity) $F = \forall y \exists z G$ and $F' = \forall y G[f(y)/z]$.

2. If F has a model, so does F'

Assume \mathcal{A} is suitable for F and $\mathcal{A}(F) = 1$.

W.l.o.g. \mathcal{A} does not define f (because f is new).

\Rightarrow for all $u \in U_{\mathcal{A}}$ there is $v \in U_{\mathcal{A}}$ s.t. $\mathcal{A}[u/y][v/z](G) = 1$ (*)

Let \mathcal{A}' be \mathcal{A} extended with a definition of f : $f^{\mathcal{A}'}(u) := v$, where v is chosen as in (*).

$\Rightarrow \mathcal{A}'(F') = 1$ because for all $u \in U_{\mathcal{A}}$:

$$\mathcal{A}'[u/y](G[f(y)/z]) = \mathcal{A}'[u/y][f^{\mathcal{A}'}[u/y](u)/z](G) \quad (\text{subs. lemma})$$

$$= \mathcal{A}'[u/y][f^{\mathcal{A}'}(u)/z](G) \quad (\text{def. of } \mathcal{A}')$$

$$= \mathcal{A}'[u/y][v/z](G) = 1 \quad (\text{def. of } \mathcal{A}' \text{ and } (*))$$

Summary: conversion to Skolem form

Input: a formula F

Output: a rectified, closed formula in Skolem form $\forall y_1 \dots \forall y_k G$,
where G is quantifier-free, that is equisatisfiable with F .

1. Rectify F by systematic renaming of bound variables.
The result is a formula F_1 equivalent to F .
2. Let y_1, y_2, \dots, y_n be the variables occurring free in F_1 .
Produce the formula $F_2 = \exists y_1 \exists y_2 \dots \exists y_n F_1$.
 F_2 is equisatisfiable with F_1 , rectified and closed.
3. Produce a formula F_3 in RPF equivalent to F_2 .
4. Eliminate the existential quantifiers in F_3
by transforming F_3 into its Skolem form F_4 .
The formula F_4 is equisatisfiable with F_3 .

Exercise

Convert into Skolem form $F = \forall x P(y, f(x, y)) \vee \neg \forall y Q(g(x), y)$

First-Order Logic

Herbrand Theory

Herbrand universe

The **Herbrand universe** $T(F)$ of a **closed** formula F in Skolem form is the set of all terms that can be constructed using the function symbols in F (including the constants!).

In the special case that F contains no constants, we first pick an arbitrary constant, say a , and then construct the terms.

Formally, $T(F)$ is inductively defined as follows:

- ▶ All constants occurring in F belong to $T(F)$; if no constant occurs in F , then $a \in T(F)$ for an arbitrary constant a .
- ▶ For every n -ary function symbol f occurring in F , if $t_1, t_2, \dots, t_n \in T(F)$ then $f(t_1, t_2, \dots, t_n) \in T(F)$.

Note: All terms in $T(F)$ are variable-free by construction!

Example

$$T(\forall x \forall y P(f(x), g(c, y))) = \{c, f(c), g(c, c), f(g(c, c)), \dots\}.$$

Herbrand structure

Let F be a closed formula in Skolem form. A structure \mathcal{A} suitable for F is a **Herbrand structure** for F if it satisfies the following conditions:

- ▶ $U^{\mathcal{A}} = T(F)$, and
- ▶ for every n -ary function symbol f occurring in F and every $t_1, \dots, t_n \in T(F)$: $f^{\mathcal{A}}(t_1, \dots, t_n) = f(t_1, \dots, t_n)$.

Fact

If \mathcal{A} is a Herbrand structure, then $\mathcal{A}(t) = t$ for all $t \in U^{\mathcal{A}}$.

A **Herbrand model** of F is a Herbrand structure suitable for F that is model of F .

Matrix of a formula

Definition

The **matrix** of a formula F is the result of removing all quantifiers (all $\forall x$ and $\exists x$) from F . The matrix is denoted by F^* .

Fundamental theorem of predicate logic

Theorem

A closed formula in Skolem form is satisfiable iff it has a Herbrand model.

Proof (\Leftarrow): If a formula has a model then it is satisfiable.

(\Rightarrow): Let \mathcal{A} be a model of a closed formula F in Skolem form. We define a Herbrand structure \mathcal{T} suitable for F :

Universe: $U_{\mathcal{T}} = T(F)$

Function symbols: $f^{\mathcal{T}}(t_1, \dots, t_n) = f(t_1, \dots, t_n)$

(If F contains no constant, then

$a^{\mathcal{A}} = u$ for some arbitrary $u \in U^{\mathcal{A}}$)

Predicate symbols: $(t_1, \dots, t_n) \in P^{\mathcal{T}}$ iff $(\mathcal{A}(t_1), \dots, \mathcal{A}(t_n)) \in P^{\mathcal{A}}$

Claim: \mathcal{T} is also a model of F .

Claim: \mathcal{T} is also a model of F .

We prove a stronger assertion:

For every closed formula G in Skolem form that contains the same function and predicate symbols as F , if $\mathcal{A} \models G$ then $\mathcal{T} \models G$

Proof By induction on the number n of universal quantifiers of G .

Basis: $n = 0$. Then G has no quantifiers at all.

Hence, G is a boolean combination of atomic formulas without variables.

So $\mathcal{A}(G) = \mathcal{T}(G)$ (why?), and we are done.

Step: $n > 0$. Let $G = \forall x H$.

$$\mathcal{A} \models G$$

$$\Rightarrow \text{for every } u \in U^{\mathcal{A}}: \mathcal{A}[u/x](H) = 1$$

$$\Rightarrow \text{for every } u \in U^{\mathcal{A}} \text{ s.t. } u = \mathcal{A}(t) \text{ for} \\ \text{some } t \in T(F): \mathcal{A}[u/x](H) = 1$$

$$\Rightarrow \text{for every } t \in T(F): \mathcal{A}[\mathcal{A}(t)/x](H) = 1$$

$$\Rightarrow \text{for every } t \in T(F): \mathcal{A}(H[t/x]) = 1 \quad (\text{Subst. Lemma})$$

$$\Rightarrow \text{for every } t \in T(F): \mathcal{T}(H[t/x]) = 1 \quad (\text{IH})$$

$$\Rightarrow \text{for every } t \in T(F): \mathcal{T}[\mathcal{T}(t)/x](H) = 1 \quad (\text{Subst. Lemma})$$

$$\Rightarrow \text{for every } t \in T(F): \mathcal{T}[t/x](H) = 1 \quad (\mathcal{T} \text{ is Herbrand struct.})$$

$$\Rightarrow \mathcal{T}(\forall x H) = 1 \quad (U^{\mathcal{T}} = T(F))$$

$$\Rightarrow \mathcal{T} \models G$$

Example

Let F be given by

$$F = \forall x (x > \mathbf{0} \rightarrow \exists y (x > y \wedge y > \mathbf{0})) .$$

The Skolem form of F is

$$G = \forall x (x > \mathbf{0} \rightarrow (x > f(x) \wedge f(x) > \mathbf{0})) .$$

The following structure \mathcal{A} is a model of G :

$$\mathcal{U}^{\mathcal{A}} = \mathbb{R}$$

$$\mathbf{0}^{\mathcal{A}} = 0$$

$$p >^{\mathcal{A}} q \Leftrightarrow p > q \quad \text{for every } p, q \in \mathbb{R}$$

$$f^{\mathcal{A}}(p) = p/2 \quad \text{for every } p \in \mathbb{R}$$

Which is the Herbrand structure \mathcal{T} given by the proof of the fundamental theorem?

Example

The Herbrand structure \mathcal{T} is given by:

$$\mathcal{U}^{\mathcal{T}} = \mathcal{T}(G) = \{\mathbf{0}, f(\mathbf{0}), f(f(\mathbf{0})), \dots\} = \{f^k(\mathbf{0}) \mid k \geq 0\}$$

$$f^{\mathcal{T}}(f^k(\mathbf{0})) = f(f^k(\mathbf{0})) = f^{k+1}(\mathbf{0})$$

$$\begin{aligned} f^k(\mathbf{0}) >^{\mathcal{T}} f^{\ell}(\mathbf{0}) &\Leftrightarrow (f^k(\mathbf{0}))^{\mathcal{A}} >^{\mathcal{A}} (f^{\ell}(\mathbf{0}))^{\mathcal{A}} \\ &\Leftrightarrow (f^{\mathcal{A}})^k(\mathbf{0}^{\mathcal{A}}) >^{\mathcal{A}} (f^{\mathcal{A}})^{\ell}(\mathbf{0}^{\mathcal{A}}) \\ &\Leftrightarrow (f^{\mathcal{A}})^k(0) > (f^{\mathcal{A}})^{\ell}(0) \\ &\Leftrightarrow 0/2^k > 0/2^{\ell} \\ &\Leftrightarrow \text{false} \end{aligned}$$

The theorem guarantees that \mathcal{T} is also a model of G . This is indeed the case because in \mathcal{T} the premise $x > \mathbf{0}$ of the implication is always false.

We have just shown:

Theorem

Let F be a closed formula in Skolem form.

Then F is satisfiable iff it has a Herbrand model.

What goes wrong if F is not closed or not in Skolem form?

Herbrand expansion

Let $F = \forall y_1 \dots \forall y_n F^*$ be a closed formula in Skolem form.

The **Herbrand expansion** of F is the set of formulas

$$E(F) = \{F^*[t_1/y_1] \dots [t_n/y_n] \mid t_1, \dots, t_n \in T(F)\}$$

Informally: the formulas of $E(F)$ are the result of substituting terms from $T(F)$ for the variables of F^* in every possible way.

Example

Some elements of $E(\forall x \forall y P(f(x), g(c, y)))$:

$$\begin{aligned} &P(f(c), g(c, c)) \quad P(f^2(c), g(c, c)) \quad P(f(c), g(c, f(c))) \\ &P(f^8(c), g(c, c)) \quad P(f(g(f(c), f(c))), g(c, f(g(c, f(c)))))) \end{aligned}$$

Note: The Herbrand expansion can be viewed as a set of propositional formulas over the set of variable-free atomic formulas.

Gödel-Herbrand-Skolem Theorem

Theorem

A closed formula F in Skolem form is satisfiable iff its Herbrand expansion $E(F)$ is satisfiable (in the sense of propositional logic).

Proof. By the fundamental theorem, it suffices to show that F has a Herbrand model iff $E(F)$ is satisfiable.

Let $F = \forall y_1 \dots \forall y_n F^*$.

\mathcal{A} is a Herbrand model of F

iff for all $t_1, \dots, t_n \in T(F)$, $\mathcal{A}[t_1/y_1] \dots [t_n/y_n](F^*) = 1$

iff for all $t_1, \dots, t_n \in T(F)$, $\mathcal{A}(F^*[t_1/y_1] \dots [t_n/y_n]) = 1$

iff for all $G \in E(F)$, $\mathcal{A}(G) = 1$

iff \mathcal{A} is a model of $E(F)$

Example

Let $F = \forall x (P(x) \vee Q(f(x)))$.

Herbrand universe:

$$T(F) = \{f^k(a) \mid k \geq 0\} = \{a, f(a), f(f(a)), \dots\}$$

Herbrand expansion:

$$\begin{aligned} E(F) &= \{P(f^k(a)) \vee Q(f^{k+1}(a)) \mid k \geq 0\} \\ &= \{P(a) \vee Q(f(a)), P(f(a)) \vee Q(f^2(a)), P(f^2(a)) \vee Q(f^3(a)), \dots\} \end{aligned}$$

\mathcal{A} is a Herbrand model of F

iff for all $k \geq 0$, $\mathcal{A}[f^k(a)/x](P(x) \vee Q(f(x))) = 1$

iff for all $k \geq 0$, $\mathcal{A}(P(x) \vee Q(f(x)))[f^k(a)/x] = 1$

iff for all $k \geq 0$, $\mathcal{A}(P(f^k(a)) \vee Q(f^{k+1}(a))) = 1$

iff \mathcal{A} is a model of $E(F)$

Herbrand's Theorem

Theorem

A closed formula F in Skolem form is unsatisfiable iff some finite subset of $E(F)$ is unsatisfiable.

Proof. Follows immediately from the Gödel-Herbrand-Skolem Theorem and the Compactness Theorem.

Example

We show that

$$F = \exists x \forall y P(x, y) \rightarrow \forall y \exists x P(x, y)$$

is valid, or, equivalently, that

$$\neg F \equiv \exists x \forall y P(x, y) \wedge \exists y \forall x \neg P(x, y)$$

is unsatisfiable.

Rectified form: $\exists x \forall y P(x, y) \wedge \exists z \forall v \neg P(v, z)$

Prenex form: $\exists x \exists z \forall y \forall v (P(x, y) \wedge \neg P(v, z))$

Skolem form: $\forall y \forall v (P(a, y) \wedge \neg P(v, b))$

Herbrand universe: $\{a, b\}$

Herbrand expansion: $\{ P(a, a) \wedge \neg P(a, b) , P(a, a) \wedge \neg P(b, b) , \\ P(a, b) \wedge \neg P(a, b) , P(a, b) \wedge \neg P(b, b) \}$

Gilmore's Algorithm

Let F be a closed formula in Skolem form
and let F_1, F_2, F_3, \dots be a computable enumeration of $E(F)$.

```
Input:  $F$   
 $n := 0$ ;  
repeat  $n := n + 1$ ;  
until  $(F_1 \wedge F_2 \wedge \dots \wedge F_n)$  is unsatisfiable;  
return "unsatisfiable"
```

The algorithm terminates iff F is unsatisfiable.

Semi-decidability Theorems

Theorem

- (a) *The unsatisfiability problem of predicate logic is (only) semi-decidable.*
- (b) *The validity problem of predicate logic is (only) semi-decidable.*

Proof. (a) Gilmore's algorithm is a semi-decision procedure.

(The problem is undecidable. Proof later)

(b) F valid iff $\neg F$ unsatisfiable.

Löwenheim-Skolem Theorem

Theorem

Every satisfiable formula of first-order predicate logic has a model with a countable universe.

Proof Let F_0 be a formula with free variables x_1, \dots, x_n for $n \geq 0$. Define $F := \exists x_1 \dots \exists x_n F_0$ and observe that F_0 has a model with universe U iff F has a model with universe U .

Let G be closed formula in Skolem form equisatisfiable with F as produced by the Normal Form transformations starting with F .

Fact: Every model of G is a model of F .

F_0 satisfiable $\Rightarrow F$ satisfiable
 $\Rightarrow G$ satisfiable
 $\Rightarrow G$ has a Herbrand model
 $\Rightarrow F$ has a model with universe $T(G)$
 $\Rightarrow F_0$ has a model with universe $T(G)$
 $\Rightarrow F_0$ has a model with countable universe
($T(G)$ is countable)

Löwenheim-Skolem Theorem

Formulas of first-order logic cannot enforce uncountable models

First-Order Logic Resolution

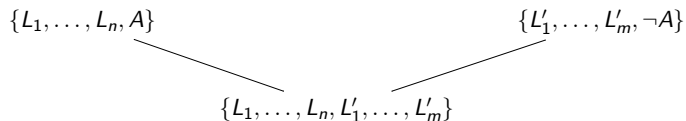
Resolution for first-order logic

Gilmore's algorithm is correct and complete,
but useless in practice.

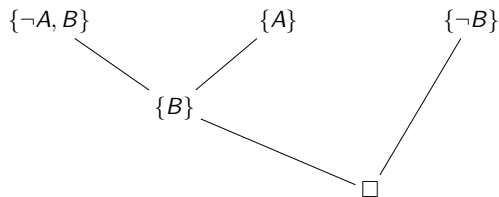
We upgrade resolution to make it work for predicate logic.

Recall: resolution in propositional logic

Resolution step:



Resolution graph:



A set of clauses is **unsatisfiable** iff the **empty clause** can be derived.

Adapting Gilmore's Algorithm

Gilmore's Algorithm:

Let F be a closed formula in Skolem form
and let F_1, F_2, F_3, \dots be an enumeration of $E(F)$.

$n := 0$;

repeat $n := n + 1$

until $(F_1 \wedge F_2 \wedge \dots \wedge F_n)$ is unsatisfiable;

– *this can be checked with any calculus for propositional logic*

return “unsatisfiable”

“any calculus” \rightsquigarrow use **resolution** for the unsatisfiability test

Terminology

Literal/clause/CNF is defined as for propositional logic but with the atomic formulas of predicate logic.

A **ground term/formula/etc** is a term/formula/etc that does not contain any variables.

An **instance** of a term/formula/etc is the result of applying a substitution to a term/formula/etc.

A **ground instance** is an instance that does not contain any variables.

Clause Herbrand expansion

Let $F = \forall y_1 \dots \forall y_n F^*$ be a closed formula in Skolem form with F^* in CNF, and let C_1, \dots, C_m be the clauses of F^* .

The **clause Herbrand expansion** of F is the set of ground clauses

$$CE(F) = \bigcup_{i=1}^m \{C_i[t_1/y_1] \dots [t_n/y_n] \mid t_1, \dots, t_n \in T(F)\}$$

Lemma

$CE(F)$ is unsatisfiable iff $E(F)$ is unsatisfiable.

Proof. Informally speaking, " $CE(F) \equiv E(F)$ ".

Ground resolution algorithm

Let F be a closed formula in Skolem form with F^* in CNF.

Let C_1, C_2, C_3, \dots be an enumeration of $CE(F)$.

```
 $n := 0;$   
 $S := \emptyset;$   
repeat  
     $n := n + 1;$   
     $S := S \cup \{C_n\};$   
until  $S \vdash_{Res} \square$   
return “unsatisfiable”
```

Ground resolution algorithm

Note: For example, $CE(F)$ can be enumerated according to the size of the substitutions.

Let $F = \forall y_1 \dots \forall y_n F^*$ and let C_1, \dots, C_m be the clauses of F^* . For every $s \geq 0$, define

$$\mathcal{C}_s = \bigcup_{i=1}^m \left\{ C_i[t_1/y_1] \dots [t_n/y_n] \left| \begin{array}{l} t_1, \dots, t_n \in T(F) \\ \text{and} \\ |t_1| + \dots + |t_n| = s \end{array} \right. \right\}$$

\mathcal{C}_s is finite for every $s \geq 0$ and $CE(F) = \bigcup_{s=0}^{\infty} \mathcal{C}_s$.

So $CE(F)$ can be enumerated by enumerating $\mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2, \dots$.

Note: The search for \square can be performed incrementally every time S is extended, keeping the clauses generated in previous steps.

Ground resolution theorem

The correctness of the ground resolution algorithm can be rephrased as follows:

Theorem

A formula $F = \forall y_1 \dots \forall y_n F^$ with F^* in CNF is unsatisfiable iff there is a sequence of ground clauses $C_1, \dots, C_m = \square$ such that for every $i = 1, \dots, m$*

- ▶ *either C_i is a ground instance of a clause $C \in F^*$,
i.e. $C_i = C[t_1/y_1] \dots [t_n/y_n]$ where $t_1, \dots, t_n \in T(F)$,*
- ▶ *or C_i is a resolvent of two clauses C_a, C_b with $a < i$ and $b < i$*

Beyond ground resolution: Intuition

Blind enumeration of ground clauses is extremely inefficient

$$F^* = \{ \{P(x)\}, \{\neg P(f(g(a, y))), Q(y)\}, \{\neg Q(g(f(z), f(z)))\} \}.$$

The algorithm can derive \square from just three ground clauses:

$$\begin{aligned} & \{P(f(g(a, g(f(a), f(a))))))\} \\ & \{\neg P(f(g(a, g(f(a), f(a))))), Q(g(f(a), f(a)))\} \\ & \{\neg Q(g(f(a), f(a)))\} \end{aligned}$$

Blind enumeration will generate the third clause early on, but it will only generate the first two after many (many!) superfluous clauses.

Beyond ground resolution: Intuition

Better: guided search with “lazy” substitutions

$$F^* = \{ \{P(x)\}, \{\neg P(f(g(a, y))), Q(y)\}, \{\neg Q(g(f(z), f(z)))\} \}$$

When resolving the first two clauses, **delay** the choice of substitution for x .

Commit only to replacing x by

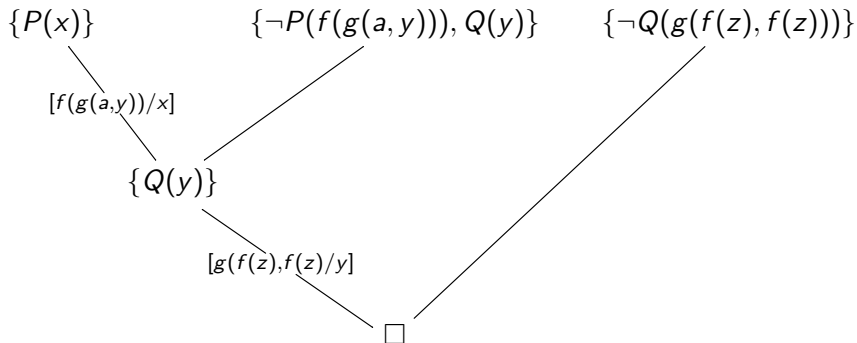
$$f(g(a, \text{whatever-}y\text{-will-be-later-replaced-by}))$$

For this:

- ▶ Allow substitutions with variables: $[f(g(a, y))/x]$.
- ▶ Apply substitutions only to two clauses that enable a new resolution step.

Beyond ground resolution: Intuition

$$F^* = \{ \{P(x)\}, \{\neg P(f(g(a, y))), Q(y)\}, \{\neg Q(g(f(z), f(z)))\} \}$$



Substitutions as functions

Substitutions are functions from variables to terms:

$[t/x]$ maps x to t (and all other variables to themselves)

Functions can be composed.

Composition of substitutions is denoted by juxtaposition:

$[t_1/x][t_2/y]$ first substitutes t_1 for x and then substitutes t_2 for y .

Example

$$(P(x, y))[f(y)/x][b/y] = (P(f(y), y))[b/y] = P(f(b), b)$$

A composition of substitutions is again a substitution. $\sigma_1\sigma_2$ is the substitution that applies σ_1 first and then σ_2 .

Substitutions are functions. Therefore

$$\sigma_1 = \sigma_2 \quad \text{iff} \quad x\sigma_1 = x\sigma_2 \text{ for all variables } x$$

Substitutions as functions

Definition

The **domain** of a substitution σ is $dom(\sigma) = \{x \mid x\sigma \neq x\}$

Example

$$dom([a/x][b/y]) = \{x, y\}$$

Substitutions are defined to have **finite domain**, and so every substitution can be written as a

simultaneous substitution $[t_1/x_1, \dots, t_n/x_n]$.

Unifier and most general unifier

Let $\mathbf{L} = \{L_1, \dots, L_k\}$ be a set of literals.

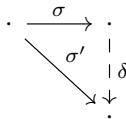
A substitution σ is a **unifier** of \mathbf{L} if

$$L_1\sigma = L_2\sigma = \dots = L_k\sigma$$

i.e. if $|\mathbf{L}\sigma| = 1$, where $\mathbf{L}\sigma = \{L_1\sigma, \dots, L_k\sigma\}$.

\mathbf{L} is **unifiable** if it has at least one unifier.

A unifier σ of \mathbf{L} is a **most general unifier (mgu)** of \mathbf{L} if for every unifier σ' of \mathbf{L} there is a substitution δ such that $\sigma' = \sigma\delta$.



Exercise

Unifiable?		Yes	No
$P(f(x))$	$P(g(y))$		
$P(x)$	$P(f(y))$		
$P(x)$	$P(f(x))$		
$P(x, f(y))$	$P(f(u), f(z))$		
$P(x, f(x))$	$P(f(y), y)$		
$P(x, g(x), g^2(x))$	$P(f(z), w, g(w))$		
$P(x, f(y))$	$P(g(y), f(a))$	$P(g(a), z)$	

Unification algorithm

Input: a set $\mathbf{L} \neq \emptyset$ of literals

$\sigma := []$ (the empty substitution)

while $|\mathbf{L}\sigma| > 1$ **do**

 Find the first position at which two literals $L_1, L_2 \in \mathbf{L}\sigma$ differ

if none of the two characters at that position is a variable

then return “non-unifiable”

else let x be the variable and t the term starting at that position

if x occurs in t

then return “non-unifiable”

else $\sigma := \sigma [t/x]$

return σ

Unification algorithm

Example

$\neg P(f(z, g(a, y)), h(z)),$

$\neg P(f(f(u, v), w), h(f(a, b)))$

Correctness of the unification algorithm

Lemma

The unification algorithm terminates.

Proof Every iteration of the **while**-loop (possibly except the last) replaces a variable x by a term t not containing x , and so the number of variables occurring in $\mathbf{L}\sigma$ decreases by one.

Lemma

If \mathbf{L} is non-unifiable then the algorithm returns “non-unifiable”.

Proof If \mathbf{L} is non-unifiable then the algorithm can never exit the loop normally.

Correctness/completeness of the unification algorithm

Lemma

If \mathbf{L} is unifiable then the algorithm returns the mgu of \mathbf{L} (and so in particular every unifiable set \mathbf{L} has an mgu).

Proof Assume \mathbf{L} is unifiable and let n be the number of iterations of the loop on input \mathbf{L} .

Let $\sigma_0 = []$, for $1 \leq i \leq n$ let σ_i be the value of σ after the i -th iteration of the loop.

We prove for every $0 \leq i \leq n$:

- (a) If $1 \leq i$, the i -th iteration does not return “non-unifiable”.
- (b) For every unifier σ' of \mathbf{L} there is a substitution δ_i such that $\sigma' = \sigma_i \delta_i$.

By (a) the algorithm exits the loop normally after n iterations.

By (b) it returns a most general unifier.

Correctness/completeness of the unification algorithm

Proof of (a) and (b) by induction on i :

Basis ($i = 0$): For (a) there is nothing to prove.

For (b) take $\delta_0 = \sigma'$.

Step ($i \Rightarrow i + 1$)

For (a), since $|\mathbf{L}\sigma_i| > 1$ and $\mathbf{L}\sigma_i$ unifiable, x and t exist and x does not occur in t , and so “non-unifiable” is not returned.

For (b): Let σ' be a unifier of \mathbf{L} . IH: $\sigma' = \sigma_i\delta_i$ for some δ_i .

δ_i must be of the form $[t_1/x_1, \dots, t_k/x_k, u/x]$ where x_1, \dots, x_k, x are distinct. Define $\delta_{i+1} = [t_1/x_1, \dots, t_k/x_k]$.

Note: $u = x\delta_i = t\delta_i = t\delta_{i+1}$ ($\sigma_i\delta_i$ is unifier (IH), x not in t)

$$\begin{aligned} & \sigma_{i+1} \delta_{i+1} \\ = & \sigma_i [t/x] \delta_{i+1} && \text{(algorithm extends } \sigma_i \text{ with } [t/x]) \\ = & \sigma_i [t_1/x_1, \dots, t_k/x_k, t\delta_{i+1}/x] \\ = & \sigma_i [t_1/x_1, \dots, t_k/x_k, u/x] && (u = t\delta_{i+1} \text{ by note)} \\ = & \sigma_i \delta_i \\ = & \sigma' && \text{(IH)} \end{aligned}$$

Resolvents for first-order logic

A substitution ρ is a **renaming** if for every variable x , $x\rho$ is a variable and ρ is injective on $dom(\rho)$.

A clause R is a **resolvent** of two clauses C_1 and C_2 iff:

- ▶ there is a renaming ρ such that
no variable occurs in both C_1 and $C_2\rho$ and
 ρ is injective on the set of variables in C_2 ;
- ▶ there are literals $L_1, \dots, L_m \in C_1$ ($m \geq 1$)
and $L'_1, \dots, L'_n \in C_2\rho$ ($n \geq 1$) such that
$$\mathbf{L} = \{\overline{L_1}, \dots, \overline{L_m}, L'_1, \dots, L'_n\}$$
is unifiable; and
- ▶ $R = ((C_1 - \{L_1, \dots, L_m\}) \cup (C_2\rho - \{L'_1, \dots, L'_n\})) \sigma$
for any mgu σ .

Example

$C_1 = \{ P(x), Q(x), P(g(y)) \}$ and $C_2 = \{ \neg P(x), R(f(x), a) \}$

Exercise

How many resolvents are there?

C_1	C_2	Resolvents
$\{P(x), Q(x, y)\}$	$\{\neg P(f(x))\}$	
$\{Q(g(x)), R(f(x))\}$	$\{\neg Q(f(x))\}$	
$\{P(x), P(f(x))\}$	$\{\neg P(y), Q(y, z)\}$	

Why renaming?

Example

$$\forall x(P(x) \wedge \neg P(f(x)))$$

Resolution for first-order logic

As for propositional logic, $F \vdash_{Res} C$ means that clause C can be derived from a set of clauses F by a sequence of resolution steps, i.e. that there is a sequence of clauses $C_1, \dots, C_m = C$ such that for every C_i

- ▶ either $C_i \in F$,
- ▶ or C_i is the resolvent of C_a and C_b where $a, b < i$.

Questions:

Correctness. Does $F \vdash_{Res} \square$ imply that F is unsatisfiable?

Completeness. Does unsatisfiability of F imply $F \vdash_{Res} \square$?

Exercise

Derive \square from the following clauses:

1. $\{\neg P(x), Q(x), R(x, f(x))\}$
2. $\{\neg P(x), Q(x), S(f(x))\}$
3. $\{T(a)\}$
4. $\{P(a)\}$
5. $\{\neg R(a, z), T(z)\}$
6. $\{\neg T(x), \neg Q(x)\}$
7. $\{\neg T(y), \neg S(y)\}$

Correctness of Resolution for First-Order Logic

Definition

The **universal closure** of a formula H with free variables x_1, \dots, x_n is the formula

$$\forall H = \forall x_1 \forall x_2 \dots \forall x_n H$$

Theorem

Let F be a closed formula in Skolem form with matrix F^* in CNF.
If $F^* \vdash_{Res} \square$ then F is unsatisfiable.

Theorem

Let F be a closed formula in Skolem form with matrix F^* in CNF.

If $F^* \vdash_{Res} \square$ then F is unsatisfiable.

Proof Let C_1, \dots, C_m be the sequence of clauses leading to \square .

We prove $\forall F^* \models \forall C_m$ by induction on m . Trivial if $C_m \in F^*$.

Let C_m be a resolvent of C_a and C_b ($a, b < m$). We prove

$$\forall C_a, \forall C_b \models \forall C_m \quad (*)$$

Thus $\forall F^* \models \forall C_m$ because $\forall F^* \models \forall C_a$ and $\forall F^* \models \forall C_b$ by IH.

Proof of (*): Assume $\mathcal{A}(\forall C_a) = \mathcal{A}(\forall C_b) = 1$. (**)

We prove $\mathcal{A}(\forall C_m) = 1$ by contradiction. Assume $\mathcal{A}(\forall C_m) = 0$.

$$\begin{aligned} \text{By def. } C_m &= ((C_a - \{L_1, \dots\}) \cup (C_b \rho - \{L'_1, \dots\}))\sigma \\ &= (C_a \sigma - \{L\}) \cup (C_b \rho \sigma - \{\bar{L}\}) \end{aligned}$$

$\Rightarrow \mathcal{A}'(C_m) = 0$ where $\mathcal{A}' = \mathcal{A}[u_1/x_1, \dots]$ for some $u_i \in U_{\mathcal{A}}$

$\Rightarrow \mathcal{A}'(C_a \sigma - \{L\}) = \mathcal{A}'(C_b \rho \sigma - \{\bar{L}\}) = 0$

$\Rightarrow \mathcal{A}'(L) = \mathcal{A}'(\bar{L}) = 1$ becs. $\mathcal{A}'(C_a \sigma) = \mathcal{A}'(C_b \rho \sigma) = 1$ becs. (**)

Contradiction.

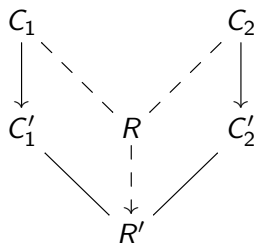
Completeness: The idea

Simulate ground resolution because that is complete

Lift the resolution proof from the ground resolution proof

Lifting Lemma

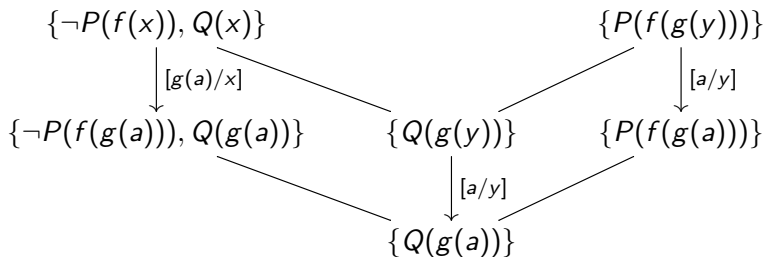
Let C_1, C_2 be two clauses and
let C'_1, C'_2 be two ground instances
with (propositional) resolvent R' .
Then there is a resolvent R of C_1, C_2
such that R' is a ground instance of R .



\rightarrow : Substitution

--- : Resolution

Lifting Lemma: example



Proof of Lifting Lemma.

(1) C'_1, C'_2 are ground instances of C_1, C_2 .

(2) R' is propositional resolvent of C'_1 and C'_2 .

We prove that R' is an instance of a resolvent of C_1 and C_2 .

Let ρ be a renaming s.t. C_1 and $C_2\rho$ have no common variables.

(1) $\Rightarrow C'_2$ is a ground instance of $C_2\rho$. \Rightarrow There are σ_1, σ_2 s.t.

$C'_1 = C_1\sigma_1$ and $C'_2 = C_2\rho\sigma_2$ and $dom(\sigma_1) \cap dom(\sigma_2) = \emptyset$.

$\Rightarrow C'_1 = C_1\sigma$ and $C'_2 = C_2\rho\sigma$ where $\sigma = \sigma_1 \cup \sigma_2$.

(2) $\Rightarrow R' = (C'_1 - \{L\}) \cup (C'_2 - \{\bar{L}\})$ where $L \in C'_1$ and $\bar{L} \in C'_2$.

\Rightarrow There are $\{L_1, \dots\} \subseteq C_1$ and $\{L'_1, \dots\} \subseteq C_2\rho$ s.t. σ is a unifier of $\{\bar{L}_1, \dots, L'_1, \dots\} =: M$.

Let σ_0 be an mgu of M . Then $\sigma = \sigma_0\delta$ for some δ .

$\Rightarrow R := ((C_1 - \{L_1, \dots\}) \cup (C_2\rho - \{L'_1, \dots\}))\sigma_0$ is resolvent.

$$\begin{aligned} R\delta &= ((C_1 - \{L_1, \dots\}) \cup (C_2\rho - \{L'_1, \dots\}))\sigma \\ &= (C_1\sigma - \{L\}) \cup (C_2\rho\sigma - \{\bar{L}\}) \\ &= (C'_1 - \{L\}) \cup (C'_2 - \{\bar{L}\}) \\ &= R' \end{aligned}$$

Completeness of Resolution for First-Order Logic

Theorem

Let F be a closed formula in Skolem form with matrix F^* in CNF.
If F is unsatisfiable then $F^* \vdash_{Res} \square$.

Proof If F is unsatisfiable, there is a ground resolution proof $C'_1, \dots, C'_n = \square$. We transform this step by step into a resolution proof $C_1, \dots, C_n = \square$ such that C'_i is a ground instance of C_i .

If C'_i is a ground instance of some clause $C \in F^*$:

Set $C_i = C$

If C'_i is a resolvent of C'_a, C'_b ($a, b < i$):

C'_a, C'_b have been transformed already into C_a, C_b s.t. C'_a, C'_b are ground instances of C_a, C_b . By the Lifting Lemma there is a resolvent R of C_a, C_b s.t. C'_i is a ground instance of R .

Set $C_i = R$.

Resolution Theorem for First-Order Logic

Theorem

Let F be a closed formula in Skolem form with matrix F^ in CNF.*

Then F is unsatisfiable iff $F^ \vdash_{Res} \square$.*

A resolution algorithm

Input: A closed formula F in Skolem form with matrix S in CNF,
i.e. S is a finite set of clauses

while $\square \notin S$ and
 there are clauses $C_a, C_b \in S$ and resolvent R of C_a and C_b
 such that $R \notin S$ (modulo renaming)
do $S := S \cup \{R\}$

The selection of resolvents must be *fair*, meaning that *every resolvent is eventually added*.

Three possible behaviours:

- ▶ The algorithm terminates and $\square \in S$
 $\Rightarrow F$ is unsatisfiable
- ▶ The algorithm terminates and $\square \notin S$
 $\Rightarrow F$ is satisfiable
- ▶ The algorithm does not terminate
 ($\Rightarrow F$ is satisfiable)

Refinements of resolution

Problems of resolution:

- ▶ Branching degree of the search space too large
- ▶ Too many dead ends
- ▶ Combinatorial explosion of the search space

Solution:

Strategies and **heuristics**: forbid certain resolution steps, which narrows the search space.

But: Completeness must be preserved!

First-Order Logic Equality

Predicate logic with equality

Predicate logic
+
distinguished predicate symbol “=” of arity 2

Semantics: A structure \mathcal{A} of predicate logic with equality always maps the predicate symbol = to the identity relation.

$$\mathcal{A}(=) = \{(d, d) \mid d \in U_{\mathcal{A}}\}$$

Expressivity

Fact

A structure is model of $\exists x \forall y x=y$ iff its universe is a singleton.

Theorem

Every satisfiable formula of predicate logic has a countably infinite model.

Proof Let F be satisfiable. Assume w.l.o.g. $F = \forall x_1 \dots \forall x_n F^*$ and the variables occurring in F^* are exactly x_1, \dots, x_n . (Bring F in closed Skolem form if needed.) Consider two cases:

$n = 0$. **Exercise.**

$n > 0$. Let $G = \forall x_1 \dots \forall x_n F^*[f(x_1)/x_1]$, where f does not occur in F^* . G is satisfiable (**why?**), and if G has a model M with universe U , then F has a model with universe $\{f^M(u) \mid u \in U\}$. By the fundamental theorem, G has a model with universe $T(G)$, which is countable infinite. So F also has a model with countably infinite universe.

Modeling equality

We assign to every formula F of predicate logic with equality a formula E_F of predicate logic.

Let Eq be a predicate symbol that does not occur in F .

E_F is the conjunction of the following formulas:

$$\forall x Eq(x, x)$$

$$\forall x \forall y (Eq(x, y) \rightarrow Eq(y, x))$$

$$\forall x \forall y \forall z ((Eq(x, y) \wedge Eq(y, z)) \rightarrow Eq(x, z))$$

For every function symbol f in F of arity n and every $1 \leq i \leq n$:

$$\forall x_1 \dots \forall x_n \forall y (Eq(x_i, y) \rightarrow \\ Eq(f(x_1, \dots, x_i, \dots, x_n), f(x_1, \dots, y, \dots, x_n)))$$

For every predicate symbol P in F of arity n and every $1 \leq i \leq n$:

$$\forall x_1 \dots \forall x_n \forall y (Eq(x_i, y) \rightarrow \\ (P(x_1, \dots, x_i, \dots, x_n) \leftrightarrow P(x_1, \dots, y, \dots, x_n)))$$

E_F expresses that Eq is a *congruence (relation)* on the symbols of the formula F .

Quotient structure

Definition

Let \mathcal{A} be a structure and let \sim be an equivalence relation on $U_{\mathcal{A}}$ that is a congruence for all the predicate and function symbols defined by $I_{\mathcal{A}}$. The **quotient structure** \mathcal{A}/\sim is defined as follows:

- ▶ $U_{\mathcal{A}/\sim} = \{[u]_{\sim} \mid u \in U_{\mathcal{A}}\}$ where $[u]_{\sim} = \{v \in U_{\mathcal{A}} \mid u \sim v\}$
- ▶ For every function symbol f defined by $I_{\mathcal{A}}$:
 $f^{\mathcal{A}/\sim}([d_1]_{\sim}, \dots, [d_n]_{\sim}) = [f^{\mathcal{A}}(d_1, \dots, d_n)]_{\sim}$
- ▶ For every predicate symbol P defined by $I_{\mathcal{A}}$:
 $P^{\mathcal{A}/\sim}([d_1]_{\sim}, \dots, [d_n]_{\sim}) = P^{\mathcal{A}}(d_1, \dots, d_n)$
- ▶ For every variable x defined by $I_{\mathcal{A}}$: $x^{\mathcal{A}/\sim} = [x^{\mathcal{A}}]_{\sim}$

Lemma

$\mathcal{A}/\sim(t) = [\mathcal{A}(t)]_{\sim}$ and $\mathcal{A}/\sim(F) = \mathcal{A}(F)$

Theorem

The formulas F and $E_F \wedge F[Eq/=]$ are equisatisfiable.

Proof

(\Leftarrow): If $E_F \wedge F[Eq/=]$ is satisfiable, then F is satisfiable.

Assume $\mathcal{A} \models E_F \wedge F[Eq/=]$. Then $Eq^{\mathcal{A}}$ is a congruence.

Define $\mathcal{B} = \mathcal{A}/_{Eq^{\mathcal{A}}}$ (extended with $=$ interpreted as identity).

We prove $\mathcal{B}(F) = 1$.

Claim 1: $\mathcal{B} \models F[Eq/=]$. Follows from the Lemma.

Claim 2: $Eq^{\mathcal{B}}$ is the identity.

$$\begin{aligned} & Eq^{\mathcal{B}}([a]_{Eq^{\mathcal{A}}}, [a']_{Eq^{\mathcal{A}}}) \\ = & Eq^{\mathcal{A}}(a, a') && \text{(Def. of quotient structure)} \\ = & ([a]_{Eq^{\mathcal{A}}} = [a']_{Eq^{\mathcal{A}}}) && \text{(Def. of equivalence class)} \end{aligned}$$

We have: $\mathcal{B}(F) \stackrel{(2)}{=} \mathcal{B}(F[Eq/=]) \stackrel{(1)}{=} 1$.

(\Leftarrow): If F is satisfiable., then $E_F \wedge F[Eq/=]$ is satisfiable.

Any model of F yields a model of $E_F \wedge F[Eq/=]$ by interpreting Eq as equality.

First-Order Logic Undecidability

[Cutland, *Computability*, Section 6.5.]

- ▶ Aim:
Show that validity of first-order formulas is undecidable
- ▶ Method:
Reduce the halting problem for register machines to validity of formulas by expressing “program behaviour” as formulas

Logical formulas can talk about computations!

Register machine programs (RMPs)

A register machine program is a sequence of instructions I_1, \dots, I_t . The instructions manipulate registers R_i ($i = 1, 2, \dots, r$) that contain (unbounded!) natural numbers.

There are 4 types of instructions:

$$R_i := 0$$

$$R_i := R_i + 1$$

$$R_i := R_j$$

$$\text{IF } R_i = R_j \text{ GOTO } p$$

Assumption: all jumps in a program go to $1, \dots, t + 1$, and execution terminates when the PC (the number of the next instruction to be executed) is $t + 1$.

The state of P during execution can be described by a tuple of $r + 1$ natural numbers

$$(n_1, \dots, n_r, k)$$

where n_i is the content of R_i and k is the value of the PC.

Undecidability

Theorem (Undecidability of the halting problem for RMPs)

It is undecidable if a given register machine program terminates when started in state $(0, \dots, 0, 1)$.

We reduce the halting problem for RMPs to the validity problem for first-order formulas.

Notation:

$P(0) \downarrow =$ “RMP P started in state $(0, \dots, 0, 1)$ terminates”

Theorem

Given an RMP P we can effectively construct a closed formula φ_P such that $P(0) \downarrow$ iff $\models \varphi_P$.

Proof by construction of φ_P from $P = I_1, \dots, I_t$.

Funct. symb.: z, s . Abbr.: $\bar{0} = z, \bar{1} = s(z), \bar{2} = s(s(z)), \dots$

Pred. symb.: R (arity: $r + 1$). (Think “reachable”).

Aim: if $R(\bar{n}_1, \dots, \bar{n}_r, \bar{k})$ then $(0, \dots, 0, 1) \xrightarrow{P} (n_1, \dots, n_r, k)$.

1) For every I_i construct closed formula Ψ_i :

$I_i = (R_n := 0)$: $\Psi_i := \forall x_1 \dots x_r (R(x_1, \dots, x_n, \dots, x_r, \bar{i}) \rightarrow R(x_1, \dots, z, \dots, x_r, \overline{i+1}))$

$I_i = (R_n := R_n + 1)$: same except $s(x_n)$ instead of z

$I_i = (R_n := R_m)$: same except x_m instead of z

$I_i = (IF R_m = R_n GOTO p)$:

$\Psi_i := \forall x_1 \dots x_r (R(x_1, \dots, x_r, \bar{i}) \rightarrow (x_m = x_n \rightarrow R(x_1, \dots, x_r, \bar{p})) \wedge (x_m \neq x_n \rightarrow R(x_1, \dots, x_r, \overline{i+1})))$

2) Define $\Psi_P := \Psi \wedge R(z, \dots, z, s(z)) \wedge \Psi_1 \wedge \dots \wedge \Psi_t$ where

$\Psi := \forall x \forall y (s(x) = s(y) \rightarrow x = y) \wedge \forall x (z \neq s(x))$.

Ψ enforces that every model is “similar enough” to \mathbb{N} .

3) Define $\varphi_P := \Psi_P \rightarrow \tau$ where $\tau := \exists x_1 \dots x_r R(x_1, \dots, x_r, s(\bar{t}))$.

Claim: $P(0) \downarrow$ iff $\models \varphi_P$, that is, $P(0) \downarrow$ iff $\models \Psi_P \rightarrow \tau$.

“ \Rightarrow ”: Assume $P(0) \downarrow$. We show: for every \mathcal{A} , if $\mathcal{A} \models \Psi_P$ then $\mathcal{A} \models \tau$. Assume $\mathcal{A} \models \Psi_P$.

Lemma

If $(0, \dots, 0, 1) \overset{P}{\rightsquigarrow} (n_1, \dots, n_r, k)$ then $\mathcal{A} \models R(\bar{n}_1, \dots, \bar{n}_r, \bar{k})$

Proof by induction on the length of the execution using $\mathcal{A} \models \Psi_P$.

Thus $\mathcal{A} \models \tau$ because $P(0) \downarrow$.

“ \Leftarrow ”: Assume $\models \Psi_P \rightarrow \tau$. We show $P(0) \downarrow$.

We have $\mathcal{N} \models \Psi_P \rightarrow \tau$ for the structure \mathcal{N} given by

$$U_{\mathcal{N}} := \mathbb{N} \quad z^{\mathcal{N}} := 0 \quad s^{\mathcal{N}}(n) := n + 1 .$$

In this structure $R^{\mathcal{N}} := \{s \mid (0, \dots, 0, 1) \overset{P}{\rightsquigarrow} s\}$ and so $\mathcal{N} \models \Psi_P$.

From $\mathcal{N} \models \Psi_P$ and $\mathcal{N} \models \Psi_P \rightarrow \tau$ we get $\mathcal{N} \models \tau$, which implies $P(0) \downarrow$.

First-Order Logic Compactness

[Harrison, Section 3.16]

More Herbrand Theory

Recall Gödel-Herbrand-Skolem:

Theorem

Let F be a closed formula in Skolem form. Then F is satisfiable iff its Herbrand expansion $E(F)$ is (propositionally) satisfiable.

$T(S)$: the set of all terms without variables constructed out of function symbols of S (plus a constant, if S contains none).

$E(S)$: set of all propositional formulas constructed by replacing the variables in the matrices of the formulas in S with terms from $T(S)$.

We have:

Theorem (1)

Let S be a set of closed formulas in Skolem form.

Then S is satisfiable iff $E(S)$ is (propositionally) satisfiable.

Proof: Show first that S is satisfiable iff it has a Herbrand model, and then that it is equivalent to the Herbrand expansion.

Transforming sets of formulas

Recall the transformation of single formulas into equisatisfiable Skolem form: close, RPF, skolemize

Theorem (2)

Let S be a countable set of closed formulas. Then we can transform it into an equisatisfiable set T of closed formulas in Skolem form. We call this transformation function skolem.

- ▶ Can all formulas in S be transformed in parallel?
- ▶ Why countable?

Transforming sets of formulas

Proof:

1. Put all formulas in S into RPF.

Problem in Skolemization step: How do we generate new function symbols if all of them have been used already in S ?

2. Rename all function symbols in S : $f_i^k \mapsto f_{2i}^k$

The result: equisatisfiable countable set $\{F_0, F_1, \dots\}$.

Unused symbols: all f_{2i+1}^k

3. Skolemize the F_i one by one using the f_{2i+1}^k not used in the Skolemization of F_0, \dots, F_{i-1}

Result is equisatisfiable with initial S .

Compactness

Theorem

Let S be a countable set of closed formulas.

If every finite subset of S is satisfiable, then S is satisfiable.

Proof every fin. $F \subseteq S$ is sat.

\Rightarrow every fin. $F \subseteq skolem(S)$ is sat. by Theorem (2)

(fin. $F \subseteq skolem(S) \Rightarrow F \subseteq skolem(S_0)$ for some fin. $S_0 \subseteq S$)

\Rightarrow for every fin. $F \subseteq skolem(S)$, $E(F)$ is prop. sat. by Theorem(1)

\Rightarrow every fin. $F' \subseteq E(skolem(S))$ is prop. sat.

(there must exist a fin. $F \subseteq skolem(S)$ s.t. $F' \subseteq E(F)$)

$\Rightarrow E(skolem(S))$ is prop. sat. by prop. compactness

$\Rightarrow skolem(S)$ is sat. by Theorem (1)

$\Rightarrow S$ is sat. by Theorem (2)

First-Order Logic

The Classical Decision Problem

Validity/satisfiability of arbitrary first-order formulas is undecidable.

What about subclasses of formulas?

Examples

$\forall x \exists y (P(x) \rightarrow P(y))$ Satisfiable? Resolution?

$\exists x \forall y (P(x) \rightarrow P(y))$ Satisfiable? Resolution?

The $\exists^*\forall^*$ class

Definition

The $\exists^*\forall^*$ class is the class of closed formulas of the form

$$\exists x_1 \dots \exists x_m \forall y_1 \dots \forall y_n F$$

where F is a quantifier-free formula that contains no function symbols of arity > 0 .

This is also called the **Bernays-Schönfinkel class**.

Corollary

(Un)satisfiability is decidable for formulas in the \exists^\forall^* class.*

Proof The Herbrand universe of $\exists^*\forall^*$ -formulas is finite.

What if a formula is not in the $\exists^*\forall^*$ class?

Try to transform it into the $\exists^*\forall^*$ class!

Example

$$\forall y \exists x (P(x) \rightarrow Q(y)) \equiv \exists x \forall y (P(x) \rightarrow Q(y))$$

Heuristic transformation procedure (may or may not work):

1. Put formula into NNF.
2. Push all quantifiers into the formula as far as possible (“miniscoping”).
3. Pull out \exists first and \forall afterwards.

Miniscoping

Perform the following transformations bottom-up, as long as possible:

- ▶ $(\exists x F) \equiv F$ if x does not occur free in F
- ▶ $\exists x (F \vee G) \equiv (\exists x F) \vee (\exists x G)$
- ▶ $\exists x (F \wedge G) \equiv (\exists x F) \wedge G$ if x is not free in G
- ▶ $\exists x F$ where F is a conjunction, x occurs free in every conjunct, and the DNF of F is of the form $F_1 \vee \dots \vee F_n$, $n \geq 2$:
 $\exists x F \equiv \exists x (F_1 \vee \dots \vee F_n)$.
- ▶ dual transformations for \forall of all of the above.

Warning: Complexity!

Miniscoping

Example

$$\begin{aligned} & \exists x (P(x) \wedge \exists y (Q(y) \vee R(x))) \\ \equiv & \exists x (P(x) \wedge (\exists y Q(y) \vee \exists y R(x))) \\ \equiv & \exists x (P(x) \wedge (\exists y Q(y) \vee R(x))) \\ \equiv & \exists x ((P(x) \wedge \exists y Q(y)) \vee (P(x) \wedge R(x))) \\ \equiv & \exists x (P(x) \wedge \exists y Q(y)) \vee \exists x (P(x) \wedge R(x)) \\ \equiv & (\exists x P(x) \wedge \exists y Q(y)) \vee \exists x (P(x) \wedge R(x)) \end{aligned}$$

The monadic class

Definition

A formula is **monadic** if it contains only unary (monadic) predicate symbols and no function symbol of arity > 0 .

Examples

All men are mortal. Socrates is a man. Socrates is mortal.

The monadic class is decidable

Theorem

For every monadic formula, the heuristic transformation procedure yields an equisatisfiable \exists^\forall^* -formula.*

Proof Put into NNF and perform miniscoping.

The result has no nested quantifiers (**Exercise!**).

First pull out all \exists , then all \forall , and existentially quantify free variables.

The result is in the $\exists^*\forall^*$ class.

Corollary

(Un)satisfiability of monadic formulas is decidable.

The finite model property

Definition

A formula F has the **finite model property** (for satisfiability) if F has a model iff F has a finite model.

Theorem

If a class of formulas has the finite model property, satisfiability is decidable.

Proof. Two semi-decision procedures, one for unsatisfiability and one for satisfiability. The procedure for satisfiability searches systematically for a model through all structures with finite domain.

The finite model property

Another proof of decidability of satisfiability for monadic formulas:

Theorem

Monadic formulas have the finite model property.

Proof

We show: A satisfiable monadic formula F with k different monadic predicate symbols P_1, \dots, P_k has a model of size $\leq 2^k$.

Given a model \mathcal{A} of F and $u, v \in U^{\mathcal{A}/\sim}$, define $u \sim v$ iff $P_i^{\mathcal{A}}(u) = P_i^{\mathcal{A}}(v)$ for every $1 \leq i \leq k$.

\sim is a congruence (immediate consequence of the definition of congruence and the fact that all predicates are monadic).

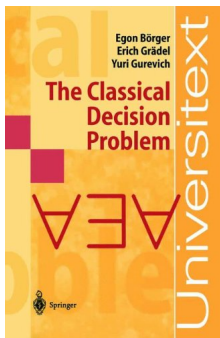
\mathcal{A}_{\sim} (the quotient of \mathcal{A} w.r.t. \sim) is also a model of F .

$|U_{\mathcal{A}/\sim}| \leq 2^k$, because an equivalence class $[u]_{\sim}$ is characterized by the bit-vector $(P_1^{\mathcal{A}}(u), \dots, P_k^{\mathcal{A}}(u))$ of length k .

Classification by quantifier prefix of prenex form

There is a **complete** classification of decidable and undecidable classes of formulas based on

- ▶ the form of the quantifier prefix of the prenex form
- ▶ the arity of the predicate and function symbols allowed
- ▶ whether “=” is allowed or not.



A complete classification

Only formulas without function symbols of arity > 0 ,
no restrictions on predicate symbols.

Satisfiability is decidable:

$\exists^* \forall^*$ (Bernays, Schönfinkel 1928, Ramsey 1930)

$\exists^* \forall \exists^*$ (Ackermann 1928)

$\exists^* \forall^2 \exists^*$ (Gödel 1932)

Satisfiability is undecidable:

$\forall^3 \exists$ (Surányi 1959)

$\forall \exists \forall$ (Kahr, Moore, Wang 1962)

Why complete?

Famous mistake by Gödel: $\exists^* \forall^2 \exists^*$ with “=” is **undecidable**
(Goldfarb 1984)

First-Order Logic

Basic Proof Theory

Gebundene Namen sind Schall und Rauch

We permit ourselves to identify formulas that differ only in the names of bound variables.

Example

$$\forall x \exists y P(x, y) = \forall u \exists v P(u, v)$$

Recall: renaming must not capture free variables!

$$\forall x P(x, y) \neq \forall y P(y, y)$$

In the following: Substitution $F[t/x]$ assumes that bound variables in F are automatically renamed to avoid capturing free variables.

Example

$$(\exists x P(x, y))[x/y] = \exists x' P(x', x)$$

All proof systems below are extensions
of the corresponding propositional systems

Sequent Calculus

Recall: Sequent Calculus rules

$$\frac{}{\perp, \Gamma \Rightarrow \Delta} \perp L$$

$$\frac{\Gamma \Rightarrow F, \Delta}{\neg F, \Gamma \Rightarrow \Delta} \neg L$$

$$\frac{F, G, \Gamma \Rightarrow \Delta}{F \wedge G, \Gamma \Rightarrow \Delta} \wedge L$$

$$\frac{F, \Gamma \Rightarrow \Delta \quad G, \Gamma \Rightarrow \Delta}{F \vee G, \Gamma \Rightarrow \Delta} \vee L$$

$$\frac{\Gamma \Rightarrow F, \Delta \quad G, \Gamma \Rightarrow \Delta}{F \rightarrow G, \Gamma \Rightarrow \Delta} \rightarrow L$$

$$\frac{}{A, \Gamma \Rightarrow A, \Delta} Ax$$

$$\frac{F, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \neg F, \Delta} \neg R$$

$$\frac{\Gamma \Rightarrow F, \Delta \quad \Gamma \Rightarrow G, \Delta}{\Gamma \Rightarrow F \wedge G, \Delta} \wedge R$$

$$\frac{\Gamma \Rightarrow F, G, \Delta}{\Gamma \Rightarrow F \vee G, \Delta} \vee R$$

$$\frac{F, \Gamma \Rightarrow G, \Delta}{\Gamma \Rightarrow F \rightarrow G, \Delta} \rightarrow R$$

Rules for quantifiers

We add the following rules:

$$\frac{F[t/x], \forall x F, \Gamma \Rightarrow \Delta}{\forall x F, \Gamma \Rightarrow \Delta} \forall L$$

$$\frac{\Gamma \Rightarrow F[y/x], \Delta}{\Gamma \Rightarrow \forall x F, \Delta} \forall R (*)$$

$$\frac{F[y/x], \Gamma \Rightarrow \Delta}{\exists x F, \Gamma \Rightarrow \Delta} \exists L (*)$$

$$\frac{\Gamma \Rightarrow F[t/x], \exists x F, \Delta}{\Gamma \Rightarrow \exists x F, \Delta} \exists R$$

(*): y not free in the conclusion of the rule

Note: $\forall L$ and $\exists R$ do not delete the principal formula, and so termination no longer guaranteed.

Soundness

Lemma

For every quantifier rule $\frac{S'}{S}$, $|S|$ and $|S'|$ are equivalent.

Theorem (Soundness)

If $\vdash_G S$ then $\models |S|$.

Proof induction on the size of the proof of $\vdash_G S$ using the above lemma and the corresponding propositional lemma (stating

$|S| \equiv |S_1| \wedge \dots \wedge |S_n|$ for every rule $\frac{S_1 \quad \dots \quad S_n}{S}$).

Completeness Proof

Construct counter model
from (possibly infinite!) failed proof search.

Let e_0, e_1, \dots be an enumeration of all terms
(over some given set of function symbols and variables)

Proof search

Construct proof tree incrementally:

1. Pick some unproved leaf $\Gamma \Rightarrow \Delta$ such that some rule is applicable.
2. Pick some principal formula in $\Gamma \Rightarrow \Delta$ fairly and apply rule.

$\forall R, \exists L$: pick some arbitrary new y .

$\forall L, \exists R$:

$$t = \begin{cases} e_0 & \text{if the p.f. has never been instantiated} \\ & \text{(on the path to the root)} \\ e_{i+1} & \text{if the previous instantiation of the p.f.} \\ & \text{(on the path to the root) used } e_i \end{cases}$$

Failed proof search: there is a branch A such that either A ends in a sequent where no rule is applicable or A is infinite.

Construction of Herbrand countermodel \mathcal{A} from A

Define a structure \mathcal{A} by:

$U^{\mathcal{A}}$ = all terms over the function symbols and variables in A .

$f^{\mathcal{A}}(t_1, \dots, t_n) = f(t_1, \dots, t_n)$.

$P^{\mathcal{A}} = \{(t_1, \dots, t_n) \mid P(t_1, \dots, t_n) \in \Gamma \text{ for some } \Gamma \Rightarrow \Delta \in A\}$.

Theorem

For all $\Gamma \Rightarrow \Delta \in A$, for all $F \in \Gamma \cup \Delta$: $\mathcal{A}(F) = \begin{cases} 1 & \text{if } F \in \Gamma \\ 0 & \text{if } F \in \Delta \end{cases}$

In particular, $\Gamma_1 \cap \Delta_2$ for any two sequents $\Gamma_1 \Rightarrow \Delta_1$ and $\Gamma_2 \Rightarrow \Delta_2$ of A .

Proof by induction on the structure of F .

Basis: $F = P(t_1, \dots, t_n)$.

Assume $F \in \Gamma$. Then $\mathcal{A}(F) = 1$ by definition.

Assume $F \in \Delta$. Then F does not belong to any Γ of A ; otherwise A would end with an application of Ax . So $\mathcal{A}(F) = 0$.

$\frac{}{F, \Gamma \Rightarrow F, \Delta} Ax$
where F atomic.

Step: F is not atomic . Then F is the principal formula of some sequent $\Gamma \Rightarrow \Delta \in A$ (fairness!).

We consider several cases, depending on the form of F and whether $F \in \Gamma$ or $F \in \Delta$:

$F = \neg G$:

Take any step $\frac{\tilde{\Gamma} \Rightarrow \tilde{\Delta}}{\Gamma \Rightarrow \Delta}$ of A .

If $\neg G \in \Gamma$ then $G \in \tilde{\Delta}$.

By IH $\mathcal{A}(G) = 0$ and so $\mathcal{A}(\neg G) = 1$.

If $\neg G \in \Delta$ then $G \in \tilde{\Gamma}$.

By IH $\mathcal{A}(G) = 1$ and so $\mathcal{A}(\neg G) = 0$.

$F = G_1 \wedge G_2$:

Take any $\frac{\tilde{\Gamma} \Rightarrow \tilde{\Delta}}{\Gamma \Rightarrow \Delta}$ of A .

If $G_1 \wedge G_2 \in \Gamma$ then $G_1 \in \tilde{\Gamma}$ and $G_2 \in \tilde{\Gamma}$.

By IH $\mathcal{A}(G_1) = \mathcal{A}(G_2) = 1$, and so $\mathcal{A}(G_1 \wedge G_2) = 1$

$$\frac{\Gamma' \Rightarrow G, \Delta}{\neg G, \Gamma' \Rightarrow \Delta} \neg L$$

$$\frac{G, \Gamma \Rightarrow \Delta'}{\Gamma \Rightarrow \neg G, \Delta'} \neg R$$

$$\frac{G_1, G_2, \Gamma \Rightarrow \Delta}{G_1 \wedge G_2, \Gamma' \Rightarrow \Delta} \wedge L$$

$$\frac{\Gamma \Rightarrow G_1, \Delta \quad \Gamma \Rightarrow G_2, \Delta}{\Gamma \Rightarrow G_1 \wedge G_2, \Delta} \wedge R$$

Completeness

Corollary

If proof search with root $\Gamma \Rightarrow \Delta$ fails, then there is a structure \mathcal{A} such that $\mathcal{A}(\bigwedge \Gamma \rightarrow \bigvee \Delta) = 0$.

Example

$$\exists x P(x) \Rightarrow \forall x P(x)$$

Corollary (Completeness)

If $\models |\Gamma \rightarrow \Delta|$ then $\vdash_G \Gamma \Rightarrow \Delta$

Proof by contradiction. If not $\vdash_G \Gamma \Rightarrow \Delta$ then proof search fails. Then there is an \mathcal{A} such that $\mathcal{A}(\bigwedge \Gamma \rightarrow \bigvee \Delta) = 0$.

Therefore not $\models |\Gamma \rightarrow \Delta|$.

Natural Deduction

Recall: Natural Deduction rules

$$\frac{F \quad G}{F \wedge G} \wedge I$$

$$\frac{F \wedge G}{F} \wedge E_1 \quad \frac{F \wedge G}{G} \wedge E_2$$

$$\frac{\begin{array}{c} [F] \\ \vdots \\ G \end{array}}{F \rightarrow G} \rightarrow I$$

$$\frac{F \rightarrow G \quad F}{G} \rightarrow E$$

$$\frac{F}{F \vee G} \vee I_1 \quad \frac{G}{F \vee G} \vee I_2$$

$$\frac{F \vee G \quad \begin{array}{c} [F] \\ \vdots \\ H \end{array} \quad \begin{array}{c} [G] \\ \vdots \\ H \end{array}}{H} \vee E$$

$$\frac{\begin{array}{c} [F] \\ \vdots \\ \perp \end{array}}{\neg F} \neg I$$

$$\frac{\neg F \quad F}{\perp} \neg E$$

$$\frac{\begin{array}{c} [\neg F] \\ \vdots \\ \perp \end{array}}{F} \perp$$

Rules for quantifiers

$$\frac{F[y/x]}{\forall x F} \quad \forall I(*)$$

$$\frac{\forall x F}{F[t/x]} \quad \forall E$$

$$\frac{F[t/x]}{\exists x F} \quad \exists I$$

$$\frac{\exists x F \quad \begin{array}{c} [F[y/x]] \\ \vdots \\ H \end{array}}{H} \quad \exists E(**)$$

(*): ($y = x$ or $y \notin fv(F)$) and
 y not free in an open assumption in the proof of $F[y/x]$

(**): ($y = x$ or $y \notin fv(F)$) and
 y not free in H or in an open assumption in the proof of the
second premise, except for $F[y/x]$

Example of a proof

$\forall x (\exists y P(y) \rightarrow Q(x)) \vdash_N \forall x \exists y (P(y) \rightarrow Q(x)):$

$$\frac{\frac{\frac{[P(z)]^3}{\exists y P(y)} \exists I:5 \quad \frac{\forall x (\exists y P(y) \rightarrow Q(x))}{\exists y P(y) \rightarrow Q(z)} \forall E:6}{Q(z)} \rightarrow E:4}{\frac{Q(z)}{P(z) \rightarrow Q(z)} \rightarrow I:3} \exists I:2} \forall I:1 \frac{\exists y (P(y) \rightarrow Q(z))}{\forall x \exists y (P(y) \rightarrow Q(x))}$$

Soundness

Theorem (Soundness)

If $\Gamma \vdash_N F$ then $\Gamma \models F$

Proof by induction of the depth of the proof tree for $\Gamma \vdash_N F$, with additional cases. We only consider one:

Case: rule applied to the root is

$$\frac{\frac{\exists x F}{H} \quad \begin{array}{c} [F[y/x]] \\ \vdots \\ H \end{array}}{H} \exists E(**)$$

(**) y not free in H or in an open assumption in the proof of the second premise, except for $F[y/x]$.

To show: $\Gamma \models H$, i.e., for every \mathcal{A} , if $\mathcal{A} \models \Gamma$ then $\mathcal{A} \models H$.

IH: $\Gamma \models \exists x F$ and $F[y/x], \Gamma \models H$.

Soundness

To show: $\Gamma \models H$, i.e., for every \mathcal{A} , if $\mathcal{A} \models \Gamma$ then $\mathcal{A} \models H$.

IH: $\Gamma \models \exists x F$ and $F[y/x], \Gamma \models H$.

Pick \mathcal{A} arbitrary.

$$\begin{array}{ll} \mathcal{A} \models \Gamma & \\ \Rightarrow \mathcal{A} \models \exists x F & (\Gamma \models \exists x F \text{ by IH}) \\ \Rightarrow \mathcal{A}[u/x] \models F \text{ for some } u \in U^{\mathcal{A}} & (\text{semantics}) \\ \Rightarrow \mathcal{A}[u/y] \models F[y/x] & (y = x \text{ or } y \notin \text{fv}(F)) \\ \quad \text{and } \mathcal{A}[u/y] \models \Gamma & (y \text{ not free in } \Gamma) \\ \Rightarrow \mathcal{A}[u/y] \models H & (\text{transit. of } \models) \\ \Rightarrow \mathcal{A} \models H & (y \text{ not free in } H) \end{array}$$

Completeness

Theorem (ND can simulate SC)

If $\vdash_G \Gamma \Rightarrow \Delta$ then $\Gamma, \neg\Delta \vdash_N \perp$ (where $\neg\{F_1, \dots\} = \{\neg F_1, \dots\}$).

Proof by induction on (the depth of) $\vdash_G \Gamma \Rightarrow \Delta$. (Omitted.)

Corollary (Completeness of ND)

If $\Gamma \models F$ then $\Gamma \vdash_N F$.

Proof as before: apply the completeness of \vdash_G . (Omitted.)

Hilbert System

Recall: Hilbert System

Axioms:

$$F \rightarrow G \rightarrow F \quad (\text{A1})$$

$$(F \rightarrow G \rightarrow H) \rightarrow (F \rightarrow G) \rightarrow F \rightarrow H \quad (\text{A2})$$

$$F \rightarrow G \rightarrow F \wedge G \quad (\text{A3})$$

$$F \wedge G \rightarrow F \quad (\text{A4})$$

$$F \wedge G \rightarrow G \quad (\text{A5})$$

$$F \rightarrow F \vee G \quad (\text{A6})$$

$$G \rightarrow F \vee G \quad (\text{A7})$$

$$F \vee G \rightarrow (F \rightarrow H) \rightarrow (G \rightarrow H) \rightarrow H \quad (\text{A8})$$

$$(\neg F \rightarrow \perp) \rightarrow F \quad (\text{A9})$$

Inference rule: modus ponens

$$\frac{F \rightarrow G \quad F}{G} \rightarrow E$$

New axioms and inference rule

Additional axioms:

$$\forall x F \rightarrow F[t/x]$$

$$F[t/x] \rightarrow \exists x F$$

$$\forall x (G \rightarrow F) \rightarrow (G \rightarrow \forall y F[y/x]) \quad (*)$$

$$\forall x (F \rightarrow G) \rightarrow (\exists y F[y/x] \rightarrow G) \quad (*)$$

(*) where $x \notin fv(G)$ and ($y = x$ or $y \notin fv(F)$)

Additional inference rule:

$$\frac{F}{\forall y F[y/x]} \quad \forall I(*)$$

(*) provided x not free in the assumptions
and ($y = x$ or $y \notin fv(F)$).

Equivalence of Hilbert and ND

As before, with additional cases.

First-order Predicate Logic Theories

Preliminary Definitions

Definition

A **signature** Σ is a set of predicate and function symbols.

A **Σ -formula** is a formula that contains only predicate and function symbols from Σ .

A **Σ -structure** is a structure that interprets all predicate and function symbols from Σ .

Definition

A **Σ -sentence** is a closed Σ -formula.

Convention: we assume that a signature Σ has been fixed, and drop Σ in Σ -formula, Σ -structure, or Σ -sentence. That is, we silently assume that all formulas, structures, and sentences are over the same fixed signature Σ .

Theories

Definition

A **theory** is a (finite or infinite) set of sentences S closed under consequence: If $S \models F$ and F is a sentence, then $F \in S$.

Fact

The set A of all sentences is a theory: If $S \models F$ for a sentence F , then in particular F is a sentence and so $F \in S$.

*The set V of all **valid** sentences is a theory: If $V \models F$ for a sentence F , then F is valid and so $F \in V$.*

$V \subseteq S$ holds for every theory S : If F is a valid sentence, then $\models F$. It follows $S \models F$ and, since S is a theory, $F \in S$.

There are two ways to define interesting theories:

- ▶ As the set of sentences satisfied by a fixed structure.
- ▶ As the set of consequences of a fixed set of sentences.

Theories from structures

Definition

Given a structure \mathcal{A} , let $Th(\mathcal{A})$ denote the set of all sentences true in \mathcal{A} . That is, $Th(\mathcal{A}) := \{F \mid F \text{ is a sentence and } \mathcal{A} \models F\}$.

Lemma

For every structure \mathcal{A} and sentence F : $\mathcal{A} \models F$ iff $Th(\mathcal{A}) \models F$.

Proof.

“ \Rightarrow ”: $\mathcal{A} \models F \Rightarrow F \in Th(\mathcal{A}) \Rightarrow Th(\mathcal{A}) \models F$.

“ \Leftarrow ”: Assume $Th(\mathcal{A}) \models F$. We prove $\mathcal{A} \models Th(\mathcal{A})$, which, together with $\mathcal{A} \models F$, implies $\mathcal{A} \models F$. To prove $\mathcal{A} \models Th(\mathcal{A})$, let $G \in Th(\mathcal{A})$. We have $\mathcal{A} \models G$ by definition of $Th(\mathcal{A})$.

Corollary

$Th(\mathcal{A})$ is a theory.

Proof. Assume $Th(\mathcal{A}) \models F$. By the lemma above, $\mathcal{A} \models F$. From the definition of $Th(\mathcal{A})$ we get $F \in Th(\mathcal{A})$.

Example

Notation: $(\mathbb{Z}, +, \leq)$ denotes the structure with universe \mathbb{Z} and the standard interpretations for the symbols $+$ and \leq .

The same notation is used for other standard structures where the interpretation of a symbol is clear from the symbol.

Example (Linear integer arithmetic)

$Th(\mathbb{Z}, +, \leq)$ is the set of all sentences over the signature $\{+, \leq\}$ that are true in the structure $(\mathbb{Z}, +, \leq)$.

Famous numerical theories

$Th(\mathbb{R}, +, <, =)$ is called **linear real arithmetic**.

It is decidable.

$Th(\mathbb{R}, +, *, <, =)$ is called **real arithmetic**.

It is decidable.

$Th(\mathbb{Z}, +, <, =)$ is called **linear integer arithmetic** or **Presburger arithmetic**.

It is decidable.

$Th(\mathbb{Z}, +, *, <, =)$ is called **integer arithmetic**.

It is not even **semidecidable** (= r.e.).

Decidability via special algorithms.

Theories from axioms

Definition

Let S be a set of sentences. $Cn(S)$ denotes the set of **consequences** of S : $Cn(S) = \{F \mid F \text{ is a sentence and } S \models F\}$

Examples

$Cn(\emptyset)$ is the set of valid sentences.

$Cn(\{\forall x \forall y \forall z (x * y) * z = x * (y * z)\})$ is the set of sentences that are true in all semigroups.

Lemma

For every set S of sentences, the set $Cn(S)$ is a theory.

Proof. We show: if F is a sentence and $Cn(S) \models F$ then $F \in Cn(S)$.

By the def. of $Cn(S)$ we have $S \models Cn(S)$. By assumption $Cn(S) \models F$, and so $S \models F$ by transitivity of \models . From the definition of $Cn(S)$ we get $F \in Cn(S)$.

Axioms

Definition

Let S be a set of sentences.

A theory T is **axiomatized** by S if $T = Cn(S)$.

A theory T is **axiomatizable** if there is some decidable or recursively enumerable S that axiomatizes T .

A theory T is **finitely axiomatizable** if there is some finite S that axiomatizes T .

Completeness and elementary equivalence

Definition

A theory T is **complete** if for every sentence F , $T \models F$ or $T \models \neg F$.

Fact

$Th(\mathcal{A})$ is complete for every structure \mathcal{A} .

$Cn(\{\forall x \forall y \forall z (x * y) * z = x * (y * z)\})$ is incomplete: neither $\forall x \forall y x * y = y * x$ nor its negation belong to it.

Definition

Two structures \mathcal{A} and \mathcal{B} are **elementarily equivalent** if $Th(\mathcal{A}) = Th(\mathcal{B})$.

Theorem

A theory T is complete iff all its models are elementarily equivalent.

Theorem

A theory T is complete iff all its models are elementarily equivalent.

Proof. We prove that T is incomplete iff two of its models are not elementarily equivalent.

“ \Rightarrow :” Assume T is incomplete. Then $T \not\models \neg F$ and $T \not\models F$ for some F . So $\mathcal{A}(\neg F) = 0$ (thus $\mathcal{A}(F) = 1$) for some model \mathcal{A} of T , and $\mathcal{B}(F) = 0$ for some model \mathcal{B} of T .

It follows $F \in Th(\mathcal{A}) \setminus Th(\mathcal{B})$, and so $Th(\mathcal{A}) \neq Th(\mathcal{B})$.

“ \Leftarrow :” Assume two models \mathcal{A} and \mathcal{B} of T are not elementarily equivalent. W.l.o.g. $Th(\mathcal{A}) \setminus Th(\mathcal{B}) \neq \emptyset$ and so $\mathcal{A}(F) = 1$ and $\mathcal{B}(F) = 0$ for some F .

We prove $T \not\models F$ and $T \not\models \neg F$. If $T \models F$, then, since $\mathcal{B} \models T$ we have $\mathcal{B} \models F$, contradicting $\mathcal{B}(F) = 0$. If $T \models \neg F$ then, since $\mathcal{A} \models T$, we have $\mathcal{A} \models \neg F$, contradicting $\mathcal{A}(F) = 1$.

Quantifier Elimination

Helpful lemmas

Recall that $\forall F := \forall x_1 \dots \forall x_n F$ where x_1, \dots, x_n are the free variables of F .

Lemma

Let S be a set of sentences. $S \models F$ iff $S \models \forall F$

Proof. Exercise.

Lemma

Let S be a set of sentences. If $S \models F \leftrightarrow G$ then $S \models H \leftrightarrow H[G/F]$.

Proof. By structural induction on H . Exercise.

Quantifier elimination

Definition

Let T be a set of formulas. We say that F and F' are T -equivalent if $T \models F \leftrightarrow F'$

Definition

A theory T admits quantifier elimination if for every formula F there is a quantifier-free T -equivalent formula G such that $fv(G) \subseteq fv(F)$. We call G a quantifier-free T -equivalent of F .

Examples

Find quantifier-free equivalent formulas in linear real arithmetic for:

$$\exists x \exists y (3 * x + 5 * y = 7) \leftrightarrow ?$$

$$\exists y (x < y \wedge y < z) \leftrightarrow ?$$

$$\forall y (x < y \wedge y < z) \leftrightarrow ?$$

Quantifier elimination

A **quantifier-elimination procedure (QEP)** for a theory T and a set of formulas \mathcal{F} is an algorithm that computes for every formula of \mathcal{F} a quantifier-free T -equivalent.

Lemma

Let T be a theory such that

- ▶ *T has a QEP for all formulas and*
- ▶ *$T \models G$ or $T \models \neg G$ for all ground formulas G (quantifier-free formula without occurrences of variables), and it is decidable which is the case.*

Then T is decidable and complete.

Quantifier elimination

Proof. Decidability.

This algorithm decides whether $T \models F$ for given F (sentence or not):

Compute a quantifier-free T -equivalent G of $\forall F$.

Decide whether $T \models G$ or $T \models \neg G$.

If $T \models G$ then answer $T \models F$, otherwise $T \not\models F$.

Correctness of the algorithm:

$$T \models F \quad \text{iff} \quad T \models \forall F \quad \text{iff} \quad T \models G$$

where we have made use of the lemmas.

Completeness. Exercise.

Simplifying quantifier elimination: one \exists

Fact

*If T has a QEP for all formulas of the form $\exists x F$, where F is quantifier-free,
then T has a QEP for all formulas.*

Essence: It is sufficient to be able to eliminate a single \exists

Construction:

Given: a QEP $qe1$ for formulas of the form $\exists x F$ where F is quantifier-free

Define: a QEP for all formulas

Method: Eliminate quantifiers bottom-up by $qe1$, use $\forall \equiv \neg \exists \neg$

Simplifying quantifier elimination: $\exists x \wedge$ literals

Fact

If T has a QEP for all $\exists x F$ where F is a conjunction of literals, all of which contain x , then T has a QEP for all $\exists x F$ where F is quantifier-free.

Construction:

Given: a QEP $qe1c$ for formulas of the form $\exists x (L_1 \wedge \dots \wedge L_n)$ where each L_i is a literal that contains x

Define: $qe1(\exists x F)$ where F is quantifier-free

Method: Put F in DNF. Distribute \exists over \vee . Apply $qe1c$.

This is the end of the generic part of quantifier elimination.
The rest is theory specific.

Simplifying quantifier elimination: Eliminating “ \neg ”

(Motivation: $\neg x < y \leftrightarrow y < x \vee y = x$ for linear orderings)

Fact

Assume that there is a computable function $aneg$ that maps every negated atom to a quantifier-free and negation-free T -equivalent formula.

If T has a QEP for all $\exists x F$ where F is a conjunction of atoms, all of which contain x , then T has a QEP for all $\exists x F$ where F is quantifier-free.

Construction:

Given: a QEP $qe1ca$ for formulas of the form $\exists x (A_1 \wedge \dots \wedge A_n)$ where each atom A_i contains x

Define: $qe1(\exists x F)$ where F quantifier-free

Method: Put F into NNF. Apply $aneg$. Put F in DNF. Distribute \exists over \vee . Apply $qe1ca$.

Quantifier Elimination
Dense Linear Orders
Without Endpoints

Dense Linear Orders Without Endpoints

Definition

Let $\Sigma = \{<, =\}$. The theory of **dense linear order without endpoints (DLO)** is the set of Σ -sentences that are consequences of the following set of axioms:

$$\forall x \forall y \forall z (x < y \wedge y < z \rightarrow x < z)$$

$$\forall x \neg(x < x)$$

$$\forall x \forall y (x < y \vee x = y \vee y < x)$$

$$\forall x \forall z (x < z \rightarrow \exists y (x < y \wedge y < z))$$

$$\forall x \exists y x < y$$

$$\forall x \exists y y < x$$

Models of DLO?

Theorem

All *countable* models of DLO are isomorphic.

Elimination of “ \neg ”

Fact

DLO has a computable function $aneg$ that maps every negated atom to a quantifier-free and negation-free DLO-equivalent formula.

$$DLO \models \neg(x = y) \leftrightarrow x < y \vee y < x$$

$$DLO \models \neg(x < y) \leftrightarrow x = y \vee y < x$$

$$aneg(\neg(x = y)) = x < y \vee y < x$$

$$aneg(\neg(x < y)) = x = y \vee y < x$$

Quantifier elimination for conjunctions of atoms

We define a QEP for formulas of the form $\exists x (A_1 \wedge \dots \wedge A_n)$, where x occurs in every A_i .

qe1ca($\exists x (A_1 \wedge \dots \wedge A_n)$):

- ▶ If some A_i is of the form $x = y$ (x and y different), apply:
$$\exists x (x = t \wedge F) \equiv F[t/x] \quad (x \text{ does not occur in } t)$$
and return $F[y/x]$.
- ▶ Drop all A_i of the form $y = y$. If no A_i left return \top .
- ▶ If some A_i is of the form $y < y$, return \perp .
- ▶ Separate the A_i into lower and upper bounds for x .
If no lower and/or no upper bounds, return \top . Otherwise use

$$DLO \models \exists x \left(\bigwedge_{i=1}^m l_i < x \wedge \bigwedge_{j=1}^n x < u_j \right) \leftrightarrow \bigwedge_{i=1}^m \bigwedge_{j=1}^n l_i < u_j$$

and return $\bigwedge_{i=1}^m \bigwedge_{j=1}^n l_i < u_j$.

Quantifier elimination for conjunctions of atoms

Example

$$\exists x (x < z \wedge y < x \wedge x < w) \equiv_{DOL} y < z \wedge y < w$$

$$\begin{aligned} \forall x \forall y (x < y) &\equiv_{DOL} \forall x \neg \exists y \neg (x < y) \\ &\equiv_{DOL} \forall x \neg \exists y (y < x \vee x = y) \\ &\equiv_{DOL} \forall x \neg (\exists y y < x \vee \exists y x = y) \\ &\equiv_{DOL} \forall x \neg (\top \vee x = x) \\ &\equiv_{DOL} \forall x \perp \\ &\equiv_{DOL} \perp \end{aligned}$$

$$\begin{aligned} \exists x \exists y \exists z (x < y \wedge y < z \wedge z < x) &\equiv_{DOL} \exists x \exists y (x < y \wedge y < x) \\ &\equiv_{DOL} \exists x x < x \\ &\equiv_{DOL} \perp \end{aligned}$$

Complexity

Quadratic blow-up with each elimination step

Therefore, eliminating all \exists from

$$\exists x_1 \dots \exists x_m F$$

where F has length n needs $O(n^{2^m})$ time and space, assuming F is DNF.

More efficient algorithms exist.

Consequences

- ▶ *DLO* has quantifier elimination.
- ▶ *DLO* is decidable and complete.
- ▶ All models of *DLO* (for example $(\mathbb{Q}, <)$ and $(\mathbb{R}, <)$) are elementarily equivalent:

We cannot distinguish models of *DLO* by first-order formulas.

Quantifier Elimination

Linear real arithmetic

Linear real arithmetic

Definition

Let $\mathcal{R}_+ = (\mathbb{R}, 0, 1, +, <, =)$. Linear real arithmetic is the theory $R_+ = Th(\mathcal{R}_+)$.

We allow the following additional function symbols:

For every $c \in \mathbb{Q}$:

- ▶ c is a constant symbol
- ▶ $c \cdot$, multiplication with c , is a unary function symbol

Example

$$\forall x \exists y (2.1x - y < 4.6 \wedge \forall z (7.3 < 2z + 4.7y \vee 3.25z > 2x))$$

Linear real arithmetic

Fact

Every formula with these additional function symbols can be effectively transformed into an \mathcal{R}_+ -equivalent formula of R_+ .

For example:

$$2.1(x - y) < 4.6$$

is \mathcal{R}_+ -equivalent to

$$\underbrace{(x + \cdots + x)}_{21 \text{ times}} < \underbrace{1 + \cdots + 1}_{46 \text{ times}} + \underbrace{y + \cdots + y}_{21 \text{ times}}$$

Linear real arithmetic

Definition

A term t is in **normal form** if $t = c_1 \cdot x_1 + \dots + c_n \cdot x_n + c$ where $c_i \neq 0$ and $x_i \neq x_j$ for every $1 \leq i \neq j \leq n$.

An atom A is in **normal form (NF)** if $A = 0 \bowtie t$ for some term t in normal form, where $\bowtie \in \{<, =\}$.

An atom is **solved for x** if it is of the form $x < t$, $x = t$ or $t < x$, where x does not occur in t .

Fact

Every atom is \mathcal{R}_+ -equivalent to an atom in normal form.

Any atom in normal form that contains x can be effectively transformed into a \mathcal{R}_+ -equivalent atom solved for x .

We let $sol_x(A)$ denote the result of solving an atom A for x .

Elimination of “ \neg ”

Fact

R_+ has a computable function $aneg$ that maps every negated atom to a quantifier-free and negation-free \mathcal{R}_+ -equivalent formula.

$$R_+ \models \neg(t = t') \leftrightarrow t < t' \vee t' < t$$

$$R_+ \models \neg(t < t') \leftrightarrow t = t' \vee t' < t$$

$$aneg(\neg(t = t')) = t < t' \vee t' < t$$

$$aneg(\neg(t < t')) = t = t' \vee t' < t$$

Fourier-Motzkin Elimination

We define a QEP for formulas of the form $\exists x (A_1 \wedge \dots \wedge A_n)$, where all A_i are atoms in NF and x occurs in every A_i .

qe1ca($\exists x (A_1 \wedge \dots \wedge A_n)$):

- ▶ Let $S = \{sol_x(A_1), \dots, sol_x(A_n)\}$
- ▶ If $(x = t) \in S$ for some t , apply:
$$\exists x (x = t \wedge F) \equiv F[t/x] \quad (x \text{ does not occur in } t)$$
and **return** $F[y/x]$.
- ▶ Separate the A_i into lower and upper bounds for x .
If no lower and/or no upper bounds, **return** \top . Otherwise

$$\text{return } \bigwedge_{(l < x) \in S} \bigwedge_{(x < u) \in S} l < u$$

All returned formulas are implicitly put into NF.

Fourier-Motzkin Elimination

Examples

$$\begin{aligned} & \exists x \exists y (3x + 5y < 7 \wedge 2x - 3y < 2) \\ \equiv_{\mathcal{R}_+} & \exists x \exists y (2/3x - 2/3 < y \wedge y < 7/5 - 3/5x) \\ \equiv_{\mathcal{R}_+} & \exists x (2/3x - 2/3 < 7/5 - 3/5x) \\ \equiv_{\mathcal{R}_+} & \exists x (x < 31/19) \\ \equiv_{\mathcal{R}_+} & \top \end{aligned}$$

$$\begin{aligned} & \exists x \forall y (3y \leq x \vee x \leq 2y) \\ \equiv_{\mathcal{R}_+} & \exists x \neg \exists y \neg (3y \leq x \vee x \leq 2y) \\ \equiv_{\mathcal{R}_+} & \exists x \neg \exists y \neg (x < 3y \wedge 2y < x) \\ \equiv_{\mathcal{R}_+} & \exists x \neg \exists y (1/3x < y \wedge y \leq 1/2x) \\ \equiv_{\mathcal{R}_+} & \exists x \neg (1/3x < 1/2x) \\ \equiv_{\mathcal{R}_+} & \exists x (1/3x \geq 1/2x) \\ \equiv_{\mathcal{R}_+} & \exists x (x \leq 0) \\ \equiv_{\mathcal{R}_+} & \top \end{aligned}$$

Can **DNF** (recall miniscoping) be avoided?

Ferrante and Rackoff's theorem

Theorem

Let F be quantifier-free and negation-free (not necessarily in DNF!) and assume all atoms that contain x are solved for x . Let

$$\begin{aligned} L &= \{l \mid (l < x) \in S_x\} & E &= \{t \mid (x = t) \in S_x\} \\ U &= \{u \mid (x < u) \in S_x\} \end{aligned}$$

where S_x is the set of atoms in F that contain x . Then:

$$R_+ \models \exists x F \leftrightarrow F[-\infty/x] \vee F[\infty/x] \vee \bigvee_{t \in E} F[t/x] \vee \bigvee_{l \in L} \bigvee_{u \in U} F[0.5(l + u)/x]$$

where $F[-\infty/x]$ ($F[\infty/x]$) is the result of applying this transformation to all solved atoms in F :

$$x < t \mapsto \top (\perp) \quad t < x \mapsto \perp (\top) \quad x = t \mapsto \perp (\perp)$$

Ferrante-Rackoff Procedure

$qe1(\exists x F)$:

1. Put F into NNF, eliminate all negations, put all atoms into normal form, solve those atoms for x that contain x .
2. Apply Ferrante and Rackoff's theorem.

Theorem

Eliminating all quantifiers with Ferrante and Rackoff's procedure from a formula of size n takes space $O(2^{cn})$ and time $O(2^{2^{dn}})$.

Quantifier Elimination Presburger Arithmetic

See [Harrison] or [Enderton] under “Presburger”

Presburger Arithmetic

Definition

Let $\mathcal{Z}_+ := (\mathbb{Z}, +, 0, 1, \leq)$. Linear integer arithmetic is the theory $Z_+ = Th(\mathcal{Z}_+)$.

We allow additional function symbols as for linear arithmetic:

For every $c \in \mathcal{Z}$:

- ▶ c is a constant symbol
- ▶ $c \cdot$, multiplication with c , is a unary function symbol

Fact

Linear integer arithmetic does not have quantifier elimination

Proof. Show that no quantifier-free formula is \mathcal{Z}_+ -equivalent to $\exists x x + x = y$.

Presburger Arithmetic

Definition

Let $\mathcal{P} := (\mathbb{Z}, +, 0, 1, \leq, 2 \mid, 3 \mid, \dots)$, where $k \mid$ is a unary predicate symbol interpreted as “ k divides ...”.

Presburger Arithmetic is the theory $P := Th(\mathcal{P})$.

Definition

An atom A is in normal form if

$$A = 0 \leq c_1 \cdot x_1 + \dots + c_n \cdot x_n + c \quad \text{or}$$

$$A = k \mid c_1 \cdot x_1 + \dots + c_n \cdot x_n + c$$

where $c_i \in \mathbb{Z} \setminus \{0\}$ and $k \geq 1$

Where necessary, atoms are put into normal form

Fact

Every atom is \mathcal{P} -equivalent to an atom in normal form.

Elimination of “ \neg ”

Fact

P has a computable function *aneg* that maps every negated atom to a quantifier-free and negation-free \mathcal{P} -equivalent formula.

$$\mathcal{P} \models \neg(s \leq t) \leftrightarrow t + 1 \leq s$$

$$\mathcal{P} \models \neg(k \mid t) \leftrightarrow k \mid t + 1 \vee k \mid t + 2 \vee \dots \vee k \mid t + (k - 1)$$

$$\text{aneg}(\neg(s \leq t)) = t + 1 \leq s$$

$$\text{aneg}(\neg(k \mid t)) = k \mid t + 1 \vee k \mid t + 2 \vee \dots \vee k \mid t + (k - 1)$$

Quantifier Elimination for \mathcal{P}

We define a QEP for formulas of the form $\exists x (A_1 \wedge \cdots \wedge A_n)$, where all A_i are atoms in NF and x occurs in every A_i .

$qe1ca(\exists x (A_1 \wedge \cdots \wedge A_n))$ proceeds in two steps.

Let $F = A_1 \wedge \cdots \wedge A_n$.

Step 1: Set all coeffs of x in F to 1 or -1:

1. Set all coeffs of x in F to the lcm m of all coeffs of x
2. Set all coeffs of x to 1 or -1 and add $\wedge m \mid x$

Step 2: Separate into upper and lower bounds, with some new reasoning because of the $k \mid t$ atoms.

Quantifier Elimination for \mathcal{P} : Step 1

The result of Step 1 is

$$R := \text{coeff1}(A_1) \wedge \cdots \wedge \text{coeff1}(A_l) \wedge m \mid x$$

where

- ▶ m is the (positive) lcm of all coefficients of x in F (e.g. $\text{lcm}\{-6, 9\} = 18$), and
- ▶ $\text{coeff1}(0 \leq \sum_{i=1}^n c_i x_i + c) = (0 \leq \sum_{i=1}^n c'_i x_i + c')$ and
 $\text{coeff1}(d \mid \sum_{i=1}^n c_i x_i + c) = (d' \mid \sum_{i=1}^n c'_i x_i + c')$

where, assuming x is the k -th variable x_k and letting $m' = m/|c_k|$ we set:

$$\begin{array}{ll} c'_i = m' \cdot c_i \text{ if } i \neq k & c'_k = \text{if } c_k > 0 \text{ then } 1 \text{ else } -1 \\ c' = m' \cdot c & d' = m' \cdot d \end{array}$$

Lemma $\mathcal{P} \models (\exists x F) \leftrightarrow (\exists x R)$

Quantifier Elimination for \mathcal{P} : Step 2

The result of Step 2 is the quantifier-free formula

$$R' := \text{if } L = \emptyset \text{ then } \bigvee_{i=0}^{m-1} \bigwedge D[i/x] \text{ else } \bigvee_{i=0}^{m-1} \bigvee_{l \in L} R[l + i/x]$$

where

$$L := \{-t \mid (0 \leq x + t) \in R\}$$

$$U := \{t \mid (0 \leq -x + t) \in R\}$$

$$D := \{(d \mid t) \in R\}$$

$$m := (\text{positive}) \text{ lcm of } \{d \mid (d \mid t) \in D \text{ for some } t\}$$

The correctness of this step follows from the

Lemma (Periodicity Lemma)

If $A \in D$, i.e. $A = (d \mid x + t)$ and $x \notin \text{fv}(t)$, and $i \equiv j \pmod{d}$ then $\mathcal{P} \models A[i/x] \leftrightarrow A[j/x]$.

Example

We eliminate the existential quantifier of

$$\exists x (3x - y + 1 > 0) \wedge (2x - 6 < z) \wedge (4 \mid 5x + 1)$$

We set all coefficients of x to 1. With $\text{lcm}\{3, 2, 5\} = 30$ we get:

$$\exists x (30x - 10y + 10 > 0) \wedge (30x - 90 < 15z) \wedge (24 \mid 30x + 6)$$

We rescale $x := 30x$ adding the atom $(30 \mid x)$ and split the inequalities in lower and upper bounds:

$$\exists x (10y - 10 < x) \wedge (x < 90 + 15z) \wedge (24 \mid x + 6) \wedge (30 \mid x)$$

Example

Periodicity: if $(24 \mid x + 6) \wedge (30 \mid x)$ holds for some x , then it also holds for every $x + k \cdot \text{lcm}\{24, 30\} = x + k \cdot 120$ where $k \in \mathbb{Z}$.

Therefore: $(10y - 10 < x) \wedge (x < 90 + 15z)$ is satisfied by x iff it is satisfied by $10y - 10 + i$ for some $1 \leq i \leq 120$.

So $\exists x (10y - 10 < x < 90 + 15z) \wedge (24 \mid x + 6) \wedge (30 \mid x)$ is \mathcal{P} -equivalent to

$$\bigvee_{i=1}^{120} \left(\begin{array}{c} 10y - 10 < 10y - 10 + i < 90 + 15z \\ \wedge \\ (24 \mid 10y - 10 + i + 6) \wedge (30 \mid 10y - 10 + i) \end{array} \right)$$

and so \mathcal{P} -equivalent to:

$$\bigvee_{i=1}^{120} \left(\begin{array}{c} (10y + i < 100 + 15z) \\ \wedge \\ (24 \mid 10w - 4 + i) \wedge (30 \mid 10w - 10 + i) \end{array} \right)$$

Incompleteness of (Integer) Arithmetic

[Schöning, van Glabbeek]



Kurt Gödel. *Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I*. 1931.

Kurt Gödel

1906 — 1978
Brünn Princeton



Arithmetic and notation

We consider formulas over the signature $\Sigma = \{+, *, <, =\}$.

Arithmetic is the set of all Σ -sentences that are true in the interpretation with universe \mathbb{N} and where $+, *, <, =$ are interpreted in the standard way.

(We substitute \mathbb{N} for \mathbb{Z} for convenience, it is an inessential detail.)

We denote the set of all sentences of arithmetic by W .

$F(x_1, \dots, x_k)$ denotes a formula in which at most the variables x_1, \dots, x_k occur free.

If $n_1, \dots, n_k \in \mathbb{N}$ then $F(n_1, \dots, n_k)$ is the result of substituting n_1, \dots, n_k for the free occurrences of x_1, \dots, x_k .

Example

$$F(x, y) = (x = y \wedge \exists x. x = y)$$

$$F(5, 7) = (5 = 7 \wedge \exists x. x = 7)$$

Arithmetically representable functions and relations

Formulas with free variables can **represent** functions and relations.

The formula

$$F(x, y) = (\exists z. y = x + z + 1)$$

represents the relation “ $x < y$ ”

The formula

$$F(x, y, z) = (\exists k. x = k * y + z \wedge z < y)$$

represents the relation “ $z = x \bmod y$ ”.

Definition

A k -ary relation $R \subseteq \mathbb{N}^k$ is **arithmetically representable** iff there is a formula $F(x_1, \dots, x_k)$ s.t. for all $n_1, \dots, n_k \in \mathbb{N}$:

$$(n_1, \dots, n_k) \in R \quad \text{iff} \quad F(n_1, \dots, n_k) \in W$$

We call F a **representation** of R .

Representing the transition relation of Turing machines

Given a deterministic Turing machine M with states $\{0, \dots, n\}$ over the tape alphabet $\{0, 1\}$, we encode a configuration c of M as a tuple $\underline{c} = (l, q, r) \in \mathbb{N}^3$ where

- ▶ q encodes the state of M ;
- ▶ l encodes the **left-string**: the string to the left of the head, read as a binary number with an additional leading 1;
- ▶ r encodes the **right-string**: the string including the square where the head is, and extending to the right, read *in reverse* as a binary number with an additional leading 1.

Definition

The **transition relation of M** is the relation $T_M \subseteq \mathbb{N}^6$ given by

$$T_M = \{(\underline{c}_1, \underline{c}_2) \mid \underline{c}_2 \text{ is the successor configuration of } \underline{c}_1\} .$$

Representing the transition relation of Turing machines

Lemma

For every Turing machine M the relation T_M is arithmetically representable.

Proof idea: Let $c_1 \xrightarrow{q,a} c_2$ denote that c_1 is a configuration with state q where the head reads a and c_2 is the successor of c_1

For every state q and symbol a of M define

$$T_M^{q,a} := \{(c_1, c_2) \mid c_1 \xrightarrow{q,a} c_2\}$$

and define an arithmetic representation of $F_M^{(q,a)}$.

For example, if $\delta(3, 0) = (5, 1, R)$ then define

$$\begin{aligned} & F_M^{3,0}(l_1, q_1, r_1, l_2, q_2, r_2) \\ & =: (q_1 = 3 \wedge q_2 = 5 \wedge l_2 = l_1 * 2 + 1 \wedge r_1 = r_2 * 2) \end{aligned}$$

The formula $F_M := \bigvee_{q \in Q_M, a \in \Sigma_M} F_M^{q,a}$ is a representation of T_M

Representing the reachability relation of Turing machines

Lemma

For every Turing machine M the transitive closure T_M^* of T_M is arithmetically representable.

We only sketch the proof of a weaker result.

Given a formula $F(x, y)$ of arithmetic representing a binary relation R we can effectively construct a formula $F^*(x, y)$ of **arithmetic with exponentiation** representing R^* .

A full proof of the lemma requires to express exponentiation in arithmetic and extend the result to formulas $F(\vec{x}, \vec{y})$.

Key idea of the proof: encode a sequence

$a_n, a_{n-1}, \dots, a_0 \in (\mathbb{N} \setminus \{0\})^*$ of arbitrary length as a pair $(t, p) \in \mathbb{N}^2$ where

- ▶ $p > a_i$ for all $0 \leq i \leq n$ and
- ▶ the word $a_n \dots a_1 a_0 \in [p]^*$ is the p -ary representation of t .

Representing the reachability relation of Turing machines

Represent the relation “ $y = a_0$ ” by the formula

$$Last(t, p, y) = y < p \wedge \exists x (t = x * p + y)$$

Represent “ $x = a_n$ ” by

$$First(t, p, x) = x < p \wedge \exists z \exists w (t = x * p^z + w \wedge w < p^z)$$

Represent “ v comes after u ” by

$$\begin{aligned} Next(t, p, u, v) = & u < p \wedge v < p \wedge \\ & \exists i \exists y \exists z (t = y * p^{i+2} + u * p^{i+1} + v * p^i + z \\ & \wedge z < p^x \wedge y + u > 0) \end{aligned}$$

Take

$$\begin{aligned} F^*(x, y) = & \exists t \exists p (First(t, p, x) \wedge Last(t, p, y) \wedge \\ & \forall u \forall v (Next(t, p, u, v) \rightarrow F(u, v))) \end{aligned}$$

Arithmetic is not semi-decidable

Theorem

W is not semi-decidable.

Proof. By reduction from the set of all pairs (M, x) where M is a Turing machine, x is an input for M , and M does not halt on x . This set is known to not be semi-decidable.

Let M be a Turing machine with states $\{0, \dots, n\}$ and let x be an input for M . Assume n is the only final state.

Let F_M be a representation of the transition relation T_M of M . Let c_0 be the initial configuration of M on input x .

Define

$$NH_{M,x} = \neg \exists l \exists q \exists r (F_M^*(c_0, l, q, r) \wedge q = n)$$

We have : $NH_{M,x} \in W$ iff M does not halt on input x .

Proof systems

What is a *proof system*? Minimal requirement: It must be decidable if a given text is a proof of a given formula.

We encode texts as natural numbers.

Definition

Let S be the set of all sentences over the signature of arithmetic. A **proof system** for arithmetic is a decidable predicate

$$Prf : \mathbb{N} \times S \rightarrow \{0, 1\}$$

(Read $Prf(p, F)$ as “ p is a proof of F ”.)

A proof system Prf is **correct** or **sound** iff $Prf(p, F)$ implies $F \in W$. (“Everything provable is true.”)

A proof system Prf is **complete** iff for every $F \in W$ there exists a proof p such that $Prf(p, F)$. (“Everything true is provable.”)

Gödel's Incompleteness Theorem

Theorem (Gödel)

There is no correct and complete proof system for arithmetic.

Proof. Assume there exists a correct and complete proof system. The following procedure semi-decides W :

```
Input: sentence  $F$ 
 $p := 0$ ;
while  $\text{Prf}(p, F) = 0$  do  $p := p + 1$ ;
output("  $F \in W$  ")
```

Corollary

For every correct proof system for arithmetic there exists a sentence F such that neither F nor $\neg F$ can be proved.

Hilbert's 10th Problem

Given a diophantine equation: To devise a process according to which it can be determined by a finite number of operations whether the equation is solvable in integers.

Hilbert, ICM, Paris, 1900

Theorem (Matiyasevich, Robinson, Davis, Putnam, 1949-1970)

It is undecidable if a diophantine equation has a solution.



Higher-Order Logic (HOL)

Types and Terms

Simply typed λ -terms

Types:

$$\begin{array}{l} \tau ::= \text{bool} \mid \dots \\ \quad \mid (\tau \rightarrow \tau) \\ \quad \mid \alpha \mid \beta \dots \end{array}$$

Terms

$$\begin{array}{l} t ::= c \mid d \mid \dots \mid f \mid h \mid \dots \\ \quad \mid (t \ t) \\ \quad \mid (\lambda x. t) \end{array}$$

We assume that every variable and constant has an attached type.

We consider only well-typed terms:

$$\frac{t_1 : \tau \rightarrow \tau' \quad t_2 : \tau}{t_1 \ t_2 : \tau'} \qquad \frac{t : \tau'}{\lambda x : \tau. t : \tau \rightarrow \tau'}$$

Base logic

Formula = term of type *bool*

Theorems: $\Gamma \vdash F$

Base constants: $= : \alpha \rightarrow \alpha \rightarrow \mathit{bool}$
 $\rightarrow : \mathit{bool} \rightarrow \mathit{bool} \rightarrow \mathit{bool}$

Inference rules

$$\overline{F \vdash F} \text{ assume}$$

$$\overline{\vdash t = t} \text{ refl}$$

$$\overline{\vdash (\lambda x. t) u = u[t/x]} \beta$$

$$\overline{\vdash \lambda x. (t x) = t} \eta \quad \text{if } x \notin \text{fv}(t)$$

$$\frac{\Gamma_1 \vdash s = t \quad \Gamma_2 \vdash F[s/x]}{\Gamma_1 \cup \Gamma_2 \vdash F[t/x]} \text{ subst}$$

$$\frac{\Gamma \vdash s = t}{\Gamma \vdash (\lambda x. s) = (\lambda x. t)} \text{ abs} \quad \text{if } x \notin \text{fv}(\Gamma)$$

Inference rules

$$\frac{\Gamma \vdash F}{\Gamma \vdash F[\tau_1/\alpha_1, \dots]} \textit{inst}$$

if α_1, \dots do not occur in Γ

Inference rules

$$\frac{\Gamma \vdash G}{\Gamma \setminus \{F\} \vdash F \rightarrow G} \rightarrow I$$

$$\frac{\Gamma_1 \vdash F \rightarrow G \quad \Gamma_2 \vdash F}{\Gamma_1 \cup \Gamma_2 \vdash G} \rightarrow E$$

$$\frac{\Gamma_1 \vdash F \rightarrow G \quad \Gamma_2 \vdash G \rightarrow F}{\Gamma_1 \cup \Gamma_2 \vdash F = G} =I$$

Definitions of standard logical symbols

$$\vdash \top = ((\lambda x. x) = (\lambda x. x))$$

$all : (\alpha \rightarrow bool) \rightarrow bool$

Notation: $\forall x. F$ abbreviates $all(\lambda x. F)$

$$\vdash all = (\lambda P. P = (\lambda x. \top))$$

$$\vdash \perp = (\forall F. F)$$

$$\vdash \neg = (\lambda F. F \rightarrow \perp)$$

$$\vdash (\wedge) = (\lambda F. \lambda G. \forall H. (F \rightarrow G \rightarrow H) \rightarrow H)$$

$$\vdash (\vee) = (\lambda F. \lambda G. \forall H. (F \rightarrow H) \rightarrow (G \rightarrow H) \rightarrow H)$$

Definitions of standard logical symbols

$ex : (\alpha \rightarrow bool) \rightarrow bool$

Notation: $\exists x. F$ abbreviates $ex(\lambda x. F)$

$\vdash ex = (\lambda P. \forall G. (\forall x. (P x \rightarrow G) \rightarrow G))$

The method of postulating what we want has many advantages; they are the same as the advantages of theft over honest toil.

Bertrand Russel

Classical logic

$$\vdash F \vee \neg F$$

Hilbert's ε

Informally: $\varepsilon x. F$ = an arbitrary but fixed x that satisfies F

Examples

$$(\varepsilon x. x = 5) = 5$$

$$(\varepsilon n. 0 \leq n \leq 2) \in \{0, 1, 2\}$$

$$(\varepsilon x. \perp) \quad ???$$

Formally: $eps : (\alpha \rightarrow bool) \rightarrow \alpha$

$\varepsilon x. F$ abbreviates $eps(\lambda x. F)$

Axiom: $P x \rightarrow P(eps P)$