Propositional Logic
Basics

Syntax of propositional logic

Definition

An atomic formula (or atom) has the form A_i where $i = 1, 2, 3, \ldots$.
Formulas are defined inductively:

- $\bot$ ("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 $A \rightarrow B \wedge \neg(C \vee D) \vee E$.

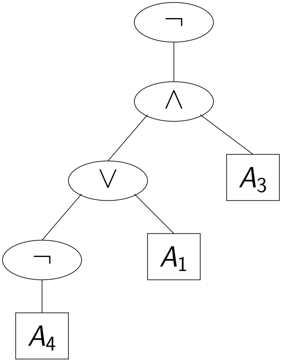
Outermost parentheses can be dropped.

Syntax tree of a formula

Every formula can be represented by a syntax tree.

Example

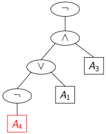
$F = \neg((\neg A_4 \vee A_1) \wedge A_3)$



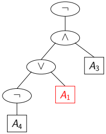
Subformulas

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

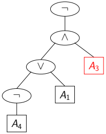
A_4



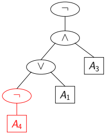
A_1



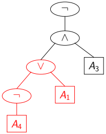
A_3



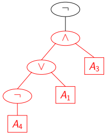
$\neg A_4$



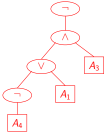
$(\neg A_4 \vee A_1)$



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



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



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 $\mathcal{P}(\bot)$, 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 $\mathcal{A} : \mathit{Atoms} \rightarrow \{0,1\}$
where *Atoms* is the set of all atoms.

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

Semantics of propositional logic (II)

$$\begin{array}{rcl}
\hat{\mathcal{A}}(A_i) & = & \mathcal{A}(A_i) \\
\hat{\mathcal{A}}(\neg F) & = & \begin{cases} 1 & \text{if } \hat{\mathcal{A}}(F) = 0 \\ 0 & \text{otherwise} \end{cases} \\
\hat{\mathcal{A}}(F \wedge G) & = & \begin{cases} 1 & \text{if } \hat{\mathcal{A}}(F) = 1 \text{ and } \hat{\mathcal{A}}(G) = 1 \\ 0 & \text{otherwise} \end{cases} \\
\hat{\mathcal{A}}(F \vee G) & = & \begin{cases} 1 & \text{if } \hat{\mathcal{A}}(F) = 1 \text{ or } \hat{\mathcal{A}}(G) = 1 \\ 0 & \text{otherwise} \end{cases} \\
\hat{\mathcal{A}}(F \rightarrow G) & = & \begin{cases} 1 & \text{if } \hat{\mathcal{A}}(F) = 0 \text{ or } \hat{\mathcal{A}}(G) = 1 \\ 0 & \text{otherwise} \end{cases}
\end{array}$$

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 $\ldots$ instead of $A_1, A_2, A_3 \ldots$

$$F_1 \leftrightarrow F_2 \quad \text{abbreviates} \quad (F_1 \wedge F_2) \vee (\neg F_1 \wedge \neg F_2)$$

$$\bigvee_{i=1}^{n} F_i \quad \text{abbreviates} \quad (\ldots((F_1 \vee F_2) \vee F_3) \vee \ldots \vee F_n)$$

$$\bigwedge_{i=1}^{n} F_i \quad \text{abbreviates} \quad (\ldots((F_1 \wedge F_2) \wedge F_3) \wedge \ldots \wedge F_n)$$

Special cases:

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

Truth tables (I)

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

$\neg$	A
1	0
0	1

A	$\vee$	B
0	0	0
0	1	1
1	1	0
1	1	1

A	$\wedge$	B
0	0	0
0	0	1
1	0	0
1	1	1

A	$\rightarrow$	B
0	1	0
0	1	1
1	0	0
1	1	1

A	$\leftrightarrow$	B
0	1	0
0	0	1
1	0	0
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 assigment that is a model of every formula in S.

Exercise

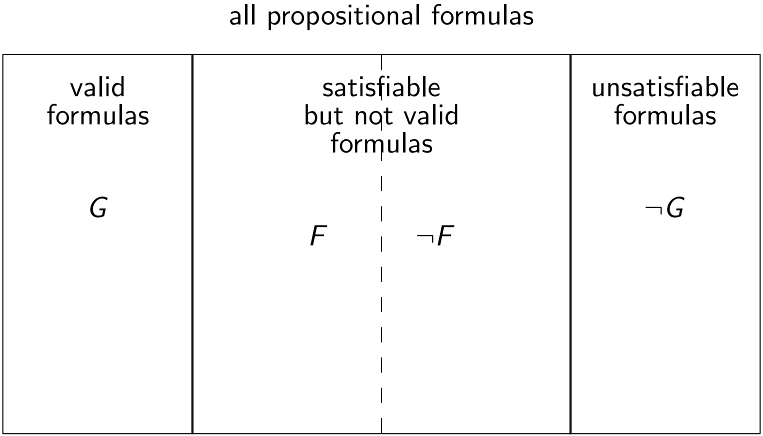
	Valid	Satisfiable	Unsatisfiable
A			
$A \vee B$			
$A \vee \neg A$			
$A \wedge \neg A$			
$A \rightarrow \neg A$			
$A \rightarrow (B \rightarrow A)$			
$A \rightarrow (A \rightarrow B)$			
$A \leftrightarrow \neg A$			

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



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

$A \lor B,\ A \to B,\ B \land R \to \neg A,\ R \ \models\ (R \land \neg A) \land B$

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 $\mathcal{A} \models A \rightarrow B$
- $A \models \neg A$ 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. $\models (\bigwedge_{i=1}^{k} F_i) \to G$

Proof of "if $F_1, \dots, F_k \models G$ then $\models \underbrace{(\bigwedge_{i=1}^{k} F_i) \to G}_{H}$".

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 $H = \bigwedge F_i \to G$.
- $\mathcal{A}(\bigwedge F_i) = 1$: 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. $F \to G$ *is valid.*
2. $F \land \neg G$ *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:

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

$\models S$ means that for all $F \in S$, $\models F$

$M \models S$ 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?

$$
\begin{aligned}
(A \wedge (A \vee B)) &\equiv A \\
(A \wedge (B \vee C)) &\equiv ((A \wedge B) \vee C) \\
(A \rightarrow (B \rightarrow C)) &\equiv ((A \rightarrow B) \rightarrow C) \\
(A \rightarrow (B \rightarrow C)) &\equiv ((A \wedge B) \rightarrow C) \\
(A \rightarrow B) &\equiv (\neg A \vee B) \\
(A \rightarrow B) &\equiv (\neg A \rightarrow \neg B) \\
(A \leftrightarrow (B \leftrightarrow C)) &\equiv ((A \leftrightarrow B) \leftrightarrow C)
\end{aligned}
$$

Observation

The following connections hold:

$$\models F \to G \quad \text{iff} \quad F \models G$$
$$\models F \leftrightarrow G \quad \text{iff} \quad F \equiv G$$

NB: “iff” means “if and only if”

Reductions between problems (I)

- Validity to Unsatisfiability:

$$F \text{ valid} \quad \text{iff} \quad \textbf{?}\neg F \text{ unsatisfiable}$$

- Unsatisfiability to Validity:

$$F \text{ unsatisfiable} \quad \text{iff} \quad \textbf{?}\ \neg F \text{ valid}$$

- Validity to Consequence:

$$F \text{ valid} \quad \text{iff} \quad \textbf{?} \models \textbf{?}\ \top \models F$$

- Consequence to Validity:

$$F \models G \quad \text{iff} \quad \textbf{?}\ F \rightarrow G \text{ valid}$$

- Validity to Equivalence:

$$F \text{ valid} \quad \text{iff} \quad \textbf{?} \equiv \textbf{?}\ F \equiv \top$$

- Equivalence to Validity:

$$F \equiv G \quad \text{iff} \quad \textbf{?}\ F \leftrightarrow G \text{ valid}$$

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

$$
\begin{array}{rcll}
(F \wedge F) & \equiv & F & \\
(F \vee F) & \equiv & F & \textit{(Idempotence)} \\
(F \wedge G) & \equiv & (G \wedge F) & \\
(F \vee G) & \equiv & (G \vee F) & \textit{(Commutativity)} \\
((F \wedge G) \wedge H) & \equiv & (F \wedge (G \wedge H)) & \\
((F \vee G) \vee H) & \equiv & (F \vee (G \vee H)) & \textit{(Associativity)} \\
(F \wedge (F \vee G)) & \equiv & F & \\
(F \vee (F \wedge G)) & \equiv & F & \textit{(Absorption)}
\end{array}
$$

Equivalences (II)

$$\begin{array}{rcll}
(F \wedge (G \vee H)) & \equiv & ((F \wedge G) \vee (F \wedge H)) & \\
(F \vee (G \wedge H)) & \equiv & ((F \vee G) \wedge (F \vee H)) & \text{(Distributivity)} \\
\neg\neg F & \equiv & F & \text{(Double negation)} \\
\neg(F \wedge G) & \equiv & (\neg F \vee \neg G) & \\
\neg(F \vee G) & \equiv & (\neg F \wedge \neg G) & \text{(deMorgan's Laws)} \\
\neg\top & \equiv & \bot & \\
\neg\bot & \equiv & \top & \\
(\top \vee G) & \equiv & \top & \\
(\top \wedge G) & \equiv & G & \\
(\bot \vee G) & \equiv & G & \\
(\bot \wedge G) & \equiv & \bot &
\end{array}$$

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	$A_1 \vee \neg A_1$
$\bot$	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 formal 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

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

("$F \equiv G \equiv H$" is an abbreviation for "$F \equiv G$ and $G \equiv H$")

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:

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

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

Definition

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

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

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

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:

$$
\begin{array}{rcl}
(F \lor (G \land H)) & \equiv & ((F \lor G) \land (F \lor H)) \\
((F \land G) \lor H) & \equiv & ((F \lor H) \land (G \lor H))
\end{array}
$$

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

$$
\begin{array}{rcl}
(F \land (G \lor H)) & \equiv & ((F \land G) \lor (F \land H)) \\
((F \lor G) \land H) & \equiv & ((F \land H) \lor (G \land H))
\end{array}
$$

Termination

Why does the transformation into NNF and CNF terminate?

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

$$
\begin{array}{rcl}
\neg\neg F & \equiv & F \\
\neg(F \land G) & \equiv & (\neg F \lor \neg G) \\
\neg(F \lor G) & \equiv & (\neg F \land \neg G) \\
(F \lor (G \land H)) & \equiv & ((F \lor G) \land (F \lor H)) \\
((F \land G) \lor H) & \equiv & ((F \lor H) \land (G \lor H))
\end{array}
$$

Define w recursively:

$w(A_i) = \dots$

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

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

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

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 instist 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

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

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 $F \neq A$: $\mathcal{A}(F[G/A]) = \mathcal{A}(F) = \mathcal{A}'(F)$
- $F = F_1 \wedge F_2$:

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

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 equivalid 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 $K \to B$ where

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

$$\begin{array}{rcll} A & \equiv & (\top \to A) & \text{fact} \\ (\neg A \vee \neg B \vee C) & \equiv & (A \wedge B \to C) & \text{rule} \\ (\neg A \vee B) & \equiv & (A \to B) & \text{rule} \\ \neg A & \equiv & (A \to \bot) & \text{goal} \\ (\neg A \vee \neg B) & \equiv & (A \wedge B \to \bot) & \text{goal} \end{array}$$

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 $K \to B$
such that $\mathcal{M}(K) = 1$ and $\mathcal{M}(B) = 0$
do
if $B = \bot$ **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 $(A_{i_1} \wedge \ldots \wedge A_{i_k} \rightarrow \bot)$ be the subformula causing "unsatisfiable".
Since $\mathcal{M}(A_{i_1}) = \cdots = \mathcal{M}(A_{i_k}) = 1$, $\mathcal{A}(A_{i_1}) = \ldots = \mathcal{A}(A_{i_k}) = 1$.
Then $\mathcal{A}(A_{i_1} \wedge \ldots \wedge A_{i_k} \rightarrow \bot) = 0$ 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

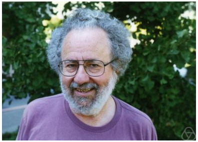
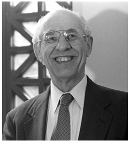
A satisfiable Horn formula has a unique model with a smallest number of true atoms.

Propositional Logic
DPLL: Davis-Putnam-Logemann-Loveland

Davis–Putnam–Logemann–Loveland

DPLL algorithm:

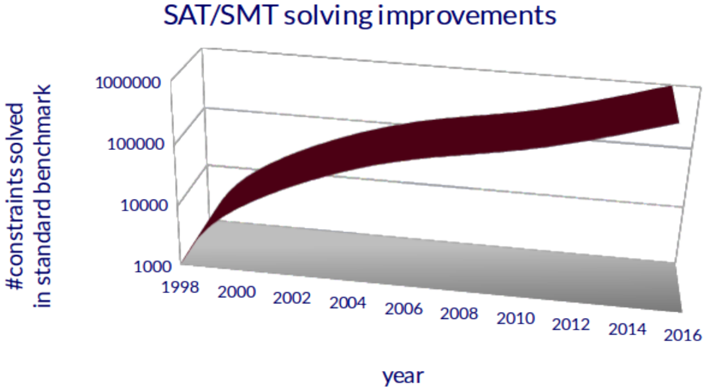
- combines search and deduction to decide satisfiability
- underlies most modern SAT solvers
- is over 50 years old

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

Representation as set of sets of literals:

$$\{\underbrace{\{L_{1,1},\ \ldots,\ L_{1,n_1}\}}_{\textit{clause}},\ \ldots,\ \{L_{k,1},\ \ldots,\ L_{1,n_k}\}\}$$

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:

$$F \equiv (A \land F[\top/A]) \lor (\neg A \land F[\bot/A]).$$

Proof By structural induction on F (exercise).

Corollary

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

DPLL — First step: partial evaluation

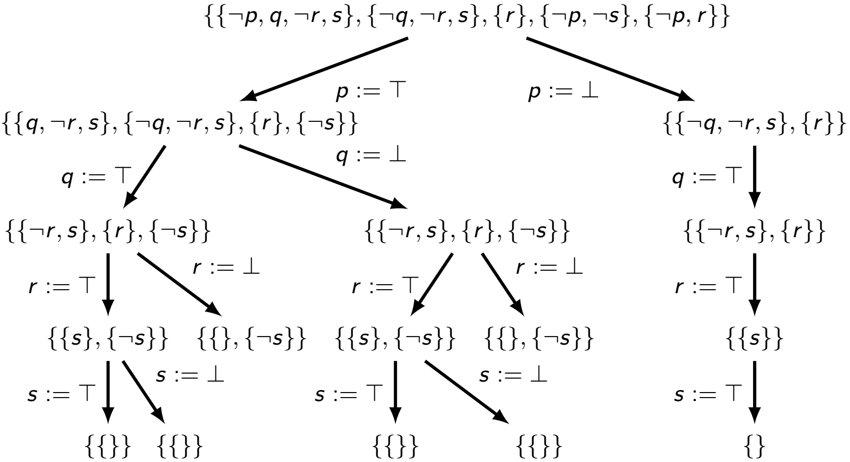
$F[\bot/A]$ and $F[\top/A]$ easy to compute in clause normal form:

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

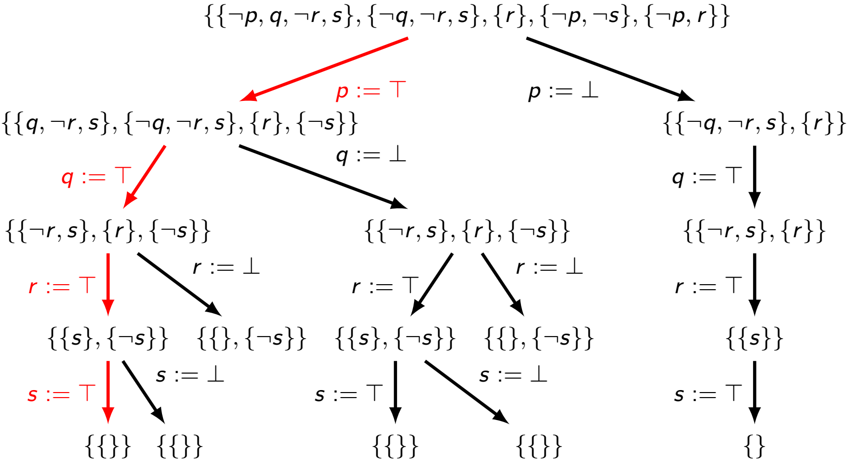
Partial evaluation algorithm:

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 $F[\bot/A]$ is satisfiable;
  if yes, return satisfiable.
  Recursively check if $F[\top/A]$ is satisfiable;
  if yes, return satisfiable, otherwise unsatisfiable.

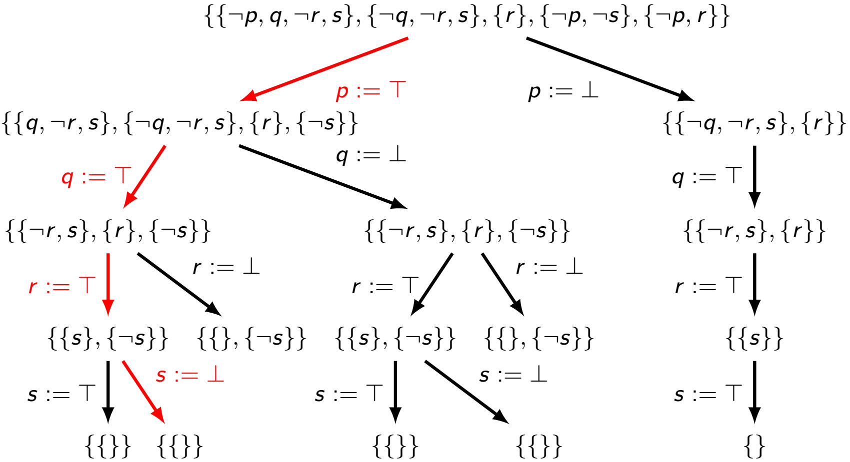
DPLL: Davis-Putnam-Logemann-Loveland



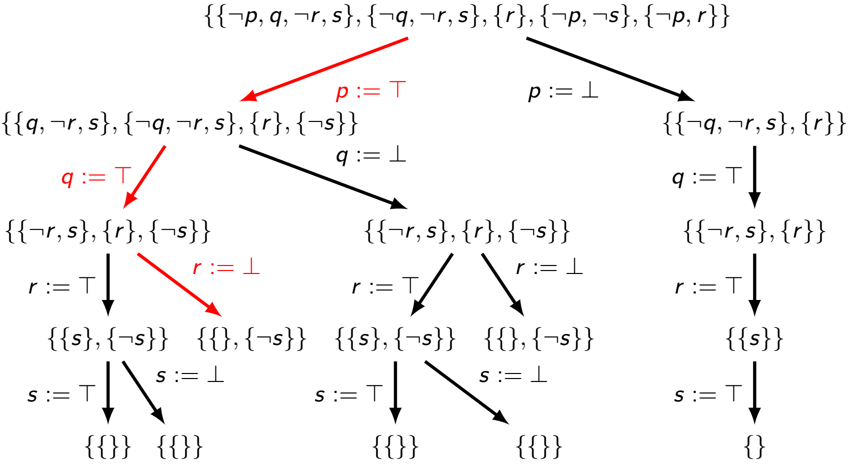
DPLL: Davis-Putnam-Logemann-Loveland



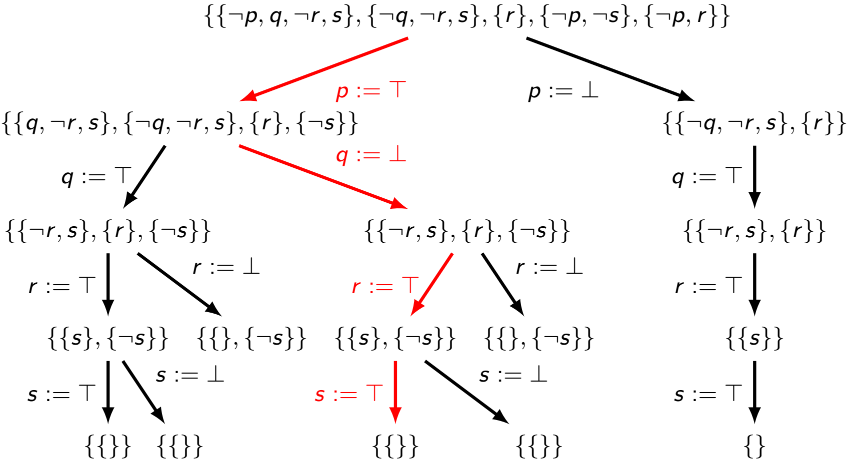
DPLL: Davis-Putnam-Logemann-Loveland



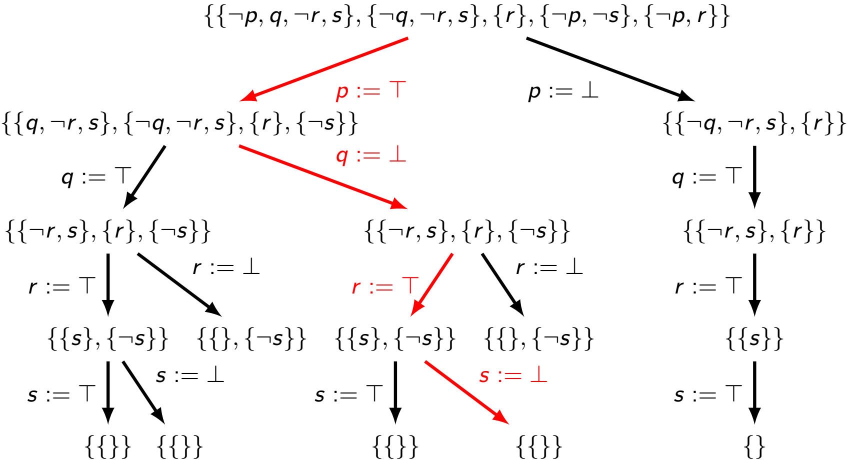
DPLL: Davis-Putnam-Logemann-Loveland



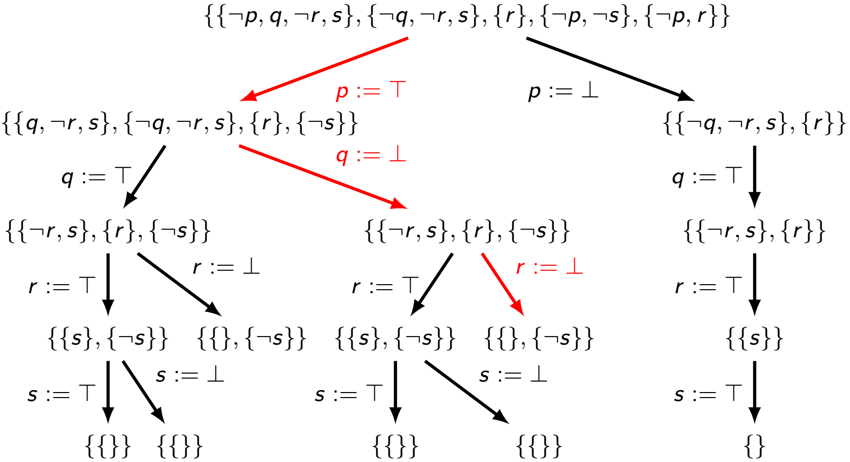
DPLL: Davis-Putnam-Logemann-Loveland



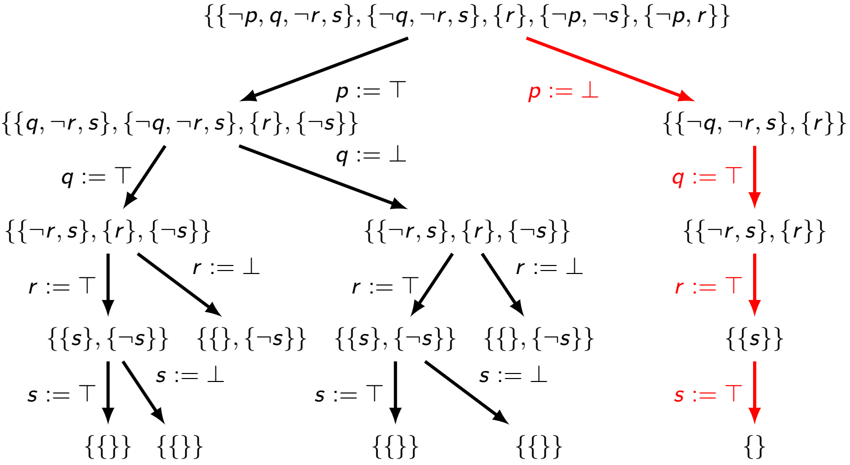
DPLL: Davis-Putnam-Logemann-Loveland



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 $\overline{L}$ 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

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

$$\Big\downarrow \text{OLR } r := \top$$

$$\{\{\neg p, q, s\}, \{\neg q, s\}, \{\neg p, \neg s\}\}$$

$$\Big\downarrow \text{PLR } p := \bot$$

$$\{\{\neg q, s\}\}$$

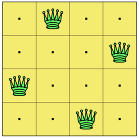
$$\Big\downarrow \text{PLR } q := \bot$$

$$\{\}$$

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 unsatisfiabiity.
- 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 $c : V \to \{1, 2, 3, 4\}$ 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 $V = \{v_1, v_2, \ldots\}$ 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 $c \in \{1, \ldots, 4\}$ and $i \in \mathbb{N}$

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

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 $i = 1, \ldots, n$ 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 $b_1 b_2 \cdots$ 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 $b_1 \cdots b_m$–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 $\overline{L}$ is defined as follows:

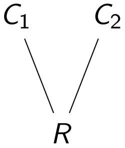
$$\overline{L} = \begin{cases} \neg A_i & \text{if } L = A_i \\ A_i & \text{if } L = \neg A_i \end{cases}$$

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

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

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

Recall: $\square$ represents $\bot$.

Resolution proof

Definition

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

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

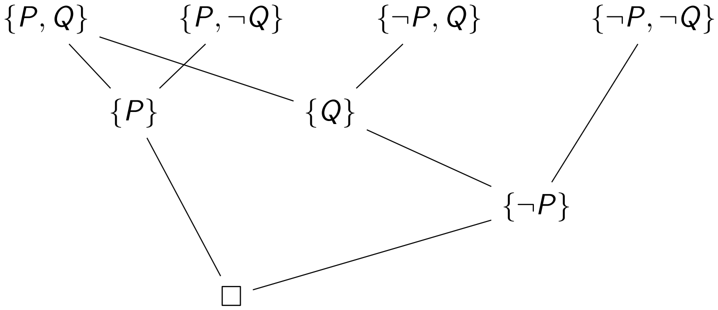
Then we can write $F \vdash_{Res} C$.

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 $R = (C_1 - \{L\}) \cup (C_2 - \{\overline{L}\})$ (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 $\mathcal{A} \models C_2 - \{\overline{L}\}$ (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 $C_0, \ldots, C_n = C$.
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 $F \vdash_{Res} \Box$ then F is unsatisfiable.

Completeness of resolution

Theorem

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

Theorem (Completeness of resolution)

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

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 $F \vdash_{Res} \Box$.

Completeness proof

Corollary

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

Idea for completeness proof:
If A is an atom of F, then both $F[\bot/A]$ 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 $F[\bot/A]$ and $F[\top/A]$, and
- "combine" them into a resolution proof for F.

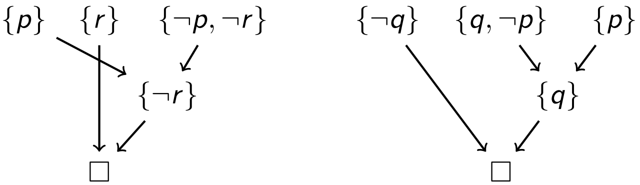
Inductive construction of resolution proofs

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

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

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

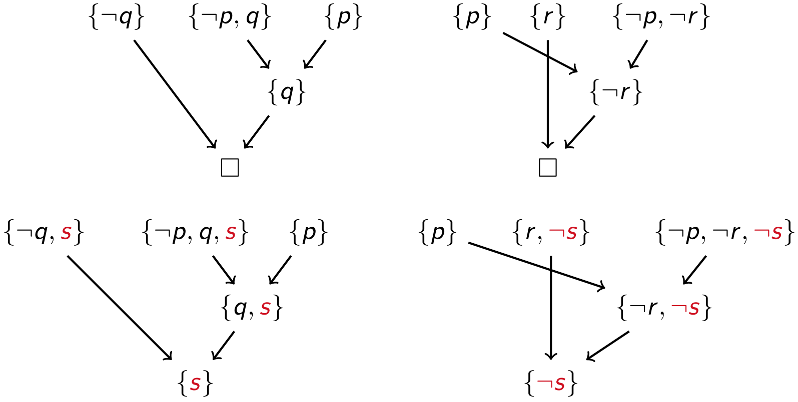
$$F[\bot/s] \equiv \{\ \{\neg q\}\ ,\ \{\neg p, q\}\ ,\ \{p\}\ \}$$



Inductive construction of resolution proofs

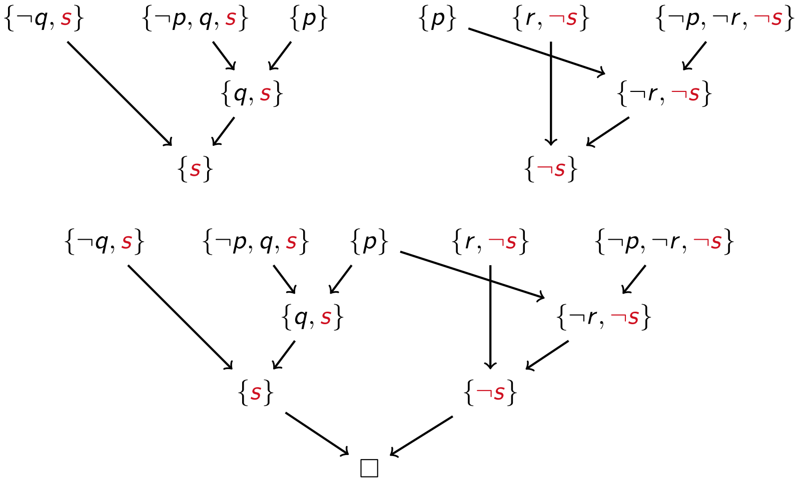
- Reintroduce s and $\neg s$.

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



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 $F \vdash_{Res} \Box$.

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

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

Step:

IH: For every unsat. set of clauses F with n dist. atoms, $F \vdash_{Res} \Box$.
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 $\neg A$.
$F[\bot/A] \equiv$ take F, remove all clauses with $\neg A$, remove all A.

Completeness proof

By IH: there are res. proofs $C_0, \dots, C_m = \Box$ from $F[\bot/A]$ and $D_0, \dots, D_n = \Box$ 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 $C'_m = \Box$ (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 $D'_n = \Box$ (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 $F \vdash_{Res} \Box$.

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 $F \cup \{\neg C\} \vdash_{Res} \Box$

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 $\Box$ is in the final F

Proof F_{init} is unsat. iff $F_{init} \vdash_{Res} \Box$ iff $\Box \in F_{final}$ 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 $F|_{\mathcal{A}} \ni \Box$, add new clause C to F by learning procedure.
   If $C = \Box$, 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 $F|_{\mathcal{A}} \ni \Box$, 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.

- $A_i \overset{C}{\mapsto} b_i$ 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

$$\begin{aligned} C_1 &= \{\neg A_1, \neg A_4, A_5\} \\ C_2 &= \{\neg A_1, A_6, \neg A_5\} \\ C_3 &= \{\neg A_1, \neg A_6, A_7\} \\ C_4 &= \{\neg A_1, \neg A_7, \neg A_5\} \\ C_5 &= \{A_1, A_4, A_6\} \end{aligned}$$

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 $(A_5 \overset{C_1}{\mapsto} 1, A_6 \overset{C_2}{\mapsto} 1, A_7 \overset{C_3}{\mapsto} 1)$.
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 $\mathcal{A} = (A_1 \mapsto b_1, \ldots, A_k \mapsto b_k)$ leads to conflict.
Find associated clauses $D_1, \ldots, D_{k+1}$ 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 $A_i \overset{C_i}{\mapsto} b_i$ 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:

$C_1 = \{\neg A_1, \neg A_4, A_5\}$	$D_8 := \{\neg A_1, \neg A_7, \neg A_5\}$	(clause C_4)
$C_2 = \{\neg A_1, A_6, \neg A_5\}$	$D_7 := \{\neg A_1, \neg A_5, \neg A_6\}$	(resolve D_8, C_3)
$C_3 = \{\neg A_1, \neg A_6, A_7\}$	$D_6 := \{\neg A_1, \neg A_5\}$	(resolve D_7, C_2)
$C_4 = \{\neg A_1, \neg A_7, \neg A_5\}$	$D_5 := \{\neg A_1, \neg A_4\}$	(resolve D_6, C_1)
$C_5 = \{A_1, A_4, A_6\}$	$D_4 := \{\neg A_1, \neg A_4\}$	
$A_1 \mapsto 1, A_2 \mapsto 0,$	$D_3 := \{\neg A_1, \neg A_4\}$	
$A_3 \mapsto 0, A_4 \mapsto 1,$	$D_2 := \{\neg A_1, \neg A_4\}$	
$A_5 \overset{C_1}{\mapsto} 1, A_6 \overset{C_2}{\mapsto} 1,$	$D_1 := \{\neg A_1, \neg A_4\}$	
$A_7 \overset{C_3}{\mapsto} 1$		

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 $A_i \overset{C_i}{\mapsto} b_i$, 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 $D_{k+1}, D_k, \cdots D_1 = C$ 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)

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

OLR:	$A_1 \mapsto 0$	$\{A_3, A_4\}$	$\{\neg A_2, \neg A_5\}$	$\{A_3, \neg A_4, A_5, \neg A_6\}$	$\{\neg A_2, \neg A_4, A_6\}$
DE:	$A_2 \mapsto 1$	$\{A_3, A_4\}$	$\{\neg A_5\}$	$\{A_3, \neg A_4, A_5, \neg A_6\}$	$\{\neg A_4, A_6\}$
OLR:	$A_5 \mapsto 0$	$\{A_3, A_4\}$		$\{A_3, \neg A_4, \neg A_6\}$	$\{\neg A_4, A_6\}$
DE:	$A_3 \mapsto 0$	$\{A_4\}$		$\{\neg A_4, \neg A_6\}$	$\{\neg A_4, A_6\}$
OLR:	$A_4 \mapsto 1$			$\{\neg A_6\}$	$\{A_6\}$
OLR:	$A_6 \mapsto 1$			$\{\ \}$	

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

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

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

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

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

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

$D_1 := \{\neg A_2, A_3\}$ (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

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

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

Intuitive reading: If $T_1, \ldots, T_n$ 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:

$$\dfrac{\dfrac{}{T_1} \quad \dfrac{\dfrac{}{U}}{T_2}}{S_1} \quad \dfrac{\dfrac{}{T_3}}{S_2}$$

:

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

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

$$\dfrac{T_1 \quad \ldots \quad T_n}{T}$$

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:

$\bot$, $\neg$, $\wedge$, $\vee$, $\rightarrow$ are primitives.

$\top$ abbreviates $\neg\bot$

A possible simplification:

$\neg F$ abbreviates $F \rightarrow \bot$

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:

$\Gamma \Rightarrow \Delta$ is provable (derivable)

iff

$\bigwedge \Gamma \models \bigvee \Delta \quad (\bigwedge \Gamma \rightarrow \bigvee \Delta$ valid)

Sequents: Notation

- We use set notation for multisets, e.g. $\{A, B \to C, A\}$
- 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

$$\frac{}{\bot, \Gamma \Rightarrow \Delta} \; \bot L \qquad \frac{}{A, \Gamma \Rightarrow A, \Delta} \; Ax$$

$$\frac{\Gamma \Rightarrow F, \Delta}{\neg F, \Gamma \Rightarrow \Delta} \; \neg L \qquad \frac{F, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \neg F, \Delta} \; \neg R$$

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

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

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

Sequent Calculus rules

Intuition: read backwards as proof search rules

$$\frac{}{\bot, \Gamma \Rightarrow \Delta}\ \bot L \qquad \frac{}{A, \Gamma \Rightarrow A, \Delta}\ Ax$$

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

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

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

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

Every rule decomposes its principal formula

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

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

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

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

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

$$\dfrac{\dfrac{}{(P \lor R) \land (Q \lor \neg R) \Rightarrow P \lor Q}\ \land L}{\Rightarrow (P \lor R) \land (Q \lor \neg R) \rightarrow P \lor Q}\ \rightarrow R$$

$$\dfrac{F, G, \Gamma \Rightarrow \Delta}{F \land G, \Gamma \Rightarrow \Delta} \quad \land L$$

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

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

$$\dfrac{\dfrac{\dfrac{}{P \vee R, Q \vee \neg R \Rightarrow P \vee Q}\ \vee R}{(P \vee R) \wedge (Q \vee \neg R) \Rightarrow P \vee Q}\ \wedge L}{\Rightarrow (P \vee R) \wedge (Q \vee \neg R) \to P \vee Q}\ \to R$$

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

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

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

$$\dfrac{\dfrac{\dfrac{\dfrac{}{P \lor R, Q \lor \neg R \Rightarrow P, Q}\ \lor L}{P \lor R, Q \lor \neg R \Rightarrow P \lor Q}\ \lor R}{(P \lor R) \land (Q \lor \neg R) \Rightarrow P \lor Q}\ \land L}{\Rightarrow (P \lor R) \land (Q \lor \neg R) \to P \lor Q}\ \to R$$

$$\dfrac{F, \Gamma \Rightarrow \Delta \quad G, \Gamma \Rightarrow \Delta}{F \lor G, \Gamma \Rightarrow \Delta}\ \lor L$$

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

$$
\dfrac{F, \Gamma \Rightarrow \Delta \qquad G, \Gamma \Rightarrow \Delta}{F \vee G, \Gamma \Rightarrow \Delta} \qquad \vee L
$$

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

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

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overline{P, Q \vee \neg R \Rightarrow P, Q}\;Ax \qquad \dfrac{}{R, Q \vee \neg R \Rightarrow P, Q}\;\vee L}{P \vee R, Q \vee \neg R \Rightarrow P, Q}\;\vee L}{P \vee R, Q \vee \neg R \Rightarrow P \vee Q}\;\vee R}{(P \vee R) \wedge (Q \vee \neg R) \Rightarrow P \vee Q}\;\wedge L}{\Rightarrow (P \vee R) \wedge (Q \vee \neg R) \to P \vee Q}\;\to R
$$

$$
\dfrac{F, \Gamma \Rightarrow \Delta \qquad G, \Gamma \Rightarrow \Delta}{F \vee G, \Gamma \Rightarrow \Delta}\;\vee L
$$

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

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

$$
\dfrac{\dfrac{}{P, Q \vee \neg R \Rightarrow P, Q}\,Ax \qquad \dfrac{\dfrac{}{R, Q \Rightarrow P, Q}\,Ax \qquad \dfrac{}{R, \neg R \Rightarrow P, Q}}{R, Q \vee \neg R \Rightarrow P, Q}\,\vee L}{\dfrac{\dfrac{\dfrac{P \vee R, Q \vee \neg R \Rightarrow P, Q}{P \vee R, Q \vee \neg R \Rightarrow P \vee Q}\,\vee R}{(P \vee R) \wedge (Q \vee \neg R) \Rightarrow P \vee Q}\,\wedge L}{\Rightarrow (P \vee R) \wedge (Q \vee \neg R) \to P \vee Q}\,\to R}\,\vee L
$$

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

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

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

$$
\dfrac{
  \dfrac{}{P, Q \lor \neg R \Rightarrow P, Q}\,Ax
  \qquad
  \dfrac{
    \dfrac{}{R, Q \Rightarrow P, Q}\,Ax
    \qquad
    \dfrac{\dfrac{}{R \Rightarrow R, P, Q}}{R, \neg R \Rightarrow P, Q}\,\neg L
  }{R, Q \lor \neg R \Rightarrow P, Q}\,\lor L
}{
  \dfrac{
    \dfrac{P \lor R, Q \lor \neg R \Rightarrow P, Q}{P \lor R, Q \lor \neg R \Rightarrow P \lor Q}\,\lor R
  }{
    \dfrac{(P \lor R) \land (Q \lor \neg R) \Rightarrow P \lor Q}{\Rightarrow (P \lor R) \land (Q \lor \neg R) \to P \lor Q}\,\to R
  }\,\land L
}\,\lor L
$$

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{P, Q \vee \neg R \Rightarrow P, Q}\;Ax \qquad \dfrac{\dfrac{}{R, Q \Rightarrow P, Q}\;Ax \qquad \dfrac{\dfrac{}{R \Rightarrow R, P, Q}\;Ax}{R, \neg R \Rightarrow P, Q}\;\neg L}{R, Q \vee \neg R \Rightarrow P, Q}\;\vee L}{P \vee R, Q \vee \neg R \Rightarrow P, Q}\;\vee L}{P \vee R, Q \vee \neg R \Rightarrow P \vee Q}\;\vee R}{(P \vee R) \wedge (Q \vee \neg R) \Rightarrow P \vee Q}\;\wedge L}{\Rightarrow (P \vee R) \wedge (Q \vee \neg R) \rightarrow P \vee Q}\;\rightarrow R
$$

$$
\dfrac{}{A, \Gamma \Rightarrow A, \Delta} \qquad Ax
$$

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 $\bigvee \emptyset = \bot$

We aim to prove: $\vdash_G S$ iff $\models |S|$

Lemma (Rule Equivalence)

For every rule $\dfrac{S_1 \quad \ldots \quad S_n}{S}$

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

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

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

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 the previous lemma.

Proof Search and Completeness

Proof search = growing a proof tree from the root

- 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

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

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

Example (failed proof)

$$\dfrac{\dfrac{\dfrac{}{P \Rightarrow P}\,Ax \quad Q \Rightarrow P}{P \vee Q \Rightarrow P}\,\vee L \quad \dfrac{P \Rightarrow Q \quad \dfrac{}{Q \Rightarrow Q}\,Ax}{P \vee Q \Rightarrow Q}\,\vee L}{P \vee Q \Rightarrow P \wedge Q}\,\wedge R$$

Falsifying assignments?

Proof search = Counterexample search

Can view sequent calculus as a search for a falsifying assignment for $|\Gamma \Rightarrow \Delta|$:

Make Γ true and Δ false

Some examples:

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

To make $F \land G$ true, make both F and G true

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

To make $F \land G$ false, make F or G false

Lemma (Search Equivalence)

At each stage of the search process,
if $S_1, \ldots, S_k$ are the unproved leaves, then $|S_0| \equiv |S_1| \wedge \ldots \wedge |S_k|$

Proof by induction on the number of search steps.
Initially trivially true (base case).
When applying a rule instance

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

we have

$$\begin{aligned} |S_0| \equiv & |S_1| \wedge \ldots \wedge |S_i| \wedge \ldots \wedge |S_k| \\ & \text{(by IH)} \\ \equiv & |S_1| \wedge \cdots \wedge |S_{i-1}| \wedge |U_1| \wedge \cdots \wedge |U_n| \wedge |S_{i+1}| \wedge \ldots \wedge |S_k| \\ & \text{(by Lemma Rule Equivalence)} \end{aligned}$$

Lemma

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

Proof If proof search fails, there is some unproved leaf

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

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

Theorem (Completeness)

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

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

Lemma

Proof search terminates.

Proof In every step, one logical operator is removed.
$\Rightarrow$ Size of sequent decreases by 1
$\Rightarrow$ Depth of proof tree is bounded by size of S_0
$\Rightarrow$ Construction of proof tree terminates. □
Observe: Breadth only bounded by $2^{\text{size of } S_0}$.

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:

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

An alternative formulation of the set version:

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

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 $\Rightarrow$

S.C.		*Tab.*		*Effect*
$\dfrac{F,\Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \neg F, \Delta}$	$\rightsquigarrow$	$\dfrac{+\neg F}{-F}$	$\rightsquigarrow$	$\begin{array}{c} +\neg F \\ \mid \\ -F \end{array}$
$\dfrac{\Gamma \Rightarrow F, G, \Delta}{\Gamma \Rightarrow F \vee G, \Delta}$	$\rightsquigarrow$	$\dfrac{+F \vee G}{\begin{array}{c} +F \\ +G \end{array}}$	$\rightsquigarrow$	$\begin{array}{c} +F \vee G \\ \mid \\ +F \\ \mid \\ +G \end{array}$
$\dfrac{\Gamma \Rightarrow F, \Delta \quad \Gamma \Rightarrow G, \Delta}{\Gamma \Rightarrow F \wedge G, \Delta}$	$\rightsquigarrow$	$\dfrac{+F \wedge G}{+F \mid +G}$	$\rightsquigarrow$	$\begin{array}{ccc} & +F \wedge G & \\ / & & \backslash \\ +F & & +G \end{array}$

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

$$
\begin{aligned}
&- A \to B \\
&- B \to C \\
&- A \\
&+ C
\end{aligned}
$$

$A \to B, B \to C, A \Rightarrow C$

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

$\Rightarrow$

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

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

1. Start with the tableau $-F_1, \ldots, +G_1, \ldots$.
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:

$$+\neg A \land \neg B$$
$$+A \lor B$$

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

Tableaux rules

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

$$\frac{-F \wedge G}{\begin{array}{c}-F\\-G\end{array}} \qquad \frac{+F \wedge G}{+F \mid +G}$$

$$\frac{-F \vee G}{-F \mid -G} \qquad \frac{+F \vee G}{\begin{array}{c}+F\\+G\end{array}}$$

$$\frac{-F \rightarrow G}{+F \mid -G} \qquad \frac{+F \rightarrow G}{\begin{array}{c}-F\\+G\end{array}}$$

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

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

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

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

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

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

Examples

$$\frac{A \quad B}{A \wedge B}\ \wedge I \qquad \frac{F \wedge G}{F}\ \wedge E_1 \qquad \frac{F \wedge G}{G}\ \wedge E_2$$

Main principles

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

Example

If from the local assumption F we can prove G
then we can prove $F \rightarrow G$.

The formal inference rule:

$$\dfrac{\begin{matrix}[F] \\ \vdots \\ G\end{matrix}}{F \rightarrow G} \rightarrow I$$

A proof tree:

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

Form the (open) assumption Q we can prove $P \rightarrow P \wedge Q$.
In symbols: $Q \vdash_N P \rightarrow P \wedge Q$

Growing the proof tree

Upwards:

Growing the proof tree

Upwards:

$$\dfrac{}{P \to P \wedge Q}$$

Growing the proof tree

Upwards:

$$\frac{}{P \to P \land Q} \to I$$

Growing the proof tree

Upwards:

$$\dfrac{\overline{P \wedge Q}}{P \to P \wedge Q} \to I$$

Growing the proof tree

Upwards:

$$\dfrac{\dfrac{}{P \wedge Q}\ \wedge I}{P \rightarrow P \wedge Q}\ \rightarrow I$$

Growing the proof tree

Upwards:

$$\dfrac{\dfrac{P \qquad Q}{P \wedge Q}\ \wedge I}{P \to P \wedge Q}\ \to I$$

Growing the proof tree

Upwards:

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

Growing the proof tree

Upwards:

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

Downwards:

Growing the proof tree

Upwards:

$$\dfrac{\dfrac{[P] \quad Q}{P \wedge Q}\ \wedge I}{P \to P \wedge Q}\ \to I$$

Downwards:

$$\dfrac{\dfrac{P \quad Q}{}}{}$$

Growing the proof tree

Upwards:

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

Downwards:

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

Growing the proof tree

Upwards:

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

Downwards:

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

Growing the proof tree

Upwards:

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

Downwards:

$$\dfrac{\dfrac{P \quad Q}{P \wedge Q}\ \wedge I}{\qquad}\ \rightarrow I$$

Growing the proof tree

Upwards:

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

Downwards:

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

Growing the proof tree

Upwards:

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

Downwards:

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

ND proof trees

The nodes of a ND proof tree are labeled by formulas.
Leaf nodes represent assumptions.
The root node is the conclusion.
Assumptions can be open or closed.
Closed assumptions are written [F].

Intuition:

- Open assumptions are used in the proof of the conclusion
- Closed assumptions are local assumptions in a subproof that have been closed (removed) by some proof rule like $\rightarrow I$.

ND proof trees are defined inductively.

- Every F is a ND proof tree (with open assumption F and conclusion F).
  Reading: From F we can prove F.
- New proof trees are constructed by the rules of ND.

Natural Deduction rules

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

$$\frac{\begin{array}{c}[F]\\ \vdots \\ G\end{array}}{F \to G} \to I \qquad\qquad \frac{F \to G \quad F}{G} \to E$$

$$\frac{F}{F \vee G} \vee I_1 \quad \frac{G}{F \vee G} \vee I_2 \qquad \frac{F \vee G \quad \begin{array}{c}[F]\\ \vdots \\ H\end{array} \quad \begin{array}{c}[G]\\ \vdots \\ H\end{array}}{H} \vee E$$

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

Natural Deduction rules

Rules for $\neg$ are special cases of rules for $\rightarrow$:

$$\frac{\begin{array}{c}[F] \\ \vdots \\ \bot\end{array}}{\neg F}\ \neg I \qquad \frac{\neg F \quad F}{\bot}\ \neg E$$

Natural Deduction rules

How to read a rule

$$\dfrac{\cdots \qquad \begin{array}{c}[F]\\ \vdots \\ G\end{array} \qquad \cdots}{\cdots}\; r$$

Forward:
Close all (or some) of the assumptions F in the proof of G when applying rule r

Backward:
In the subproof of G you can use the local assumption $[F]$.

Can use labels to show which rule application closed which assumptions.

Soundness

Definition

$\Gamma \vdash_N F$ if there is a proof tree with root F and open assumptions contained in the set of formulas Γ.

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 case: no rule, $F \in \Gamma$

Step: Case analysis of last rule

Case $\rightarrow E$:
IH: $\Gamma \models F \rightarrow G \qquad \Gamma \models F$
To show: $\Gamma \models G$
Assume $\mathcal{A} \models \Gamma \Rightarrow^{IH} \mathcal{A}(F \rightarrow G) = 1$ and $\mathcal{A}(F) = 1 \Rightarrow \mathcal{A}(G) = 1$

Soundness

Case

$$\dfrac{\begin{matrix}[F]\\ \vdots \\ G\end{matrix}}{F \rightarrow G} \rightarrow I$$

IH: $\Gamma, F \models G$
To show: $\Gamma \models F \rightarrow G$
iff for all $\mathcal{A}$, $\mathcal{A} \models \Gamma \Rightarrow \mathcal{A} \models F \rightarrow G$
iff for all $\mathcal{A}$, $\mathcal{A} \models \Gamma \Rightarrow (\mathcal{A} \models F \Rightarrow \mathcal{A} \models G)$
iff for all $\mathcal{A}$, $\mathcal{A} \models \Gamma$ and $\mathcal{A} \models F \;\Rightarrow\; \mathcal{A} \models G$
iff IH

Completeness

Towards completeness

ND can simulate truth tables

Lemma (Tertium non datur)

$\vdash_N F \vee \neg F$

Corollary (Cases)

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

Definition

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

Towards completeness

Lemma (1)

If $atoms(F) \subseteq \{A_1, \ldots, A_n\}$ *then* $A_1^{\mathcal{A}}, \ldots, A_n^{\mathcal{A}} \vdash_N F^{\mathcal{A}}$

Proof by induction on F

Lemma (2)

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

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

Completeness

Theorem (Completeness)

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

Proof

Relating
Sequent Calculs and Natural Deduction

Constructive approach to relating proof systems:

- Show how to transform proofs in one system into proofs in another system
- Implicit in inductive (meta)proof

Theorem (ND can simulate SC)

If $\vdash_G \Gamma \Rightarrow \Delta$ *then* $\Gamma, \neg\Delta \vdash_N \bot$ *(where* $\neg\{F_1, \ldots\} = \{\neg F_1, \ldots\}$*)*

Proof by induction on (the depth of) $\vdash_G \Gamma \Rightarrow \Delta$

Corollary (Completeness of ND)

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

Proof If $\Gamma \models F$ then $\Gamma_0 \models F$ for some finite $\Gamma_0 \subseteq \Gamma$.

Two completness proofs

- Direct
- By simulating a complete system

Theorem (SC can simulate ND)

If $\Gamma \vdash_N F$ *and* Γ *is finite then* $\vdash_G \Gamma \Rightarrow F$

Proof by induction on $\Gamma \vdash_N F$

Corollary

If $\Gamma \vdash_N F$ *then there is some finite* $\Gamma_0 \subseteq \Gamma$ *such that* $\vdash_G \Gamma_0 \Rightarrow F$

Hilbert Systems
Propositional Logic

(See the book by Troelstra and Schwichtenberg)

Easy to define, hard to use

No context management

A Hilbert system for propositional logic consists of

- a set of axioms (formulae)
- and a single inference rule, $\rightarrow E$ or modus ponens:

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

Proof trees for some Hilbert system are labeled with formulas.
The only inference rule is $\rightarrow E$.

Definition

We write $\Gamma \vdash_H F$ if there is a proof tree with root F whose leaves are either axioms or elements of Γ.

Alternative proof presentation

Proofs in Hilbert systems are frequently shown as lists of lines

1. F_1 *justification*$_1$
2. F_2 *justification*$_2$

$\vdots$

i. F_i *justification*$_i$

$\vdots$

justification$_i$ is either
assumption, *axiom* or $\rightarrow E(j, k)$ where $j, k < i$

Like linearized tree but also allows sharing of subproofs

Notational convention:

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

Note: $F \rightarrow G \rightarrow H \quad \equiv \quad F \land G \rightarrow H$

$F \rightarrow G \rightarrow H \quad \not\equiv \quad (F \rightarrow G) \rightarrow H$

Example (A simple Hilbert system)

Axioms:

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

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

Example (A simple Hilbert system)

Axioms:
$$F \to (G \to F) \quad \text{(A1)}$$
$$(F \to G \to H) \to (F \to G) \to F \to H \quad \text{(A2)}$$

A proof of $F \to F$:

Example (A simple Hilbert system)

Axioms:
$$F \to (G \to F) \quad \text{(A1)}$$
$$(F \to G \to H) \to (F \to G) \to F \to H \quad \text{(A2)}$$

A proof of $F \to F$:

Example (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 $F \rightarrow F$:

1 :

2 :

3 :

4 :

5 : $F \rightarrow F$

Example (A simple Hilbert system)

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

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

A proof of $F \to F$:

1 :

2 :

3 : $\_\_\_\_\_\_\_\_\_\_ \to (F \to F)$

4 : $\_\_\_\_\_\_\_\_\_\_$

5 : $F \to F$ $\to E : 3, 4$

Example (A simple Hilbert system)

Axioms:
$$F \to (G \to F) \quad \text{(A1)}$$
$$(F \to G \to H) \to (F \to G) \to F \to H \quad \text{(A2)}$$

A proof of $F \to F$:

1 :

2 :

3 : $(F \to F \to F) \to (F \to F)$

4 : $F \to F \to F$

5 : $F \to F$ $\quad\quad \to\!E : 3, 4$

Example (A simple Hilbert system)

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

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

A proof of $F \to F$:

1 :

2 :

3 : $(F \to F \to F) \to (F \to F)$

4 : $F \to F \to F$ $A1$

5 : $F \to F$ $\to E : 3, 4$

Example (A simple Hilbert system)

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

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

A proof of $F \to F$:

1 : ______________ $\to (F \to F \to F) \to F \to F$

2 : ______________

3 : $(F \to F \to F) \to (F \to F)$ $\quad\to E : 2, 1$

4 : $F \to F \to F$ $\quad A1$

5 : $F \to F$ $\quad\to E : 3, 4$

Example (A simple Hilbert system)

Axioms:

$$F \to (G \to F) \qquad \text{(A1)}$$

$$(F \to G \to H) \to (F \to G) \to F \to H \qquad \text{(A2)}$$

A proof of $F \to F$:

$$1 : (F \to (F \to F) \to F) \to (F \to F \to F) \to F \to F$$

$$2 : F \to (F \to F) \to F$$

$$3 : (F \to F \to F) \to (F \to F) \qquad {\to}E : 2, 1$$

$$4 : F \to F \to F \qquad A1$$

$$5 : F \to F \qquad {\to}E : 3, 4$$

Example (A simple Hilbert system)

Axioms:

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

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

A proof of $F \to F$:

$$1 : (F \to (F \to F) \to F) \to (F \to F \to F) \to F \to F$$

$$2 : F \to (F \to F) \to F \qquad A1$$

$$3 : (F \to F \to F) \to (F \to F) \qquad \to E : 2, 1$$

$$4 : F \to F \to F \qquad A1$$

$$5 : F \to F \qquad \to E : 3, 4$$

Example (A simple Hilbert system)

Axioms:

$$F \to (G \to F) \qquad \text{(A1)}$$

$$(F \to G \to H) \to (F \to G) \to F \to H \qquad \text{(A2)}$$

A proof of $F \to F$:

$$1 : (F \to (F \to F) \to F) \to (F \to F \to F) \to F \to F \qquad A2$$

$$2 : F \to (F \to F) \to F \qquad A1$$

$$3 : (F \to F \to F) \to (F \to F) \qquad {\to}E : 2, 1$$

$$4 : F \to F \to F \qquad A1$$

$$5 : F \to F \qquad {\to}E : 3, 4$$

Example (A simple Hilbert system)

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

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

A proof of $F \to F$:

$1:$	$(F \to (F \to F) \to F) \to (F \to F \to F) \to F \to F$	$A2$
$2:$	$F \to (F \to F) \to F$	$A1$
$3:$	$(F \to F \to F) \to (F \to F)$	$\to E : 2, 1$
$4:$	$F \to F \to F$	$A1$
$5:$	$F \to F$	$\to E : 3, 4$

$\Rightarrow \vdash_H F \to F$

Theorem (Deduction Theorem)

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

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

Proof "$\Leftarrow$":
$\Gamma \vdash_H F \rightarrow G$
$\Rightarrow \; F, \Gamma \vdash_H F \rightarrow G$
$\Rightarrow \; F, \Gamma \vdash_H G$ by $\rightarrow E$ because $F, \Gamma \vdash_H F$

Theorem (Deduction Theorem)

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

$$F, \Gamma \vdash_H G \quad \textit{iff} \quad \Gamma \vdash_H F \to G$$

Proof "$\Rightarrow$":

Theorem (Deduction Theorem)

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

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

Proof "$\Rightarrow$":
By induction on (the length/depth of) the proof of $F, \Gamma \vdash_H G$

Theorem (Deduction Theorem)

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

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

Proof "$\Rightarrow$":
By induction on (the length/depth of) the proof of $F, \Gamma \vdash_H G$
Then by cases on the last proof step:

Theorem (Deduction Theorem)

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

$$F, \Gamma \vdash_H G \quad \textit{iff} \quad \Gamma \vdash_H F \to G$$

Proof "$\Rightarrow$":
By induction on (the length/depth of) the proof of $F, \Gamma \vdash_H G$
Then by cases on the last proof step:

Case $G = F$:

Theorem (Deduction Theorem)

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

$$F, \Gamma \vdash_H G \quad \textit{iff} \quad \Gamma \vdash_H F \to G$$

Proof "$\Rightarrow$":
By induction on (the length/depth of) the proof of $F, \Gamma \vdash_H G$
Then by cases on the last proof step:

Case $G = F$: see proof of $F \to F$ from $A1$ and $A2$

Theorem (Deduction Theorem)

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

$$F, \Gamma \vdash_H G \quad \textit{iff} \quad \Gamma \vdash_H F \to G$$

Proof "$\Rightarrow$":
By induction on (the length/depth of) the proof of $F, \Gamma \vdash_H G$
Then by cases on the last proof step:

Case $G = F$: see proof of $F \to F$ from $A1$ and $A2$

Case $G \in \Gamma$ or axiom:

Theorem (Deduction Theorem)

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

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

Proof "$\Rightarrow$":
By induction on (the length/depth of) the proof of $F, \Gamma \vdash_H G$
Then by cases on the last proof step:

Case $G = F$: see proof of $F \rightarrow F$ from $A1$ and $A2$

Case $G \in \Gamma$ or axiom: by $A1$ and ...

Theorem (Deduction Theorem)

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

$$F, \Gamma \vdash_H G \quad \textit{iff} \quad \Gamma \vdash_H F \to G$$

Proof "$\Rightarrow$":
By induction on (the length/depth of) the proof of $F, \Gamma \vdash_H G$
Then by cases on the last proof step:

Case $G = F$: see proof of $F \to F$ from $A1$ and $A2$

Case $G \in \Gamma$ or axiom: by $A1$ and ...

Case $\to E$ from $H \to G$ and H:

Theorem (Deduction Theorem)

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

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

Proof "$\Rightarrow$":
By induction on (the length/depth of) the proof of $F, \Gamma \vdash_H G$
Then by cases on the last proof step:

Case $G = F$: see proof of $F \rightarrow F$ from $A1$ and $A2$

Case $G \in \Gamma$ or axiom: by $A1$ and ...

Case $\rightarrow E$ from $H \rightarrow G$ and H:

$$\frac{\qquad\qquad}{F \rightarrow G}$$

Theorem (Deduction Theorem)

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

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

Proof "$\Rightarrow$":
By induction on (the length/depth of) the proof of $F, \Gamma \vdash_H G$
Then by cases on the last proof step:

Case $G = F$: see proof of $F \rightarrow F$ from $A1$ and $A2$

Case $G \in \Gamma$ or axiom: by $A1$ and ...

Case $\rightarrow E$ from $H \rightarrow G$ and H:

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

Theorem (Deduction Theorem)

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

$$F, \Gamma \vdash_H G \quad \textit{iff} \quad \Gamma \vdash_H F \to G$$

Proof "$\Rightarrow$":
By induction on (the length/depth of) the proof of $F, \Gamma \vdash_H G$
Then by cases on the last proof step:

Case $G = F$: see proof of $F \to F$ from $A1$ and $A2$

Case $G \in \Gamma$ or axiom: by $A1$ and ...

Case $\to E$ from $H \to G$ and H:

$$\frac{F \to H}{F \to G}$$

Theorem (Deduction Theorem)

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

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

Proof "$\Rightarrow$":
By induction on (the length/depth of) the proof of $F, \Gamma \vdash_H G$
Then by cases on the last proof step:

Case $G = F$: see proof of $F \rightarrow F$ from $A1$ and $A2$

Case $G \in \Gamma$ or axiom: by $A1$ and ...

Case $\rightarrow E$ from $H \rightarrow G$ and H:

$$\dfrac{\dfrac{}{(F \rightarrow H) \rightarrow F \rightarrow G} \qquad F \rightarrow H}{F \rightarrow G}$$

Theorem (Deduction Theorem)

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

$$F, \Gamma \vdash_H G \quad \textit{iff} \quad \Gamma \vdash_H F \to G$$

Proof "$\Rightarrow$":
By induction on (the length/depth of) the proof of $F, \Gamma \vdash_H G$
Then by cases on the last proof step:

Case $G = F$: see proof of $F \to F$ from $A1$ and $A2$

Case $G \in \Gamma$ or axiom: by $A1$ and ...

Case $\to E$ from $H \to G$ and H:

$$\dfrac{\dfrac{F \to H \to G}{(F \to H) \to F \to G} \qquad F \to H}{F \to G}$$

Theorem (Deduction Theorem)

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

$$F, \Gamma \vdash_H G \quad \textit{iff} \quad \Gamma \vdash_H F \to G$$

Proof "$\Rightarrow$":
By induction on (the length/depth of) the proof of $F, \Gamma \vdash_H G$
Then by cases on the last proof step:

Case $G = F$: see proof of $F \to F$ from $A1$ and $A2$

Case $G \in \Gamma$ or axiom: by $A1$ and ...

Case $\to E$ from $H \to G$ and H:

$$\dfrac{\dfrac{(F \to H \to G) \to (F \to H) \to F \to G \quad F \to H \to G}{(F \to H) \to F \to G} \quad F \to H}{F \to G}$$

Hilbert System

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

$$
\begin{aligned}
&F \to G \to F && (A1)\\
&(F \to G \to H) \to (F \to G) \to F \to H && (A2)\\
&F \to G \to F \wedge G && (A3)\\
&F \wedge G \to F && (A4)\\
&F \wedge G \to G && (A5)\\
&F \to F \vee G && (A6)\\
&G \to F \vee G && (A7)\\
&F \vee G \to (F \to H) \to (G \to H) \to H && (A8)\\
&(\neg F \to \bot) \to F && (A9)
\end{aligned}
$$

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: $\vdash_N \quad\rightsquigarrow\quad \vdash_H + \to I \quad\rightsquigarrow\quad \vdash_H$

1. Transform a ND-proof tree into a proof tree containing Hilbert axioms, $\to E$ and $\to I$
   by replacing all other ND rules by Hilbert proofs incl. $\to I$
   Principle: ND rule $\rightsquigarrow$ 1 axiom + $\to I/E$

2. Eliminate the $\to I$ rules by the Deduction Theorem

Lemma (ND can simulate Hilbert)

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

Proof by induction on $\Gamma \vdash_H F$.

- 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 $i = 1, 2, 3 \ldots$.

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

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

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

Terms are inductively defined as follows:

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

Function symbols of arity 0 are called constant symbols.
Instead of $f_i^0()$ we write f_i^0.

Syntax of predicate logic: formulas

If P is a predicate symbol of arity k and $t_1, \ldots, t_k$ are terms then $P(t_1, \ldots, t_k)$ 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 $\forall x F$ and $\exists x F$:

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

Subformulas again correspond to subtrees.

Sructural induction of formulas

Like for propositional logic but

- Different base case: $\mathcal{P}(P(t_1, \ldots, t_k))$
- 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, ...	instead of	x_1, x_2, x_3, ...
a, b, c, ...	for	constant symbols
f, g, h, ...	for	function symbols of arity > 0
P, Q, R, ...	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 $\vee$ 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 $\exists x G$ or $\forall x G$; 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

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

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 $U_\mathcal{A}^k \to U_\mathcal{A}$,
- predicate symbols of arity k to functions of type $U_\mathcal{A}^k \to \{0, 1\}$ (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

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

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

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

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

$I_{\mathcal{A}}(g)$ 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 $\forall x\ P(x, f(x)) \land Q(g(a, z))$ 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:

$$
\begin{aligned}
\mathcal{A}(x) &= x^{\mathcal{A}} \\
\mathcal{A}(c) &= c^{\mathcal{A}} \\
\mathcal{A}(f(t_1, \ldots, t_k)) &= f^{\mathcal{A}}(\mathcal{A}(t_1), \ldots, \mathcal{A}(t_k))
\end{aligned}
$$

Example

$\mathcal{A}(f(g(a, z))) =$

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

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

$$\mathcal{A}(P(t_1, \dots, t_k)) = \begin{cases} 1 & \text{if } (\mathcal{A}(t_1), \dots, \mathcal{A}(t_k)) \in P^{\mathcal{A}} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathcal{A}(\forall x\ F) = \begin{cases} 1 & \text{if for every } d \in U_{\mathcal{A}},\ (\mathcal{A}[d/x])(F) = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\mathcal{A}(\exists x\ F) = \begin{cases} 1 & \text{if for some } d \in U_{\mathcal{A}},\ (\mathcal{A}[d/x])(F) = 1 \\ 0 & \text{otherwise} \end{cases}$$

$\mathcal{A}[d/x]$ coincides with $\mathcal{A}$ everywhere except that $x^{\mathcal{A}[d/x]} = d$.

Example

$\mathcal{A}(\forall x\ P(x, f(x)) \wedge Q(g(a, z))) =$

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
  $F' = (A_1 \vee \neg A_2 \wedge A_3)$.

  Exercise
  F is satifiable/valid iff F' is satisfiable/valid

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:

$$\mathcal{A}(=) \;=\; \{(d,d) \mid d \in U_{\mathcal{A}}\}$$

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
$\forall x\ P(a)$			
$\exists x\ (\neg P(x) \vee P(a))$			
$P(a) \to \exists x\ P(x)$			
$P(x) \to \exists x\ P(x)$			
$\forall x\ P(x) \to \exists x\ P(x)$			
$\forall x\ P(x) \wedge \neg\forall y\ P(y)$			

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
$\forall x \forall y\ F \ \equiv\ \forall y \forall x\ F$		
$\forall x \exists y\ F \ \equiv\ \exists x \forall y\ F$		
$\exists x \exists y\ F \ \equiv\ \exists y \exists x\ F$		
$\forall x\ F \vee \forall x\ G \ \equiv\ \forall x\ (F \vee G)$		
$\forall x\ F \wedge \forall x\ G \ \equiv\ \forall x\ (F \wedge G)$		
$\exists x\ F \vee \exists x\ G \ \equiv\ \exists x\ (F \vee G)$		
$\exists x\ F \wedge \exists x\ G \ \equiv\ \exists x\ (F \wedge G)$		

Equivalences

Theorem

1. $\neg\forall x F \equiv \exists x \neg F$
   $\neg\exists x F \equiv \forall x \neg F$
2. *If x does not occur free in G then:*
   $(\forall x F \wedge G) \equiv \forall x(F \wedge G)$
   $(\forall x F \vee G) \equiv \forall x(F \vee G)$
   $(\exists x F \wedge G) \equiv \exists x(F \wedge G)$
   $(\exists x F \vee G) \equiv \exists x(F \vee G)$
3. $(\forall x F \wedge \forall x G) \equiv \forall x(F \wedge G)$
   $(\exists x F \vee \exists x G) \equiv \exists x(F \vee G)$
4. $\forall x \forall y F \equiv \forall y \forall x F$
   $\exists x \exists y F \equiv \exists y \exists x F$

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:

$F_1 \to F_2$	abbreviates	$\neg F_1 \vee F_2$
$\top$	abbreviates	$P_1^0 \vee \neg P_1^0$
$\bot$	abbreviates	$P_1^0 \wedge \neg P_1^0$

Substitution

- Substitutions replace *free* variables by terms.
  (They 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* 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 subsitutions 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

Warning

If t contains a variable that is bound in F,
substitution may lead to variable capture:

$$(\forall x\ P(x, y))[f(x)/y] \ = \ \forall x\ P(x, f(x))$$

Variable capture should be avoided

Substitution lemmas

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 corresponding lemma on terms:

Lemma

$\mathcal{A}(u[t/x]) \;=\; (\mathcal{A}[\mathcal{A}(t)/x])(u)$

Proof by structural induction on u

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 $d \in U_{\mathcal{A}}$

Aim:

Transform any formula into an *equisatisfiable closed* formula

$$\forall x_1 \ldots \forall x_n \, G$$

where G is *quantifier-free*.

Rectified Formulas

Definition

A formula is rectified if no variable occurs both bound and free and if all quantifiers in the formula bind different variables.

Lemma

Let $F = QxG$ *be a formula where* $Q \in \{\forall, \exists\}$.
Let y *be a variable that does not occur in* G.
Then $F \equiv QyG[y/x]$.

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:

$$\begin{aligned}
\neg\forall x\ F &\equiv \exists x\ \neg F\\
\neg\exists x\ F &\equiv \forall x\ \neg F\\
Qx\ F \land G &\equiv Qx\ (F \land G)\\
F \land Qx\ G &\equiv Qx\ (F \land G)\\
Qx\ F \lor G &\equiv Qx\ (F \lor G)\\
F \lor Qx\ G &\equiv Qx\ (F \lor G)
\end{aligned}$$

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

$\exists x \forall y \exists z \forall u \exists v\, P(x, y, z, u, v)$

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 Every iteration produces an equisatisfiable formula.
Let (for simplicity) $F = \forall y \exists z\ G$ and $F' = \forall y\ G[f(y)/z]$.

1. $F' \models F$

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. $\mathcal{A}[u/y][v/z](G) = 1$

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

Skolem form

Theorem

A formula in RPF and its Skolem form are equisatisfiable.

Proof 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 $\mathcal{A}(F) = 1$.

Wlog $\mathcal{A}$ does not define f (because f is new)

$\Rightarrow$ for all $u \in U_{\mathcal{A}}$ there is a $v \in U_{\mathcal{A}}$ s.t. $\mathcal{A}[u/y][v/z](G) = 1 \quad (*)$

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 $u \in U_{\mathcal{A}}$:

$$\begin{aligned} & \mathcal{A}'[u/y](G[f(y)/z]) \\ = & \mathcal{A}'[u/y][f^{\mathcal{A}'}(u)/z](G) \\ = & \mathcal{A}'[u/y][v/z](G) \\ = & 1 \end{aligned}$$

Summary: conversion to Skolem form

Input: a formula F

Output: an equisatisfiable, rectified, closed formula
in Skolem form $\forall y_1 \dots \forall y_k\ G$ where G is quantifier-free

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.

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.

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 $a \in T(F)$
  where a is some arbitrary constant.
- For every n-ary function symbol f occurring in F,
  if $t_1, t_2, \ldots, t_n \in T(F)$ then $f(t_1, t_2, \ldots, t_n) \in T(F)$.

Note: All terms in $T(F)$ are variable-free by construction!

Example

$F = \forall x \forall y\ P(f(x), g(c, y))$

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 $t_1, \ldots, t_n \in T(F)$: $f^{\mathcal{A}}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$.

Fact
If $\mathcal{A}$ is a Herbrand structure, then $\mathcal{A}(t) = t$ for all $t \in U_{\mathcal{A}}$.

We call a Herbrand structure that is a model a Herbrand model.

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

Let F be a closed formula in Skolem form.
Then F is satisfiable iff it has a Herbrand model.

Proof If F has a Herbrand model then it is satisfiable.

For the other direction let $\mathcal{A}$ be an arbitrary model of F.
We define a Herbrand structure $\mathcal{T}$ as follows:

Universe	$U_{\mathcal{T}} = T(F)$
Function symbols	$f^{\mathcal{T}}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$
If F contains no constant:	$a^{\mathcal{A}} = u$ for some arbitrary $u \in U_{\mathcal{A}}$
Predicate symbols	$(t_1, \ldots, t_n) \in P^{\mathcal{T}}$ iff $(\mathcal{A}(t_1), \ldots, \mathcal{A}(t_n)) \in P^{\mathcal{A}}$

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 fun. and pred. 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.
Therefore $\mathcal{A}(G) = \mathcal{T}(G)$ (why?), and we are done.

Induction step: $G = \forall x\ H$.

$$
\begin{array}{lll}
 & \mathcal{A} \models G & \\
\Rightarrow & \text{for every } u \in U_{\mathcal{A}}\text{: } \mathcal{A}[u/x](H) = 1 & \\
\Rightarrow & \text{for every } u \in U_{\mathcal{A}} \text{ of the form } u = \mathcal{A}(t) & \\
 & \quad \text{where } t \in T(F)\text{: } \mathcal{A}[u/x](H) = 1 & \\
\Rightarrow & \text{for every } t \in T(F)\text{: } \mathcal{A}[\mathcal{A}(t)/x](H) = 1 & \\
\Rightarrow & \text{for every } t \in T(F)\text{: } \mathcal{A}(H[t/x]) = 1 & \text{(substitution lemma)} \\
\Rightarrow & \text{for every } t \in T(F)\text{: } \mathcal{T}(H[t/x]) = 1 & \text{(induction hypothesis)} \\
\Rightarrow & \text{for every } t \in T(F)\text{: } \mathcal{T}[\mathcal{T}(t)/x](H) = 1 & \text{(substitution lemma)} \\
\Rightarrow & \text{for every } t \in T(F)\text{: } \mathcal{T}[t/x](H) = 1 & (\mathcal{T} \text{ is Herbrand structure}) \\
\Rightarrow & \mathcal{T}(\forall x\ H) = 1 & (U_{\mathcal{T}} = T(F)) \\
\Rightarrow & \mathcal{T} \models G &
\end{array}
$$

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 $F = \forall y_1 \ldots \forall y_n F^*$ be a closed formula in Skolem form.
The Herbrand expansion of F is the set of formulas

$$E(F) = \{F^*[t_1/y_1] \ldots [t_n/y_n] \mid t_1, \ldots, t_n \in T(F)\}$$

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

$E(\forall x \forall y\ P(f(x), g(c, y)) =$

Note The Herbrand expansion can be viewed as a set of propositional formulas.

Gödel-Herbrand-Skolem Theorem

Theorem

Let F be a closed formula in Skolem form.
Then F 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:
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 $E(F)$

Herbrand's Theorem

Theorem

Let F be a closed formula in Skolem form.
F is unsatisfiable iff some finite subset of $E(F)$ is unsatisfiable.

Proof Follows immediately from the Gödel-Herbrand-Skolem Theorem and the Compactness Theorem.

Gilmore's Algorithm

Let F be a closed formula in Skolem form
and let $F_1, F_2, F_3, \ldots$ be a computable enumeration of $E(F)$.

Input: F

$n := 0$;
repeat $n := n + 1$;
until $(F_1 \wedge F_2 \wedge \ldots \wedge F_n)$ 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 $x_1, \dots, x_n$. 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 an equisatisfiable, closed formula in Skolem form as produced by the Normal Form transformations starting with F.

Fact: Every model of G is a model of F. (Check this!)

$$\begin{array}{lll} F_0 \text{ satisfiable} & \Rightarrow & F \text{ satisfiable} \\ & \Rightarrow & G \text{ satisfiable} \\ & \Rightarrow & G \text{ has a Herbrand model } \mathcal{T} \\ & \Rightarrow & F \text{ also has that model } \mathcal{T} \\ & \Rightarrow & F_0 \text{ has a countable model} \\ & & \text{(Herbrand universes are countable)} \end{array}$$

Löwenheim-Skolem Theorem

Formulas of first-order logic cannot enforce uncountable models

Formulas of first-order logic cannot axiomatize the real numbers
because there will always be countable models

First-Order Logic
Resolution

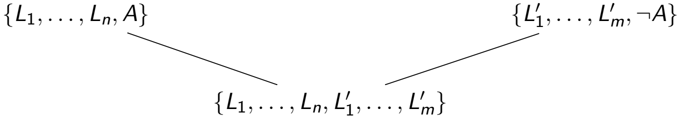
Resolution for first-order logic

Gilmore's algorithm is correct and complete,
but useless in practice.

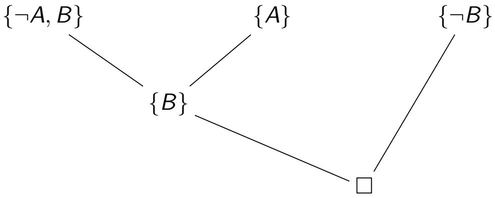
We upgrade resolution to make it work for predicate logic.

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 $F_1, F_2, F_3, \ldots$ be an enumeration of $E(F)$.

$n := 0$;
repeat $n := n + 1$
until $(F_1 \wedge F_2 \wedge \ldots \wedge F_n)$ 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 $F = \forall y_1 \ldots \forall y_n\, F^*$ be a closed formula in Skolem form with F^* in CNF, and let $C_1, \ldots, C_m$ be the clauses of F^*.

The clause Herbrand expansion of F is the set of ground clauses

$$CE(F) = \bigcup_{i=1}^{m} \{\, C_i[t_1/y_1] \ldots [t_n/y_n] \mid t_1, \ldots, t_n \in T(F)\}$$

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 $C_1, C_2, C_3, \ldots$ be an enumeration of $CE(F)$.

$n := 0$;
$S := \emptyset$;
repeat
  $n := n + 1$;
  $S := S \cup \{C_n\}$;
until $S \vdash_{Res} \Box$

return "unsatisfiable"

Note: The search for $\Box$ can be performed incrementally every time S is extended.

Example

$F^* = \{\{\neg P(x), \neg P(f(a)), Q(y)\},\ \{P(y)\},\ \{\neg P(g(b,x)), \neg Q(b)\}\}$

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* $C_1, \dots, C_m = \Box$ *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* $b < i$

Where do the ground substitutions come from?

Better:

- allow substitutions with variables
- only instantiate clauses enough to allow one (new kind of) resolution step

Example

Resolve $\{P(x), Q(x)\}$ and $\{\neg P(f(y)), R(y)\}$

Substitutions as functions

Substitutions are functions from variables to terms:
$[t/x]$ maps x to t (and all other variales 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

$$(P(x,y))[f(y)/x][b/y] \;=\; (P(f(y),y))[b/y] \;=\; P(f(b),b)$$

Similarly we can compose arbitrary substitutions σ_1 and σ_2:
$\sigma_1\sigma_2$ is the substitution that applies σ_1 first and then σ_2.

Substitutions are functions. Therefore

$$\sigma_1 = \sigma_2 \quad \text{iff} \quad \text{for all variables } x,\ x\sigma_1 = x\sigma_2$$

Substitutions as functions

Definition

The domain of a substitution: $dom(\sigma) = \{x \mid x\sigma \neq x\}$

Example

$dom([a/x][b/y]) = \{x, y\}$

Substitutions are defined to have finite domain.
Therefore every substitution can be written as a simultaneous substitution $[t_1/x_1, \ldots, t_n/x_n]$.

Unifier and most general unifier

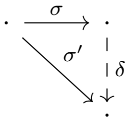
Let $\mathbf{L} = \{L_1, \ldots, L_k\}$ be a set of literals.

A substitution σ is a unifier of $\mathbf{L}$ if

$$L_1\sigma = L_2\sigma = \cdots = L_k\sigma$$

i.e. if $|\mathbf{L}\sigma| = 1$, where $\mathbf{L}\sigma = \{L_1\sigma, \ldots, L_k\sigma\}$.

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

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 **L** *is non-unifiable then the algorithm returns "non-unifiable".*

Proof If **L** is non-unifiable then the algorithm can never exit the loop normally.

Correctness/completeness of the unification algorithm

Lemma

If **L** *is unifiable then the algorithm returns the mgu of* **L** *(and so in particular every unifiable set* **L** *has an mgu).*

Proof Assume **L** is unifiable and let n be the number of iterations of the loop on input **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 **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 $[t_1/x_1, \ldots, t_k/x_k, u/x]$ where $x_1, \ldots, x_k, x$ are distinct. Define $\delta_{i+1} = [t_1/x_1, \ldots, t_k/x_k]$.
Note $u = x\delta_i = t\delta_i = t\delta_{i+1}$ ($\sigma_i\delta_i$ is unifier (IH), x not in t)

$$
\begin{array}{lll}
 & \sigma_{i+1}\,\delta_{i+1} & \\
= & \sigma_i\,[t/x]\,\delta_{i+1} & \text{(algorithm extends } \sigma_i \text{ with } [t/x]) \\
= & \sigma_i\,[t_1/x_1, \ldots, t_k/x_k, t\delta_{i+1}/x] & \\
= & \sigma_i\,[t_1/x_1, \ldots, t_k/x_k, u/x] & \text{(Note } u = t\delta_{i+1}) \\
= & \sigma_i\,\delta_i & \\
= & \sigma' & \text{(IH)}
\end{array}
$$

The standard view of unification

A unification problem is a pair of terms $s =^? t$
(or a set of pairs $\{s_1 =^? t_1, \ldots, s_n =^? t_n\}$)

A unifier is a substitution σ such that $s\sigma = t\sigma$
(or $s_1\sigma = t_1\sigma, \ldots, s_n\sigma = t_n\sigma$)

Renaming

Definition

A substitution ρ is a renaming if for every variable x, $x\rho$ is a variable and ρ is injective on $dom(\rho)$.

Resolvents for first-order logic

A clause R is a resolvent of two clauses C_1 and C_2 if the following holds:

- 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 $L_1, \ldots, L_m \in C_1$ $(m \geq 1)$
  and literals $L'_1, \ldots, L'_n \in C_2\,\rho$ $(n \geq 1)$ such that

$$\mathbf{L} = \{\overline{L_1}, \ldots, \overline{L_m}, L'_1, \ldots, L'_n\}$$

  is unifiable. Let σ be an mgu of $\mathbf{L}$.
- $R = ((C_1 - \{L_1, \ldots, L_m\}) \cup (C_2\,\rho - \{L'_1, \ldots, L'_n\}))\sigma$

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 $C_1, \ldots, C_m = C$ 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 $F \vdash_{Res} \Box$ imply that F is unsatisfiable?

Completeness Does unsatisfiability of F imply $F \vdash_{Res} \Box$?

Exercise

Derive $\Box$ 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 $x_1, \ldots, x_n$:

$$\forall H \ = \ \forall x_1 \forall x_2 \ldots \forall x_n H$$

Theorem

Let F be a closed formula in Skolem form with matrix F^* in CNF. If $F^* \vdash_{Res} \Box$ then F is unsatisfiable.

Theorem

Let F be a closed formula in Skolem form with matrix F^* in CNF.
If $F^* \vdash_{Res} \Box$ then F is unsatisfiable.

Proof Let $C_1, \dots, C_m$ be the sequence of clauses leading to $\Box$.
By induction on i: if $\forall F^* \models \forall C_i$. Trivial if $C_i \in F^*$.
Let C_i be a resolvent of C_a and C_b $(a, b < i)$. We prove

$$\forall C_a, \forall C_b \models \forall C_i \qquad (*)$$

Thus $\forall F^* \models \forall C_i$ because $\forall F^* \models \forall C_a$ and $\forall F^* \models \forall C_b$ by IH.

Proof of $(*)$: Assume $\mathcal{A}(\forall C_a) = \mathcal{A}(\forall C_b) = 1 \qquad (**)$

$$\begin{aligned} C_i &= ((C_a - \{L_1, \dots\}) \cup (C_b\rho - \{L'_1, \dots\}))\sigma \\ &= (C_a\sigma - \{L\}) \cup (C_b\rho\sigma - \{\overline{L}\}) \end{aligned}$$

Indirect proof of $\mathcal{A}(\forall C_i) = 1$. Assume $\mathcal{A}(\forall C_i) = 0$.
$\Rightarrow \mathcal{A}'(C_i) = 0$ where $\mathcal{A}' = \mathcal{A}[u_1/x_1, \dots]$ for some $u_i \in U_{\mathcal{A}}$
$\Rightarrow \mathcal{A}'(C_a\sigma - \{L\}) = \mathcal{A}'(C_b\rho\sigma - \{\overline{L}\}) = 0$
$\Rightarrow \mathcal{A}'(L) = \mathcal{A}'(\overline{L}) = 1$ becs. $\mathcal{A}'(C_a\sigma) = \mathcal{A}'(C_b\rho\sigma) = 1$ becs. $(**)$
Contradiction

Completeness: The idea

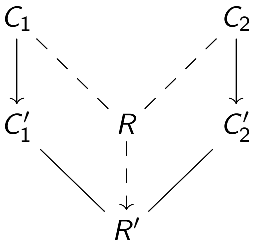
Simulate ground resolution because that is complete

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'.

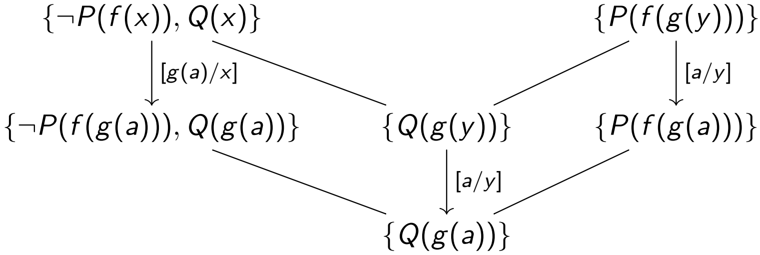
Then there is a resolvent R of C_1, C_2 such that R' is a ground instance of R.



$\rightarrow$: 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

(3) 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 $C_2\rho$. Thus 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) $\Rightarrow$ $R' = (C_1' - \{L\}) \cup (C_2' - \{\overline{L}\})$ where $L \in C_1'$ and $\overline{L} \in C_2'$

$\Rightarrow$ there are $\{L_1, \dots\} \subseteq C_1$ and $\{L_1', \dots\} \subseteq C_2\rho$

s.t. σ is a unifier of $\{\overline{L_1}, \dots, L_1', \dots\} =: M$.

Let σ_0 be an mgu of M and let $\sigma = \sigma_0\delta$ for some δ

$\Rightarrow$ A resolvent of C_1 and C_2:

$R := ((C_1 - \{L_1, \dots\}) \cup (C_2\rho - \{L_1', \dots\}))\sigma_0$

$R\delta = ((C_1 - \{L_1, \dots\}) \cup (C_2\rho - \{L_1', \dots\}))\sigma$

$= (C_1\sigma - \{L\}) \cup (C_2\rho\sigma - \{\overline{L}\})$

$= (C_1' - \{L\}) \cup (C_2' - \{\overline{L}\})$

$= R'$

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 $F^* \vdash_{Res} \Box$.

Proof If F is unsatisfiable, there is a ground resolution proof $C'_1, \ldots, C'_n = \Box$. We transform this step by step into a resolution proof $C_1, \ldots, C_n = \Box$ 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 $F^* \vdash_{Res} \Box$.

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 $\Box \notin S$ 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:*
every resolvent is added eventually

Three possible behaviours:

- The algorithm terminates and $\Box \in S$
  $\Rightarrow F$ is unsatisfiable
- The algorithm terminates and $\Box \notin S$
  $\Rightarrow F$ 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. We assume w.l.o.g. that $F = \forall x_1 \ldots \forall x_n F^*$ and the variables occurring in F^* are exactly $x_1, \ldots, x_n$. (If necessary bring F into closed Skolem form).
We consider two cases:
$n = 0$. Exercise.
$n > 0$. Let $G = \forall x_1 \ldots \forall x_n F^*[f(x_1)/x_1]$, where f is a function symbol that does not occur in F^*. G is satisfiable (why?).
If G has a model M with universe U, then F has a model with universe $\{f^M(u) \mid u \in (U)\}$. Because G has a Herbrand model with countably infinite universe $T(G)$ (by the Fundamental Theorem), F also has a model with countably infinite universe $\{f(t) \mid t \in T(G)\}$.

Modelling equality

Let F be a formula of predicate logic with equality.
Let Eq be a predicate symbol that does not occur in F.
Let E_F be 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 in F.

Quotient structure

Definition

Let $\mathcal{A}$ be a structure and $\sim$ an equivalence relation on $U_{\mathcal{A}}$ that is a congruence relation 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

$\mathcal{A}/_{\sim}(t) = [\mathcal{A}(t)]_{\sim}$

Lemma

$\mathcal{A}/_{\sim}(F) = \mathcal{A}(F)$

Theorem

The formulas F and $E_F \wedge F[Eq/=]$ are equisatisfiable.

Proof We show that if $E_F \wedge F[Eq/=]$ is sat., then F is satisfiable.
Assume $\mathcal{A} \models E_F \wedge F[Eq/=]$.
$\Rightarrow Eq^{\mathcal{A}}$ is an congruence relation.

Let $\mathcal{B} = \mathcal{A}/_{Eq^{\mathcal{A}}}$ (extended with $=$ interpreted as identity).
$\Rightarrow \mathcal{B} \models F[Eq/=]$

By construction $Eq^{\mathcal{B}}$ is identity:
$Eq^{\mathcal{B}}([a], [a']) = Eq^{\mathcal{A}}(a, a') = ([a]_{Eq^{\mathcal{A}}} = [a']_{Eq^{\mathcal{A}}})$
$\Rightarrow \mathcal{B}(F[Eq/=]) = \mathcal{B}(F)$
$\Rightarrow \mathcal{B} \models F$

Conversely, it is easy to see that any model of F can be turned into 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 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 $I_1, \ldots, I_t$.
The instructions manipulate registers R_i $(i = 1, 2, \ldots)$
that contain (unbounded!) natural numbers.
There are 4 instructions:

$$\begin{array}{l} R_n := 0 \\ R_n := R_n + 1 \\ R_n := R_m \\ \text{IF } R_m = R_n \text{ GOTO } p \end{array}$$

Assumption: all jumps in a program go to $1, \ldots, t+1$;
execution terminates when the PC is $t+1$.

Let r be the maximal index of any register used in a program P.
Then the state of P during execution can be described by a tuple of natural numbers

$$(n_1, \ldots, n_r, k)$$

where n_i is the contents of R_i and k is the PC (the number of the next instruction to be executed).

Undecidability

Theorem (Undecidability of the halting problem for RMPs)
It is undecidable if a given register machine program terminates when started in state $(0, \ldots, 0, 1)$.

We reduce the halting problem for RMPs to the validity problem for first-order formulas.

Notation:
$P(0)\downarrow$ = "RMP P started in state $(0, \ldots, 0, 1)$ 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 $P = I_1, \ldots, I_t$.
Funct. symb.: z, s. Abbr.: $\overline{0} = z$, $\overline{1} = s(z)$, $\overline{2} = s(s(z))$, ...
Pred. symb.: R (arity: $r+1$) "reachable"
Aim: if $R(\overline{n_1}, \ldots \overline{n_r}, \overline{k})$ then $(0, \ldots, 0, 1) \overset{P}{\rightsquigarrow} (n_1, \ldots, n_r, k)$
For every I_i construct closed formula Ψ_i:

$I_i = (R_n := 0)$: $\Psi_i := \forall x_1 \ldots x_r \ (R(x_1, \ldots, x_n, \ldots, x_r, \bar{i}) \rightarrow R(x_1, \ldots, z, \ldots, x_r, s(\bar{i}))$

$I_i = (R_n := R_n + 1)$: the same except $s(x_n)$ instead of z

$I_i = (R_n := R_m)$: the same except x_m instead of z

$I_i = (IF\ R_m = R_n\ GOTO\, p)$:
$\Psi_i := \forall x_1 \ldots x_r (R(x_1, \ldots, x_r, \bar{i}) \rightarrow (x_m = x_n \rightarrow R(x_1, \ldots, x_r, \overline{p})) \wedge (x_m \neq x_n \rightarrow R(x_1, \ldots, x_r, s(\bar{i})))$

$\Psi_P := \Psi \wedge R(z, \ldots, z, s(z)) \wedge \Psi_1 \wedge \cdots \wedge \Psi_t$

Ψ enforces that every model is similar to $\mathbb{N}$:
$\Psi := \forall x \forall y (s(x) = s(y) \rightarrow x = y) \wedge \forall x (z \neq s(x))$
(How can models of Ψ differ from $\mathbb{N}$?)

$\varphi_P := \Psi_P \rightarrow \tau$ where $\tau := \exists x_1 \dots x_r\, R(x_1, \dots, x_r, s(\bar{t}))$

Claim: $P(0)\downarrow$ iff $\models \varphi_P$

"$\Rightarrow$": Assume $P(0)\downarrow$, show $\models \varphi_P$. Assume $\mathcal{A} \models \Psi_P$.

Lemma

If $(0, \dots, 0, 1) \overset{P}{\rightsquigarrow} (n_1, \dots, n_r, k)$ *then* $\mathcal{A} \models R(\overline{n_1}, \dots, \overline{n_r}, \overline{k})$

Proof by induction on the length of the execution using $\mathcal{A} \models \Psi_P$.

Thus $\mathcal{A} \models \tau$ because $P(0)\downarrow$.

"$\Leftarrow$": $\models \varphi_P \;\Rightarrow\; \mathcal{N} \models \varphi_P \;\Rightarrow\; (\mathcal{N} \models \Psi_P \Rightarrow \mathcal{N} \models \tau) \;\Rightarrow\; P(0)\downarrow$
where $U_{\mathcal{N}} := \mathbb{N}$, $z^{\mathcal{N}} := 0$ $s^{\mathcal{N}}(n) := n + 1$,
$R^{\mathcal{N}} := \{s \mid (0, \dots, 0, 1) \overset{P}{\rightsquigarrow} s\}$

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

$$\exists x_1 \ldots \exists x_m \forall y_1 \ldots \forall y_n \; F$$

where F is quantifier-free and contains no function symbols of arity > 0.

This is also called the Bernays-Schönfinkel class.

Corollary

Unsatisfiability is decidable for formulas in the $\exists^*\forall^*$ *class.*

What if a formula is not in the $\exists^*\forall^*$ class?
Try to transform it into the $\exists^*\forall^*$ class!

Example

$\forall y \exists x\, (P(x) \wedge Q(y))$

Heuristic transformation procedure:

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 $F_1 \vee \cdots \vee F_n$, $n \geq 2$:
  $\exists x\ F \ \equiv\ \exists x\ (F_1 \vee \cdots \vee F_n)$

Together with the dual transformations for $\forall$

Example

$\exists x\ (P(x) \wedge \exists y\ (Q(y) \vee R(x)))$

Warning: Complexity!

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. Sokrates is a man. Sokrates is mortal.

The monadic class is decidable

Theorem

Satisfiability of monadic formulas is decidable.

Proof Put into NNF. Perform miniscoping.
The result has no nested quantifiers (Exercise!).
First pull out all $\exists$, then all $\forall$.
Existentially quantify free variables.
The result is in the $\exists^*\forall^*$ class.

Corollary

Validity 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 formula has the finite model property, satisfiability is decidable.

Theorem

Monadic formulas have the finite model property.

The finite model property

Theorem

Monadic formulas have the finite model property.

Proof A satisfiable monadic formula F
with k different monadic predicate symbols $P_1, \ldots, P_k$
has a model of size $\leq 2^k$.
Given a model $\mathcal{A}$ of F, define $\sim$ such that $|U_{\mathcal{A}/\sim}| \leq 2^k$:

$u \sim v$ iff for all i, $P_i^{\mathcal{A}}(u) = P_i^{\mathcal{A}}(v)$

Why $|U_{\mathcal{A}/\sim}| \leq 2^k$?
Every class $[u]_\sim$ can be viewed as a bit-vector of length k:
$(P_1^{\mathcal{A}}(u), \ldots, P_k^{\mathcal{A}}(u))$

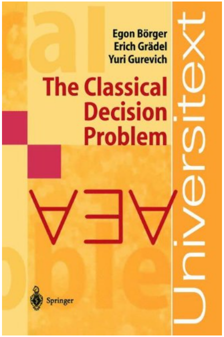
Obvious: $\sim$ is an equivalence.

$\sim$ is a congruence: if $u \sim v$ then $P_i^{\mathcal{A}}(u) = P_i^{\mathcal{A}}(u)$ for all i

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 identifty formulas that differ only in the names of bound variables.

Example

$\forall x \exists y\, P(x, y) \;=\; \forall u \exists v\, P(u, v)$

The renaming must not capture free variables:

$\forall x\, P(x, y) \;\neq\; \forall y\, P(y, y)$

Substitution $F[t/x]$ assumes that bound variables in F are automatically renamed to avoid capturing free variables in t.

Example

$(\forall x\, P(x, y))[f(x)/y] \;= \forall x'\, P(x', f(x))$

All proof systems below are extensions
of the corresponding propositional systems

Sequent Calculus

Sequent Calculus rules

We add the following rules to those for propositional logic:

$$\frac{F[t/x], \forall x\, F, \Gamma \Rightarrow \Delta}{\forall x\, F, \Gamma \Rightarrow \Delta}\ \forall L \qquad \frac{\Gamma \Rightarrow F[y/x], \Delta}{\Gamma \Rightarrow \forall x\, F, \Delta}\ \forall R(*)$$

$$\frac{F[y/x], \Gamma \Rightarrow \Delta}{\exists x\, F, \Gamma \Rightarrow \Delta}\ \exists L(*) \qquad \frac{\Gamma \Rightarrow F[t/x], \exists x\, F, \Delta}{\Gamma \Rightarrow \exists x\, F, \Delta}\ \exists R$$

$(*)$: y not free in the conclusion of the rule

Note: $\forall L$ and $\exists R$ do not delete the principal formula

Soundness

Lemma

For every quantifier rule $\dfrac{S'}{S}$, $|S|$ *and* $|S'|$ *are equivalid.*

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
$(|S| \equiv |S_1| \wedge \ldots \wedge |S_n|)$.

Completeness Proof

Construct counter model
from (possibly infinite!) failed proof search

Let $e_0, e_1, \ldots$ be an enumeration of all terms
(over some given set of function symbols and variables)

Proof search

Construct proof tree incrementally:

1. Pick some uproved 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
A ends in a sequent where no rule is applicable
or A is infinite.

Construction of Herbrand countermodel $\mathcal{A}$ from A

$U_{\mathcal{A}}$ = all terms over the function symbols and variables in A

$f^{\mathcal{A}}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$

$P^{\mathcal{A}} = \{(t_1, \ldots, t_n) \mid P(t_1, \ldots, t_n) \in \Gamma$ for some $\Gamma \Rightarrow \Delta \in A\}$

Theorem

For all $\Gamma \Rightarrow \Delta \in A$, *for all* $F \in \Gamma \cup \Delta$: $\mathcal{A}(F) = \begin{cases} 1 & \textit{if } F \in \Gamma \\ 0 & \textit{if } F \in \Delta \end{cases}$

Proof by induction on the structure of F

$F = P(t_1, \ldots, t_n)$:
$F \in \Gamma \Rightarrow \mathcal{A}(F) = 1$ by def
$F \in \Delta \Rightarrow F \notin$ any $\Gamma \in A$, (A would end in Ax) $\Rightarrow \mathcal{A}(F) = 0$

F not atomic $\Rightarrow F$ must be p.f. in some $\Gamma \Rightarrow \Delta \in A$ (fairness!)
Let $\Gamma' \Rightarrow \Delta'$ be the next sequent in A

$F = \neg G$: $F \in \Gamma$ iff $G \in \Delta'$ iff $\mathcal{A}(G) = 0$ (IH) iff $\mathcal{A}(F) = 1$

$F = G_1 \wedge G_2$:
$F \in \Gamma \Rightarrow G_1, G_2 \in \Gamma' \Rightarrow A(G_1) = \mathcal{A}(G_2) = 1$ (IH) $\Rightarrow \mathcal{A}(F) = 1$
$F \in \Delta \Rightarrow G_1 \in \Delta'$ or $G_2 \in \Delta' \Rightarrow \mathcal{A}(G_1) = 0$ or $\mathcal{A}(G_2) = 0$ (IH)
$\Rightarrow \mathcal{A}(F) = 0$

$F = \forall x\, G$: $F \in \Delta \Rightarrow G[y/x] \in \Delta' \Rightarrow \mathcal{A}(G[y/x]) = 0$ (IH)
$\Rightarrow \mathcal{A}[\mathcal{A}(y)/x](G) = 0 \Rightarrow \mathcal{A}(F) = 0$

Completeness

Corollary

If proof search with root $\Gamma \Rightarrow \Delta$ *fails,*
then there is a structure $\mathcal{A}$ *such that* $\mathcal{A}(\bigwedge \Gamma \to \bigvee \Delta) = 0$*.*

Example

$\exists x\, P(x) \Rightarrow \forall x\, P(x)$

Corollary (Completeness)

If $\models |\Gamma \to \Delta|$ *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 $\mathcal{A}(\bigwedge \Gamma \to \bigvee \Delta) = 0$.
Therefore not $\models |\Gamma \to \Delta|$.

Natural Deduction

Natural Deduction rules

$$\frac{F[y/x]}{\forall x\, F}\ \forall I(*) \qquad \frac{\forall x\, F}{F[t/x]}\ \forall E$$

$$\frac{F[t/x]}{\exists x\, F}\ \exists I \qquad \frac{\exists x\, F \qquad \begin{array}{c}[F[y/x]]\\ \vdots \\ H\end{array}}{H}\ \exists E(**)$$

$(*)$: $(y = x$ 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 $F[y/x]$

Theorem (Soundness)

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

Proof as before, with additional cases:

$$\dfrac{\exists x\, F \qquad \begin{array}{c}[F[y/x]]\\ \vdots \\ H\end{array}}{H}\ \exists E(**) \qquad \text{IH: } \Gamma \models \exists x F \text{ and } F[y/x], \Gamma \models H$$

Show $\Gamma \models H$. Assume $\mathcal{A} \models \Gamma$.

$\Rightarrow \mathcal{A} \models \exists x\, F$ (by IH) $\Rightarrow$ there is a $u \in U_{\mathcal{A}}$ s.t. $\mathcal{A}[u/x] \models F$

$\Rightarrow \mathcal{A}[u/y] \models F[y/x]$ because $y = x$ or $y \notin fv(F)$

$\mathcal{A}[u/y] \models \Gamma$ because y not free in Γ

$\Rightarrow \mathcal{A}[u/y] \models H$ by IH

$\Rightarrow \mathcal{A} \models H$ because y not free in H

Theorem (ND can simulate SC)

If $\vdash_G \Gamma \Rightarrow \Delta$ *then* $\Gamma, \neg\Delta \vdash_N \bot$ *(where* $\neg\{F_1, \ldots\} = \{\neg F_1, \ldots\}$*)*

Proof by induction on (the depth of) $\vdash_G \Gamma \Rightarrow \Delta$

Corollary (Completeness of ND)

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

Proof as before: compactness, completeness of $\vdash_G$, translation to $\vdash_N$

Translation from $\vdash_N$ to $\vdash_G$ also as before: $I \mapsto R$, $E \mapsto L + cut$

Hilbert System

Hilbert System

Additional rule $\forall I$:
if F is provable then $\forall y\, F[y/x]$ is provable
provided x not free in the assumptions and ($y = x$ or $y \notin fv(F)$)

Additional axioms:
$\forall x\, F \rightarrow F[t/x]$
$F[t/x] \rightarrow \exists x\, F$
$\forall x(G \rightarrow F) \rightarrow (G \rightarrow \forall y\, F[y/x]) \quad (*)$
$\forall x(F \rightarrow G) \rightarrow (\exists y\, F[y/x] \rightarrow G) \quad (*)$

$(*)$ if $x \notin fv(G)$ and ($y = x$ or $y \notin fv(F)$)

Equivalence of Hilbert and ND

As before, with additional cases.

First-order Predicate Logic
Theories

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 sentence is a closed formula.

In the sequel, S is a set of sentences.

Theories

Definition

A theory is a set of sentences S such that S is closed under consequence: If $S \models F$ and F is a sentence, then $F \in S$.

Let $\mathcal{A}$ be a Σ-structure:
$Th(\mathcal{A})$ is the set of all sentences true in $\mathcal{A}$:
$Th(\mathcal{A}) = \{F \mid F\ \Sigma\text{-sentence and } \mathcal{A} \models F\}$

Lemma

Let $\mathcal{A}$ be a Σ-structure and F a Σ-sentence.
Then $\mathcal{A} \models F$ iff $Th(\mathcal{A}) \models F$.

Corollary

$Th(\mathcal{A})$ is a theory.

Lemma

Let $\mathcal{A}$ be a Σ-structure and F a Σ-sentence.
Then $\mathcal{A} \models F$ 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 $Th(\mathcal{A}) \models F$
$\Rightarrow$ for all $\mathcal{B}$, if $\mathcal{B} \models Th(\mathcal{A})$ then $\mathcal{B} \models F$
$\Rightarrow \mathcal{A} \models F$ because $\mathcal{A} \models Th(\mathcal{A})$

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

$Th(\mathbb{R}, +, \leq)$ is called linear real arithmetic.
It is decidable.

$Th(\mathbb{R}, +, *, \leq)$ is called real arithmetic.
It is decidable.

$Th(\mathbb{Z}, +, \leq)$ is called linear integer arithmetic or Presburger arithmetic.
It is decidable.

$Th(\mathbb{Z}, +, *, \leq)$ is called integer arithmetic.
It is not even semidecidable (= r.e.).

Decidability via special algorithms.

Consequences

Definition

Let S be a set of Σ-sentences.

$Cn(S)$ is the set of consequences of S:
$Cn(S) = \{F \mid F \ \Sigma\text{-sentence and } S \models F\}$

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

If S is a set of Σ-sentences, $Cn(S)$ is a theory.

Proof Assume F is closed and $Cn(S) \models F$. Show $F \in Cn(S)$, i.e. $S \models F$. Assume $\mathcal{A} \models S$. Thus $\mathcal{A} \models Cn(S)$ (*) and hence $\mathcal{A} \models F$, i.e. $S \models F$. (*): Assume $G \in Cn(S)$, i.e. $S \models G$. With $\mathcal{A} \models S$ the desired $\mathcal{A} \models G$ follows.

Axioms

Definition

Let S be a set of Σ-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.

Example
$Cn(\{\forall x \forall y \forall z\ (x * y) * z = x * (y * z)\})$ is incomplete:
neither $\forall x \forall y\ x * y = y * x$ nor its negation are present.

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 If T is unsatisfiable, then T is complete (because $T \models F$ for all F) and all models are elementarily equivalent.
Now assume T has a model $\mathcal{M}$.
"$\Rightarrow$"
Assume T is complete. Let $F \in Th(\mathcal{M})$.
We cannot have $T \models \neg F$ because $\mathcal{M} \models T$ would imply $\mathcal{M} \models \neg F$ but $\mathcal{M} \models F$ because $F \in Th(\mathcal{M})$. Thus $T \models F$ by completeness.
Therefore every formula that is true in some model of T is true in all models of T.
"$\Leftarrow$"
Assume all models of T are elem.eq. Let F be closed.
Either $\mathcal{M} \models F$ or $\mathcal{M} \models \neg F$. By elem.eq. $T \models F$ or $T \models \neg F$.
Why? Assume $\mathcal{M} \models F$ (similar for $\mathcal{M} \models \neg F$).
To show $T \models F$, assume $\mathcal{A} \models T$ and show $\mathcal{A} \models F$.
$\Rightarrow Th(\mathcal{A}) = Th(\mathcal{M})$ by elem.eq.
$\Rightarrow$ for all closed F, $\mathcal{A} \models F$ iff $\mathcal{M} \models F$
$\Rightarrow \mathcal{A} \models F$ because $\mathcal{M} \models F$

Quantifier Elimination

Helpful lemmas

Let S be a set of sentences.

Lemma

$S \models F$ iff $S \models \forall F$

Lemma

If $S \models F \leftrightarrow G$ then $S \models H[F] \leftrightarrow H[G]$,
i.e. one can replace a subformula F of H by G.

Quantifier elimination

Definition

If $T \models F \leftrightarrow F'$ we say that F and F' are T-equivalent.

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

In linear real arithmetic:

$$\exists x \exists y\ (3 * x + 5 * y = 7)\ \leftrightarrow\ ?$$

$$\forall y\ (x < y \wedge y < z)\ \leftrightarrow\ ?$$

$$\exists y\ (x < y \wedge y < z)\ \leftrightarrow\ ?$$

Quantifier elimination

A quantifier-elimination procedure (QEP) for a theory T and a set of formulas $\mathcal{F}$ is a function that computes for every $F \in \mathcal{F}$ a quantifier-free T-equivalent.

Lemma

Let T be a theory such that

- *T has a QEP for all formulas and*
- *for all ground formulas G, $T \models G$ or $T \models \neg G$, and it is decidable which is the case.*

Then T is decidable and complete.

Simplifying quantifier elimination: one $\exists$

Fact

If T has a QEP for all $\exists x\, F$ 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: $\exists x \bigwedge$ *literals*

Lemma

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 $\exists x\,(L_1 \wedge \cdots \wedge L_n)$ where each L_i is a literal that contains x

Define: $qe1(\exists x\, F)$ where F is quantifier-free
Method: DNF; miniscoping; $qe1c$

This is the end of the generic part of quantifier elimination.
The rest is theory specific.

Eliminating "$\neg$"

Motivation: $\neg x < y \;\leftrightarrow\; y < x \vee y = x$ for linear orderings

Assume that there is a computable function *aneg* that maps every negated atom to a quantifier-free and negation-free T-equivalent formula.

Lemma

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 $\exists x\, (A_1 \wedge \cdots \wedge A_n)$ where each atom A_i contains x

Define: $qe1(\exists x\, F)$ where F quantifier-free
Method: NNF; *aneg*; DNF; miniscoping; *qe1ca*

Quantifier Elimination
Dense Linear Orders Without Endpoints

Dense Linear Orders Without Endpoints

$\Sigma = \{<, =\}$

Let DLO stand for "dense linear order without endpoints" and for 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 DLOs are isomorphic.

Quantifier elimination example

Example

$DLO \models \exists y\ (x < y \wedge y < z)\ \leftrightarrow$

Eliminiation of "$\neg$"

Elimination of negative literals (function *aneg*):

$DLO \models \neg x = y \leftrightarrow x < y \lor y < x$

$DLO \models \neg x < y \leftrightarrow x = y \lor y < x$

Quantifier elimination for conjunctions of atoms

QEP $qe1ca(\exists x\ (A_1 \wedge \cdots \wedge A_n)$ where x occurs in all A_i:

1. Eliminate "$=$": Drop all A_i of the form $x = x$.

If some A_i is of the form $x = y$ (x and y different), eliminate $\exists x$:

$$\exists x\ (x = t \wedge F)\ \equiv\ F[t/x] \qquad (x \text{ does not occur in } t)$$

Otherwise:

2. Eliminate $x < x$: return $\perp$
3. Separate atoms into lower and upper bounds for x and use

$$DLO \models\ \exists x(\bigwedge_{i=1}^{m} l_i < x \wedge \bigwedge_{j=1}^{n} x < u_j)\ \leftrightarrow\ \bigwedge_{i=1}^{m}\bigwedge_{j=1}^{n} l_i < u_j$$

Special case: $\bigwedge_{k=1}^{0} F_k = \top$

Examples

$\exists x\ (x < z \wedge y < x \wedge x < y')\ \leftrightarrow$?
$\forall x\ (x < y)\ \leftrightarrow$?
$\exists x \exists y \exists z\ (x < y \wedge y < z \wedge z < x)\ \leftrightarrow$?

Complexity

Quadratic blow-up with each elimination step

$\Rightarrow$ Eliminating all $\exists$ from

$$\exists x_1 \ldots \exists x_m\ F$$

where F has length n needs $O(n^{2^m})$, assuming F is DNF.

Consequences

- $Cn(DLO)$ has quantifier elimination
- $Cn(DLO)$ is decidable and complete
- All models of DLO (for example $(\mathbb{Q}, <)$ and $(\mathbb{R}, <)$) are elementarily equivalent:
  you cannot distinguish models of DLO by first-order formulas.

Quantifier Elimination
Linear real arithmetic

Linear real arithmetic

$\mathcal{R}_+ = (\mathbb{R}, 0, 1, +, <, =)$, $R_+ = Th(\mathcal{R}_+)$

For convenience 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

A term in normal form: $c_1 \cdot x_1 + \ldots + c_n \cdot x_n + c$
where $c_i \neq 0$, $x_i \neq x_j$ if $i \neq j$.

Every atom A is R_+-equivalent to an atom $0 \bowtie t$ in normal form (NF) where $\bowtie \in \{<, =\}$ and t is in normal form.

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.
Any atom A in normal form that contains x can be transformed into an R_+-equivalent atom solved for x.
Function $sol_x(A)$ solves A for x.

Eliminiation of "$\neg$"

Elimination of negative literals (function *aneg*):

$R_+ \models \neg x = y \leftrightarrow x < y \lor y < x$

$R_+ \models \neg x < y \leftrightarrow x = y \lor y < x$

Fourier-Motzkin Elimination

QEP $qe1ca(\exists x\ (A_1 \wedge \cdots \wedge A_n)$, all A_i in NF and contain x:

1. Let $S = \{sol_x(A_1), \ldots, sol_x(A_n)\}$
2. Eliminate "$=$":

If $(x = t) \in S$ for some t, eliminate $\exists x$:

$$\exists x\ (x = t \wedge F)\ \equiv\ F[t/x] \qquad (x \text{ does not occur in } t)$$

Otherwise return

$$\bigwedge_{(l<x)\in S}\ \bigwedge_{(x<u)\in S} l < u$$

Special case: empty $\bigwedge$ is $\top$

All returned formulas are implicitly put into NF.

Examples

$\exists x \exists y\ (3x + 5y < 7 \wedge 2x - 3y < 2)\ \leftrightarrow\ ?$
$\exists x \forall y\ (3y \leq x \vee x \leq 2y)\ \leftrightarrow\ ?$

Can DNF be avoided?

Ferrante and Rackoff's theorem

Theorem

Let F be quantifier-free and negation-free and assume all atoms that contain x are solved for x. Let S_x be the set of atoms in F that contain x. Let $L = \{l \mid (l < x) \in S_x\}$, $U = \{u \mid (x < u) \in S_x\}$, $E = \{t \mid (x = t) \in S_x\}$. Then

$$R_+ \models \exists x\, F \;\leftrightarrow\; F[-\infty/x] \vee F[\infty/x] \vee \bigvee_{t \in E} F[t/x] \vee \bigvee_{l \in L} \bigvee_{u \in U} F[0.5(l+u)/x]$$

(note: empty $\bigvee$ is $\bot$) where $F[-\infty/x]$ ($F[\infty/x]$) is the following transformation of all solved atoms in F:

$$\begin{aligned} x < t &\mapsto \top\ (\bot)\\ t < x &\mapsto \bot\ (\top)\\ x = t &\mapsto \bot\ (\bot)\end{aligned}$$

Examples

$\exists x\, (y < x \wedge x < z) \leftrightarrow ?$

$\exists x\, x < y \leftrightarrow ?$

Ferrante and Rackoff's procedure

Define $qe1(\exists x\ F)$:

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

Linear integer arithmetic: $\mathcal{Z}_+ := (\mathbb{Z}, +, 0, 1, \leq)$

A problem with $\mathcal{Z}_+$:

$\mathcal{Z}_+ \models \exists x\; x + x = y \;\leftrightarrow\;$?

Fact Linear integer arithmetic does not have quantifier elimination

Presburger Arithmetic is linear integer arithmetic extended with the unary functions "$2 \mid .$", "$3 \mid .$", ...

(Alternative: "$. = . \pmod 2$", "$. = . \pmod 3$", ...)

Notation: $\mathcal{P} := \mathcal{Z}_+$ extended with "$k \mid .$"

For convenience: add constants $c \in \mathbb{Z}$ and multiplication with constants $c \in \mathbb{Z}$

Normal form of atoms:

$0 \leq c_1 \cdot x_1 + \ldots + c_n \cdot x_n + c$

$k \mid c_1 \cdot x_1 + \ldots + c_n \cdot x_n + c$

where $c_i \neq 0$ and $k \geq 1$

Where necessary, atoms are put into normal form

Presburger Arithmetic

Elimination of $\neg$:

$\mathcal{Z}_+ \models \neg\, s \leq t \leftrightarrow t + 1 \leq s$

$\mathcal{Z}_+ \models \neg\, k \mid t \leftrightarrow k \mid t + 1 \vee k \mid t + 2 \vee \cdots \vee k \mid t + (k-1)$

Elimination of $\neg \mid$ expensive and not really necessary.
Can treat $\neg \mid$ like $\mid$

Quantifier Elimination for $\mathcal{P}$

Step 1

$qe1ca(\exists x\ F)$
where $F = A_1 \wedge \cdots \wedge A_l$
where all A_i are atoms in normal form which contain x

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 $\wedge\ m \mid x$

Quantifier Elimination for $\mathcal{P}$

Step 1

$qe1ca(\exists x\ A_1 \wedge \cdots \wedge A_l)$

Step 1: Set all coeffs of x in F to 1 or -1
The details, in one step:

Let m be the (positive) lcm of all coeffs of x (eg lcm $\{-6, 9\} = 18$)
Let R be $coeff1(A_1) \wedge \cdots \wedge coeff1(A_l)\ \wedge m \mid x$ (result)
where
$coeff1(0 \leq c_1 \cdot x_1 + \ldots + c_n \cdot x_n + c) = (0 \leq c'_1 \cdot x_1 + \ldots + c'_n \cdot x_n + c')$
$coeff1(d \mid c_1 \cdot x_1 + \ldots + c_n \cdot x_n + c) = (d' \mid c'_1 \cdot x_1 + \ldots + c'_n \cdot x_n + c')$
$x_k = x$
$m' = m/|c_k|$
$c'_i = m' \cdot c_i$ if $i \neq k$
$c'_k = if\ \ c_k > 0\ \ then\ 1\ \ else\ \ -1$
$c' = m' \cdot c$
$d' = m' \cdot d$

Lemma $\mathcal{P} \models (\exists x\ F) \leftrightarrow (\exists x\ R)$

Quantifier Elimination for $\mathcal{P}$

Step 2

$$A_L := \text{set of all } 0 \leq x + t \text{ in } R \qquad L := \{-t \mid (0 \leq x + t) \in A_L\}$$
$$A_U := \text{set of all } 0 \leq -x + t \text{ in } R \qquad U := \{t \mid (0 \leq -x + t) \in A_U\}$$

$D :=$ the set of all $d \mid t$ in R

$m :=$ the (pos.) lcm of $\{d \mid (d \mid t) \in D \text{ for some } t\}$

The quantifier-free result:

$$R' := \quad \textit{if } L = \emptyset$$
$$\text{then } \bigvee_{i=0}^{m-1} \bigwedge D[i/x]$$
$$\text{else } \bigvee_{i=0}^{m-1} \bigvee_{l \in L} R[l + i/x]$$

Optimisation: use U instead of L

Lemma (Periodicity Lemma)

If $A \in D$, i.e. $A = (d \mid x + t)$ and $x \notin fv(t)$, and $i \equiv j \pmod d$ then $\mathcal{P} \models A[i/x] \leftrightarrow A[j/x]$.

Incompleteness of (Integer) Arithmetic

[Schöning, *Theoretische Informatik*]

Kurt Gödel. *Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I.* 1931.

Kurt Gödel
1906 (Brünn) –
1978 (Princeton)

Syntax of arithmetic:

$$
\begin{array}{rccl}
\text{Variablen:} & V & \to & x \mid y \mid z \mid \ldots \\
\text{Zahlen:} & N & \to & 0 \mid 1 \mid 2 \mid \ldots \\
\text{Terme:} & T & \to & V \mid N \mid (T + T) \mid (T * T) \\
\text{Formeln:} & F & \to & (T = T) \mid \neg F \mid (F \wedge F) \mid (F \vee F) \mid \exists V.\, F
\end{array}
$$

We consider $\forall x.\, F$ as an abberviation for $\neg\exists x.\, \neg F$.

Definition

An occurrence of a variable x in a formula F is bound iff the occurrence is in a subformula of the form $\exists x.\, F'$ within F'.
An occurrence is free iff it is not bound.

Notation: $F(x_1, \ldots, x_k)$ denotes a formula in which at most the variables $x_1, \ldots, x_k$ occur free.

If $n_1, \ldots, n_k \in \mathbb{N}$ then $F(n_1, \ldots, n_k)$ is the result of substituting $n_1, \ldots, n_k$ for the free occurrences of $x_1, \ldots, x_k$.

Example

$$\begin{aligned} F(x, y) &= (x = y \ \wedge \ \exists x.\, x = y) \\ F(5, 7) &= (5 = 7 \ \wedge \ \exists x.\, x = 7) \end{aligned}$$

A sentence is a formula without free variables.

Example

$$\exists x.\ \exists y.\, x = y$$

S is the set of arithmetic sentences.

Definition

W is the set of true sentences of arithmetic:

$$\begin{array}{rll} (t_1 = t_2) \in W & \text{iff} & t_1 \text{ and } t_2 \text{ have the same value.} \\ \neg F \in W & \text{iff} & F \notin W \\ (F \wedge G) \in W & \text{iff} & F \in W \text{ and } G \in W \\ (F \vee G) \in W & \text{iff} & F \in W \text{ or } G \in W \\ \exists x.\, F(x) \in W & \text{iff} & \text{there is some } n \in \mathbb{N} \text{ s.t. } F(n) \in W \end{array}$$

Fact

For every sentence F: $F \in W$ iff $\neg F \notin W$,

NB If a formula with free variables is true or not can depend on the value of the free variables:

$$\exists x.\, x + x = y$$

Therefore absolute truth only makes sense for sentences.

Formulas can represent functions and relations.

Examples

$$F(x, y) = (\exists z.\ y = x + z + 1)$$

represents "$x < y$": $t_1 < t_2$ is an abbreviation of $F(t_1, t_2)$.

$$F(x, y, z) = (\exists k.\ x = k * y + z \wedge z < y)$$

represents "$z = x \bmod y$"

Definition

A partial function $f : \mathbb{N}^k \to \mathbb{N}$ is arithmetically representable iff there is a formula $F(x_1, \ldots, x_k, y)$ s.t. for all $n_1, \ldots, n_k, m \in \mathbb{N}$:

$$f(n_1, \ldots, n_k) = m \quad \text{iff} \quad F(n_1, \ldots, n_k, m) \in W$$

Theorem

Every WHILE-computable function is arithmetically representable.

Theorem

W is not decidable.

Proof.

Let $U \subseteq \mathbb{N}$ be a semi-decidabe but not decidable set.
$\Rightarrow \chi'_U$ is WHILE-computable
$\Rightarrow \chi'_U$ is arithmetically representable by some $F(x, y)$
$\Rightarrow n \in U$ iff $\chi'_U(n) = 1$ iff $F(n, 1) \in W$
$\Rightarrow W$ is not decidable. □

Corollary

W is not semi-decidable.

What is a *proof system*? Minimal requirement:
It must decidable if a given text is a poof of a given formula.

We code proofs as natural numbers.

Definition

A proof system for arithmetic is a decidable predicate

$$Prf : \mathbb{N} \times S \rightarrow \{0, 1\}$$

where $Prf(p, F)$ means "'p is a proof for the sentence F"'.
A proof system Prf is correct iff

$$Prf(p, F) \Rightarrow F \in W.$$

A proof system Prf is complete iff

$$F \in W \Rightarrow \text{ there exists a } p \text{ with } Prf(p, F).$$

Theorem (Gödel)

There is no correct and complete proof system for arithmetic.

Proof.

With every correct and complete proof system
$\chi'_W(F)$ can be programmed:

$p := 0$
while $Prf(p, F) = 0$ **do** $p := p + 1$
output(1)

□

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 (Yuri Matiyasevich, Julia Robinson, Martin Davis, Hilary Putnam, 1949-1970)

It is in general undecidable if a diophantine equation has a solution.

An Isabelle Proof

J. Bayer, M. David, B. Stock, A. Pal, D. Schleicher.
Diophantine Equations and the DPRM Theorem.
Archive of Formal Proofs. 2022.

DPRM = Davis, Putnam, Robinson, Matiyasevich

Higher-Order Logic (HOL)

Types and Terms

Simly typed λ-terms

Types:

$$
\begin{array}{lll}
\tau & ::= & \mathit{bool} \mid \ldots \\
 & \mid & (\tau \to \tau) \\
 & \mid & \alpha \mid \beta \ldots
\end{array}
$$

Terms

$$
\begin{array}{lll}
t & ::= & c \mid d \mid \cdots \mid f \mid h \mid \ldots \\
 & \mid & (t\ t) \\
 & \mid & (\lambda x.\ t)
\end{array}
$$

We assume that every variable and constant has an attached type.
We consider only well-typed terms:

$$
\frac{t_1 : \tau \to \tau' \quad t_2 : \tau}{t_1\ t_2 : \tau'} \qquad \frac{t : \tau'}{\lambda x{:}\tau.\ t : \tau \to \tau'}
$$

Base logic

Formula = term of type *bool*

Theorems: $\Gamma \vdash F$

Base constants: $= : \alpha \to \alpha \to bool$
$\to : bool \to bool \to bool$

Inference rules

$$\frac{}{F \vdash F}\ \textit{assume}$$

$$\frac{}{\vdash t = t}\ \textit{refl}$$

$$\frac{}{\vdash (\lambda x.\ t)\ u = u[t/x]}\ \beta$$

$$\frac{}{\vdash \lambda x.\ (t\ x) = t}\ \eta \quad \text{if } x \notin \mathit{fv}(t)$$

$$\frac{\Gamma_1 \vdash s = t \quad \Gamma_2 \vdash F[s/x]}{\Gamma_1 \cup \Gamma_2 \vdash F[t/x]}\ \textit{subst}$$

$$\frac{\Gamma \vdash s = t}{\Gamma \vdash (\lambda x.\ s) = (\lambda x.\ t)}\ \textit{abs} \quad \text{if } x \notin \mathit{fv}(\Gamma)$$

Inference rules

$$\frac{\Gamma \vdash F}{\Gamma \vdash F[\tau_1/\alpha_1, \ldots]}\ \textit{inst}$$

if $\alpha_1, \ldots$ do not occur in Γ

Inference rules

$$\frac{\Gamma \vdash G}{\Gamma \backslash \{F\} \vdash F \to G} \to I$$

$$\frac{\Gamma_1 \vdash F \to G \quad \Gamma_2 \vdash F}{\Gamma_1 \cup \Gamma_2 \vdash G} \to E$$

$$\frac{\Gamma_1 \vdash F \to G \quad \Gamma_2 \vdash G \to F}{\Gamma_1 \cup \Gamma_2 \vdash F = G} =I$$

Definitions of standard logical symbols

$$\vdash \top = ((\lambda x.\ x) = (\lambda x.\ x))$$

all : $(\alpha \to \mathit{bool}) \to \mathit{bool}$
Notation: $\forall x.\ F$ abbreviates $\mathit{all}(\lambda x.\ F)$

$$\vdash \mathit{all} = (\lambda P.\ P = (\lambda x.\ \top))$$

$$\vdash \bot = (\forall F.\ F)$$

$$\vdash \neg = (\lambda F.\ F \to \bot)$$

$$\vdash (\wedge) = (\lambda F.\ \lambda G.\ \forall H.\ (F \to G \to H) \to H)$$

$$\vdash (\vee) = (\lambda F.\ \lambda G.\ \forall H.\ (F \to H) \to (G \to H) \to H)$$

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

$(\varepsilon\, x.\, x = 5) = 5$
$(\varepsilon\, n.\, 0 \leq n \leq 2) \in \{0, 1, 2\}$
$(\varepsilon\, x.\, \bot)$???

Formally: $eps : (\alpha \to bool) \to \alpha$
$\varepsilon\, x.\, F$ appreviates $eps(\lambda x.\, F)$
Axiom: $P\ x \to P(eps\ P)$