Non-classical logics

Lecture 2: Classical logic, Part 2

30.10.2013

Viorica Sofronie-Stokkermans

sofronie@uni-koblenz.de

Winter Semester 2013/2014

Last time

- Propositional logic (Syntax, Semantics)
- Problems: Checking unsatisfiability

  NP complete

  PTIME for certain fragments of propositional logic
- Normal forms (CNF/DNF)
- Translations to CNF/DNF
- Methods for checking satisfiability

  The Resolution Procedure

  Semantic Tableaux

The Propositional Resolution Calculus

Resolution inference rule:

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

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

(Positive) factorisation inference rule:

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

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

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

Soundness and Completeness

Theorem 1.6. Propositional resolution is sound.

for both the resolution rule and the positive factorization rule

the conclusion of the inference is entailed by the premises.

If N is satisfiable, we cannot deduce $\perp$ from N using the inference rules of the propositional resolution calculus.

If we can deduce $\perp$ from N using the inference rules of the propositional resolution calculus then N is unsatisfiable

Theorem 1.7. Propositional resolution is refutationally complete.

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

Notation

$N \vdash_{Res} \bot$: we can deduce $\bot$ starting from N and using the inference rules of the propositional resolution calculus.

Completeness of Resolution

How to show refutational completeness of propositional resolution:

- We have to show: $N \models \bot \;\Rightarrow\; N \vdash_{Res} \bot$,
  or equivalently: If $N \not\vdash_{Res} \bot$, then N has a model.

- Idea: Suppose that we have computed sufficiently many inferences (and not derived $\bot$).

  Now order the clauses in N according to some appropriate ordering, inspect the clauses in ascending order, and construct a series of valuations.

- The limit valuation can be shown to be a model of N.

Clause Orderings

1. We assume that $\succ$ is any fixed ordering on propositional variables that is *total* and well-founded.

2. Extend $\succ$ to an ordering $\succ_L$ on literals:

$$\begin{array}{lcll} [\neg]P & \succ_L & [\neg]Q & \text{, if } P \succ Q \\ \neg P & \succ_L & P & \end{array}$$

3. Extend $\succ_L$ to an ordering $\succ_C$ on clauses:
   $\succ_C = (\succ_L)_{\mathrm{mul}}$, the multi-set extension of $\succ_L$.

   Notation: $\succ$ also for $\succ_L$ and $\succ_C$.

(well-founded)

Multi-Set Orderings

Let $(M, \succ)$ be a partial ordering. The multi-set extension of $\succ$ to multi-sets over M is defined by

$$\begin{aligned} S_1 \succ_{\text{mul}} S_2 :\Leftrightarrow\ & S_1 \neq S_2 \\ & \text{and } \forall m \in M : [S_2(m) > S_1(m) \\ & \quad \Rightarrow \quad \exists m' \in M : (m' \succ m \text{ and } S_1(m') > S_2(m'))] \end{aligned}$$

Theorem 1.11:

a) $\succ_{\text{mul}}$ is a partial ordering.

b) $\succ$ well-founded $\Rightarrow$ $\succ_{\text{mul}}$ well-founded

c) $\succ$ total $\Rightarrow$ $\succ_{\text{mul}}$ total

Proof:

see Baader and Nipkow, page 22–24.

Example

Suppose $P_5 \succ P_4 \succ P_3 \succ P_2 \succ P_1 \succ P_0$. Then:

$$
\begin{array}{rc}
 & P_0 \vee P_1 \\
\prec & P_1 \vee P_2 \\
\prec & \neg P_1 \vee P_2 \\
\prec & \neg P_1 \vee P_4 \vee P_3 \\
\prec & \neg P_1 \vee \neg P_4 \vee P_3 \\
\prec & \neg P_5 \vee P_5
\end{array}
$$

Stratified Structure of Clause Sets

Let $A \succ B$. Clause sets are then stratified in this form:

	B	$\ldots \vee B$ $\ldots \vee B \vee B$ $\neg B \vee \ldots$	all D where $\max(D) = B$
$\prec$	$\vdots$	$\vdots$	
	A	$\ldots \vee A$ $\ldots \vee A \vee A$ $\neg A \vee \ldots$	all C where $\max(C) = A$

Stratified Structure of Clause Sets

Let $A \succ B$. Clause sets are then stratified in this form:

	B	$\dots \lor B$ $\dots \lor B \lor B$ $\neg B \lor \dots$	all D where $\max(D) = B$
$\prec$	$\vdots$	$\vdots$	
	A	$\dots \lor A$ $\dots \lor A \lor A$ $\neg A \lor \dots$	all C where $\max(C) = A$

Closure of Clause Sets under *Res*

$$
\begin{aligned}
Res(N) &= \{C \mid C \text{ is concl. of a rule in } Res \text{ w/ premises in } N\} \\
Res^0(N) &= N \\
Res^{n+1}(N) &= Res(Res^n(N)) \cup Res^n(N), \text{ for } n \geq 0 \\
Res^*(N) &= \textstyle\bigcup_{n \geq 0} Res^n(N)
\end{aligned}
$$

N is called saturated (wrt. resolution), if $Res(N) \subseteq N$.

Proposition 1.12

(i) $Res^*(N)$ is saturated.

(ii) Res is refutationally complete, iff for each set N of ground clauses:

$$N \models \bot \ \Leftrightarrow \ \bot \in Res^*(N)$$

Construction of Interpretations

Given: set N of clauses, atom ordering $\succ$.

Wanted: Valuation $\mathcal{A}$ such that

- "many" clauses from N are valid in $\mathcal{A}$;
- $\mathcal{A} \models N$, *if* N *is saturated and* $\bot \notin N$.

Construction according to $\succ$, starting with the minimal clause.

Main Ideas of the Construction

- Clauses are considered in the order given by $\prec$. We construct a model for N incrementally.

- When considering C, one already has a partial interpretation I_C (initially $I_C = \emptyset$) available.

  In what follows, instead of referring to partial valuations $\mathcal{A}_C$ we will refer to partial interpretations I_C (the set of atoms which are true in the valuation $\mathcal{A}_C$).

- If C is true in the partial interpretation I_C, nothing is done. ($\Delta_C = \emptyset$).

- If C is false, one would like to change I_C such that C becomes true.

Example

Let $P_5 \succ P_4 \succ P_3 \succ P_2 \succ P_1 \succ P_0$ (max. literals in red)

	clauses C	$I_C = \mathcal{A}_C^{-1}(1)$	Δ_C	Remarks
1	$\neg P_0$			
2	$P_0 \vee P_1$			
3	$P_1 \vee P_2$			
4	$\neg P_1 \vee P_2$			
5	$\neg P_1 \vee P_4 \vee P_3 \vee P_0$			
6	$\neg P_1 \vee \neg P_4 \vee P_3$			
7	$\neg P_1 \vee P_5$			

Example

Let $P_5 \succ P_4 \succ P_3 \succ P_2 \succ P_1 \succ P_0$ (max. literals in red)

	clauses C	$I_C = \mathcal{A}_C^{-1}(1)$	Δ_C	Remarks
1	$\neg P_0$	$\emptyset$	$\emptyset$	true in $\mathcal{A}_C$
2	$P_0 \vee P_1$			
3	$P_1 \vee P_2$			
4	$\neg P_1 \vee P_2$			
5	$\neg P_1 \vee P_4 \vee P_3 \vee P_0$			
6	$\neg P_1 \vee \neg P_4 \vee P_3$			
7	$\neg P_1 \vee P_5$			

Example

Let $P_5 \succ P_4 \succ P_3 \succ P_2 \succ P_1 \succ P_0$ (max. literals in red)

	clauses C	$I_C = \mathcal{A}_C^{-1}(1)$	Δ_C	Remarks
1	$\neg P_0$	$\emptyset$	$\emptyset$	true in $\mathcal{A}_C$
2	$P_0 \vee P_1$	$\emptyset$	$\{P_1\}$	P_1 maximal
3	$P_1 \vee P_2$			
4	$\neg P_1 \vee P_2$			
5	$\neg P_1 \vee P_4 \vee P_3 \vee P_0$			
6	$\neg P_1 \vee \neg P_4 \vee P_3$			
7	$\neg P_1 \vee P_5$			

Example

Let $P_5 \succ P_4 \succ P_3 \succ P_2 \succ P_1 \succ P_0$ (max. literals in red)

	clauses C	$I_C = \mathcal{A}_C^{-1}(1)$	Δ_C	Remarks
1	$\neg P_0$	$\emptyset$	$\emptyset$	true in $\mathcal{A}_C$
2	$P_0 \vee P_1$	$\emptyset$	$\{P_1\}$	P_1 maximal
3	$P_1 \vee P_2$	$\{P_1\}$	$\emptyset$	true in $\mathcal{A}_C$
4	$\neg P_1 \vee P_2$			
5	$\neg P_1 \vee P_4 \vee P_3 \vee P_0$			
6	$\neg P_1 \vee \neg P_4 \vee P_3$			
7	$\neg P_1 \vee P_5$			

Example

Let $P_5 \succ P_4 \succ P_3 \succ P_2 \succ P_1 \succ P_0$ (max. literals in red)

	clauses C	$I_C = \mathcal{A}_C^{-1}(1)$	Δ_C	Remarks
1	$\neg P_0$	$\emptyset$	$\emptyset$	true in $\mathcal{A}_C$
2	$P_0 \vee P_1$	$\emptyset$	$\{P_1\}$	P_1 maximal
3	$P_1 \vee P_2$	$\{P_1\}$	$\emptyset$	true in $\mathcal{A}_C$
4	$\neg P_1 \vee P_2$	$\{P_1\}$	$\{P_2\}$	P_2 maximal
5	$\neg P_1 \vee P_4 \vee P_3 \vee P_0$			
6	$\neg P_1 \vee \neg P_4 \vee P_3$			
7	$\neg P_1 \vee P_5$			

Example

Let $P_5 \succ P_4 \succ P_3 \succ P_2 \succ P_1 \succ P_0$ (max. literals in red)

	clauses C	$I_C = \mathcal{A}_C^{-1}(1)$	Δ_C	Remarks
1	$\neg P_0$	$\emptyset$	$\emptyset$	true in $\mathcal{A}_C$
2	$P_0 \vee P_1$	$\emptyset$	$\{P_1\}$	P_1 maximal
3	$P_1 \vee P_2$	$\{P_1\}$	$\emptyset$	true in $\mathcal{A}_C$
4	$\neg P_1 \vee P_2$	$\{P_1\}$	$\{P_2\}$	P_2 maximal
5	$\neg P_1 \vee P_4 \vee P_3 \vee P_0$	$\{P_1, P_2\}$	$\{P_4\}$	P_4 maximal
6	$\neg P_1 \vee \neg P_4 \vee P_3$			
7	$\neg P_1 \vee P_5$			

Example

Let $P_5 \succ P_4 \succ P_3 \succ P_2 \succ P_1 \succ P_0$ (max. literals in red)

	clauses C	$I_C = \mathcal{A}_C^{-1}(1)$	Δ_C	Remarks
1	$\neg P_0$	$\emptyset$	$\emptyset$	true in $\mathcal{A}_C$
2	$P_0 \vee P_1$	$\emptyset$	$\{P_1\}$	P_1 maximal
3	$P_1 \vee P_2$	$\{P_1\}$	$\emptyset$	true in $\mathcal{A}_C$
4	$\neg P_1 \vee P_2$	$\{P_1\}$	$\{P_2\}$	P_2 maximal
5	$\neg P_1 \vee P_4 \vee P_3 \vee P_0$	$\{P_1, P_2\}$	$\{P_4\}$	P_4 maximal
6	$\neg P_1 \vee \neg P_4 \vee P_3$	$\{P_1, P_2, P_4\}$	$\emptyset$	P_3 not maximal; *min. counter-ex.*
7	$\neg P_1 \vee P_5$	$\{P_1, P_2, P_4\}$	$\{P_5\}$	

$I = \{P_1, P_2, P_4, P_5\} = \mathcal{A}^{-1}(1)$: $\mathcal{A}$ is not a model of the clause set

$\Rightarrow$ there exists a counterexample.

Main Ideas of the Construction

- Clauses are considered in the order given by $\prec$.
- When considering C, one already has a partial interpretation I_C (initially $I_C = \emptyset$) available.
- If C is true in the partial interpretation I_C, nothing is done. ($\Delta_C = \emptyset$).
- If C is false, one would like to change I_C such that C becomes true.

Main Ideas of the Construction

- Changes should, however, be *monotone*. One never deletes anything from I_C and the truth value of clauses smaller than C should be maintained the way it was in I_C.
- Hence, one chooses $\Delta_C = \{A\}$ if, and only if, C is false in I_C, if A occurs positively in C (*adding A will make C become true*) and if this occurrence in C is strictly maximal in the ordering on literals (*changing the truth value of A has no effect on smaller clauses*).

Resolution Reduces Counterexamples

$$\frac{\neg P_1 \vee P_4 \vee P_3 \vee P_0 \quad \neg P_1 \vee \neg P_4 \vee P_3}{\neg P_1 \vee \neg P_1 \vee P_3 \vee P_3 \vee P_0}$$

Construction of I for the extended clause set:

	clauses C	I_C	Δ_C	Remarks
1	$\neg P_0$	$\emptyset$	$\emptyset$	
2	$P_0 \vee P_1$	$\emptyset$	$\{P_1\}$	
3	$P_1 \vee P_2$	$\{P_1\}$	$\emptyset$	
4	$\neg P_1 \vee P_2$	$\{P_1\}$	$\{P_2\}$	
8	$\neg P_1 \vee \neg P_1 \vee P_3 \vee P_3 \vee P_0$	$\{P_1, P_2\}$	$\emptyset$	P_3 occurs twice *minimal counter-ex.*
5	$\neg P_1 \vee P_4 \vee P_3 \vee P_0$	$\{P_1, P_2\}$	$\{P_4\}$	
6	$\neg P_1 \vee \neg P_4 \vee P_3$	$\{P_1, P_2, P_4\}$	$\emptyset$	counterexample
7	$\neg P_1 \vee P_5$	$\{P_1, P_2, P_4\}$	$\{P_5\}$	

The same I, but smaller counterexample, hence some progress was made.

Factorization Reduces Counterexamples

$$\frac{\neg P_1 \vee \neg P_1 \vee P_3 \vee P_3 \vee P_0}{\neg P_1 \vee \neg P_1 \vee P_3 \vee P_0}$$

Construction of I for the extended clause set:

	clauses C	I_C	Δ_C	Remarks
1	$\neg P_0$	$\emptyset$	$\emptyset$	
2	$P_0 \vee P_1$	$\emptyset$	$\{P_1\}$	
3	$P_1 \vee P_2$	$\{P_1\}$	$\emptyset$	
4	$\neg P_1 \vee P_2$	$\{P_1\}$	$\{P_2\}$	
9	$\neg P_1 \vee \neg P_1 \vee P_3 \vee P_0$	$\{P_1, P_2\}$	$\{P_3\}$	
8	$\neg P_1 \vee \neg P_1 \vee P_3 \vee P_3 \vee P_0$	$\{P_1, P_2, P_3\}$	$\emptyset$	true in $\mathcal{A}_C$
5	$\neg P_1 \vee P_4 \vee P_3 \vee P_0$	$\{P_1, P_2, P_3\}$	$\emptyset$	
6	$\neg P_1 \vee \neg P_4 \vee P_3$	$\{P_1, P_2, P_3\}$	$\emptyset$	true in $\mathcal{A}_C$
7	$\neg P_3 \vee P_5$	$\{P_1, P_2, P_3\}$	$\{P_5\}$	

The resulting $I = \{P_1, P_2, P_3, P_5\}$ is a model of the clause set.

Construction of Candidate Models Formally

Let $N, \succ$ be given. We define sets I_C and Δ_C for all ground clauses C over the given signature inductively over $\succ$:

$$I_C \quad := \quad \bigcup_{C \succ D} \Delta_D$$

$$\Delta_C \quad := \quad \begin{cases} \{A\}, & \text{if } C \in N,\ C = C' \vee A,\ A \succ C',\ I_C \not\models C \\ \emptyset, & \text{otherwise} \end{cases}$$

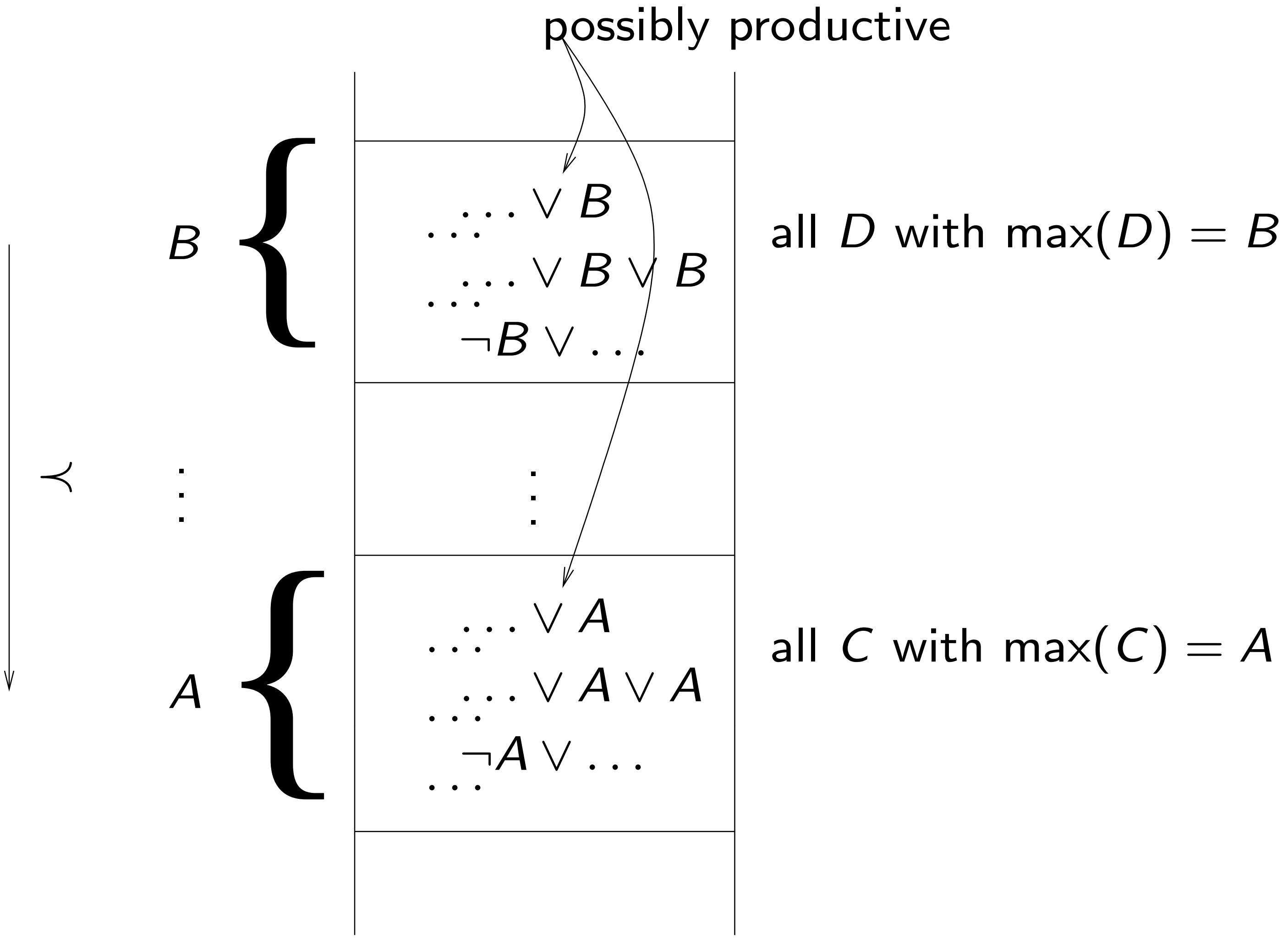
We say that C produces A, if $\Delta_C = \{A\}$.

The candidate model for N (wrt. $\succ$) is given as $I_N^{\succ} := \bigcup_C \Delta_C$.

We also simply write I_N, or I, for $I_N^{\succ}$ if $\succ$ is either irrelevant or known from the context.

Structure of $N, \succ$

Let $A \succ B$; producing a new atom does not affect smaller clauses.



Model Existence Theorem

Theorem 1.14 (Bachmair & Ganzinger):
Let $\succ$ be a clause ordering, let N be saturated wrt. *Res*, and suppose that $\bot \notin N$. Then $I_N^{\succ} \models N$.

Corollary 1.15:
Let N be saturated wrt. *Res*. Then $N \models \bot \Leftrightarrow \bot \in N$.

Model Existence Theorem

Proof:

Suppose $\bot \notin N$, but $I_N^{\succ} \not\models N$. Let $C \in N$ minimal (in $\succ$) such that $I_N^{\succ} \not\models C$. Since C is false in I_N, C is not productive. As $C \neq \bot$ there exists a maximal atom A in C.

Case 1: $C = \neg A \vee C'$ (i.e., the maximal atom occurs negatively)
$\Rightarrow I_N \models A$ and $I_N \not\models C'$
$\Rightarrow$ some $D = D' \vee A \in N$ produces A. As $\frac{D' \vee A \qquad \neg A \vee C'}{D' \vee C'}$, we infer that $D' \vee C' \in N$, and $C \succ D' \vee C'$ and $I_N \not\models D' \vee C'$
$\Rightarrow$ contradicts minimality of C.

Case 2: $C = C' \vee A \vee A$. Then $\frac{C' \vee A \vee A}{C' \vee A}$ yields a smaller counterexample $C' \vee A \in N$. $\Rightarrow$ contradicts minimality of C.

Ordered Resolution with Selection

Ideas for improvement:

1. In the completeness proof (Model Existence Theorem) one only needs to resolve and factor maximal atoms
   $\Rightarrow$ if the calculus is restricted to inferences involving maximal atoms, the proof remains correct
   $\Rightarrow$ *order restrictions*

2. In the proof, it does not really matter with which negative literal an inference is performed
   $\Rightarrow$ choose a negative literal don't-care-nondeterministically
   $\Rightarrow$ *selection*

Selection Functions

A selection function is a mapping

$$S : C \quad \mapsto \quad \text{set of occurrences of } \textit{negative} \text{ literals in } C$$

Example of selection with selected literals indicated as $\boxed{X}$:

$$\boxed{\neg A} \vee \neg A \vee B$$

$$\boxed{\neg B_0} \vee \boxed{\neg B_1} \vee A$$

Ordered resolution

In the completeness proof, we talk about (strictly) maximal literals of clauses.

Resolution Calculus $Res_S^{\succ}$

$$\frac{C \vee A \qquad D \vee \neg A}{C \vee D}$$ [ordered resolution with selection]

if

(i) $A \succ C$;

(ii) nothing is selected in C by S;

(iii) $\neg A$ is selected in $D \vee \neg A$,
or else nothing is selected in $D \vee \neg A$ and $\neg A \succeq \max(D)$.

Note: For positive literals, $A \succ C$ is the same as $A \succ \max(C)$.

Resolution Calculus $Res_S^{\succ}$

$$\frac{C \vee A \vee A}{(C \vee A)} \qquad \text{[ordered factoring]}$$

if A is maximal in C and nothing is selected in C.

Search Spaces Become Smaller

1	$A \lor B$	
2	$A \lor \boxed{\neg B}$	
3	$\neg A \lor B$	
4	$\neg A \lor \boxed{\neg B}$	
5	$B \lor B$	Res 1, 3
6	B	Fact 5
7	$\neg A$	Res 6, 4
8	A	Res 6, 2
9	$\perp$	Res 8, 7

we assume $A \succ B$ and S as indicated by $\boxed{X}$. The maximal literal in a clause is depicted in red.

With this ordering and selection function the refutation proceeds strictly deterministically in this example. Generally, proof search will still be non-deterministic but the search space will be much smaller than with unrestricted resolution.

Today

Propositional Logic

- Tableaux calculus

First-order Logic

1.6 Semantic Tableaux

Literature:

M. Fitting: First-Order Logic and Automated Theorem Proving, Springer-Verlag, New York, 1996, chapters 3, 6, 7.

R. M. Smullyan: First-Order Logic, Dover Publ., New York, 1968, revised 1995.

Like resolution, semantic tableaux were developed in the sixties, by R. M. Smullyan on the basis of work by Gentzen in the 30s and of Beth in the 50s.
(According to Fitting, semantic tableaux were first proposed by the Polish scientist Z. Lis in a paper in Studia Logica 10, 1960 that was only recently rediscovered.)

Idea

Idea (for the propositional case):

A set $\{F \wedge G\} \cup N$ of formulas has a model if and only if $\{F \wedge G,\ F,\ G\} \cup N$ has a model.

A set $\{F \vee G\} \cup N$ of formulas has a model if and only if $\{F \vee G,\ F\} \cup N$ or $\{F \vee G,\ G\} \cup N$ has a model.

(and similarly for other connectives).

To avoid duplication, represent sets as paths of a tree.

Continue splitting until two complementary formulas are found $\Rightarrow$ inconsistency detected.

A Tableau for $\{P \wedge \neg(Q \vee \neg R),\ \neg Q \vee \neg R\}$

1. $P \wedge \neg(Q \vee \neg R)$

2. $\neg Q \vee \neg R$

3. $\neg Q$	4. $\neg R$
5. P	10. P
6. $\neg(Q \vee \neg R)$	11. $\neg(Q \vee \neg R)$
7. $\neg Q$	
8. $\neg\neg R$	
9. R	

This tableau is not "maximal", however the first "path" is. This path is not "closed", hence the set $\{1, 2\}$ is satisfiable. (These notions will all be defined below.)

Properties

Properties of tableau calculi:

analytic: inferences according to the logical content of the symbols.

goal oriented: inferences operate directly on the goal to be proved (unlike, e. g., resolution).

global: some inferences affect the entire proof state (set of formulas), as we will see later.

Propositional Expansion Rules

Expansion rules are applied to the formulas in a tableau and expand the tableau at a leaf. We append the conclusions of a rule (horizontally or vertically) at a *leaf*, whenever the premise of the expansion rule matches a formula appearing *anywhere* on the path from the root to that leaf.

Negation Elimination

$$\frac{\neg\neg F}{F} \qquad \frac{\neg\top}{\bot} \qquad \frac{\neg\bot}{\top}$$

Propositional Expansion Rules

α-**Expansion**

(for formulas that are essentially conjunctions: append subformulas α_1 and α_2 one on top of the other)

$$\frac{\alpha}{\begin{array}{c}\alpha_1\\ \alpha_2\end{array}}$$

β-**Expansion**

(for formulas that are essentially disjunctions: append β_1 and β_2 horizontally, i. e., branch into β_1 and β_2)

$$\frac{\beta}{\beta_1 \mid \beta_2}$$

Classification of Formulas

conjunctive			disjunctive		
α	α_1	α_2	β	β_1	β_2
$X \wedge Y$	X	Y	$\neg(X \wedge Y)$	$\neg X$	$\neg Y$
$\neg(X \vee Y)$	$\neg X$	$\neg Y$	$X \vee Y$	X	Y
$\neg(X \rightarrow Y)$	X	$\neg Y$	$X \rightarrow Y$	$\neg X$	Y

We assume that the binary connective $\leftrightarrow$ has been eliminated in advance.

Tableaux: Notions

A semantic tableau is a marked (by formulas), finite, unordered tree and inductively defined as follows: Let $\{F_1, \ldots, F_n\}$ be a set of formulas.

(i) The tree consisting of a single path

$$\begin{array}{c} F_1 \\ \vdots \\ F_n \end{array}$$

is a tableau for $\{F_1, \ldots, F_n\}$.
(We do not draw edges if nodes have only one successor.)

Tableaux: Notions

(ii) If T is a tableau for $\{F_1, \ldots, F_n\}$ and if T' results from T by applying an expansion rule then T' is also a tableau for $\{F_1, \ldots, F_n\}$.

A path (from the root to a leaf) in a tableau is called closed, if it either contains $\perp$, or else it contains both some formula F and its negation $\neg F$. Otherwise the path is called open.

A tableau is called closed, if all paths are closed.

A tableau proof for F is a closed tableau for $\{\neg F\}$.

Tableaux: Notions

A path P in a tableau is called maximal, if for each non-atomic formula F on P there exists a node in P at which the expansion rule for F has been applied.

In that case, if F is a formula on P, P also contains:

(i) F_1 and F_2, if F is a α-formula,

(ii) F_1 or F_2, if F is a β-formula, and

(iii) F', if F is a negation formula, and F' the conclusion of the corresponding elimination rule.

A tableau is called maximal, if each path is closed or maximal.

Tableaux: Notions

A tableau is called strict, if for each formula the corresponding expansion rule has been applied at most once on each path containing that formula.

A tableau is called clausal, if each of its formulas is a clause.

A Sample Proof

One starts out from the negation of the formula to be proved.

1. $\neg[(P \rightarrow (Q \rightarrow R)) \rightarrow ((P \vee S) \rightarrow ((Q \rightarrow R) \vee S))]$
2. $(P \rightarrow (Q \rightarrow R))$ $[1_1]$
3. $\neg((P \vee S) \rightarrow ((Q \rightarrow R) \vee S))$ $[1_2]$
4. $P \vee S$ $[3_1]$
5. $\neg((Q \rightarrow R) \vee S))$ $[3_2]$
6. $\neg(Q \rightarrow R)$ $[5_1]$
7. $\neg S$ $[5_2]$

8. $\neg P$ $[2_1]$ 9. $Q \rightarrow R$ $[2_2]$

10. P $[4_1]$ 11. S $[4_2]$

There are three paths, each of them closed.

Properties of Propositional Tableaux

We assume that T is a tableau for $\{F_1, \ldots, F_n\}$.

Theorem 1.8:
$\{F_1, \ldots, F_n\}$ satisfiable $\Leftrightarrow$ some path (i.e., the set of its formulas) in T is satisfiable.

(Proof by induction over the structure of T.)

Corollary 1.9:
T closed $\Rightarrow \{F_1, \ldots, F_n\}$ unsatisfiable

Properties of Propositional Tableaux

Theorem 1.10:

Let T be a strict propositional tableau. Then T is finite.

Proof:

New formulas resulting from expansion are either $\perp$, $\top$ or subformulas of the expanded formula. By strictness, on each path a formula can be expanded at most once. Therefore, each path is finite, and a finitely branching tree with finite paths is finite (König's Lemma).

Conclusion: Strict and maximal tableaux can be effectively constructed.

Refutational Completeness

Theorem 1.11:

Let P be a maximal, open path in a tableau. Then set of formulas on P is satisfiable.

Theorem 1.12:

$\{F_1, \ldots, F_n\}$ satisfiable $\Leftrightarrow$ there exists no closed strict tableau for $\{F_1, \ldots, F_n\}$.

Consequences

The validity of a propositional formula F can be established by constructing a strict, maximal tableau T for $\{\neg F\}$:

- T closed $\Leftrightarrow$ F valid.
- It suffices to test complementarity of paths wrt. atomic formulas.
- Which of the potentially many strict, maximal tableaux one computes does not matter. In other words, tableau expansion rules can be applied don't-care non-deterministically ("proof confluence").

Checking validity of formulae

Nota bene: We cannot check the validity of a formula F by constructing a strict, maximal tableau for F.

Checking validity of formulae

Nota bene: We cannot check the validity of a formula F by constructing a strict, maximal tableau for F.

Example: Let $F \quad := \quad (P \vee Q)$

A strict, maximal tableau for F is:

$$1. \quad P \vee Q$$

$$2. \quad P \quad [1_1] \qquad\qquad 3. \quad Q \quad [1_2]$$

This shows that F is satisfiable. Nothing can be inferred about the validity of F this way.

To check whether F is valid, we construct a strict, maximal tableau T for $\neg F$. If T is closed, then $\neg F$ is unsatisfiable, hence F is valid; otherwise F is not valid.

(In the example below, we can construct a strict, maximal tableau for $\neg F$ which is not closed, so F is not valid.)

Part 2: First-Order Logic

First-order logic

- formalizes fundamental mathematical concepts
- is expressive (Turing-complete)
- is not too expressive
  (e. g. not axiomatizable: natural numbers, uncountable sets)
- has a rich structure of decidable fragments
- has a rich model and proof theory

First-order logic is also called (first-order) predicate logic.

2.1 Syntax

Syntax:

- non-logical symbols (domain-specific)
  $\Rightarrow$ terms, atomic formulas
- logical symbols (domain-independent)
  $\Rightarrow$ Boolean combinations, quantifiers

Signature

A signature

$$\Sigma = (\Omega, \Pi),$$

fixes an alphabet of non-logical symbols, where

- Ω is a set of function symbols f with arity $n \geq 0$, written f/n,
- Π is a set of predicate symbols p with arity $m \geq 0$, written p/m.

If $n = 0$ then f is also called a constant (symbol).
If $m = 0$ then p is also called a propositional variable.
We use letters P, Q, R, S, to denote propositional variables.

Variables

Predicate logic admits the formulation of abstract, schematic assertions.
(Object) variables are the technical tool for schematization.

We assume that

$$X$$

is a given countably infinite set of symbols which we use for (the denotation of) variables.

Terms

Terms over Σ (resp., Σ-terms) are formed according to these syntactic rules:

$$
\begin{array}{rclll}
s, t, u, v & ::= & x & , x \in X & \text{(variable)} \\
& | & f(s_1, ..., s_n) & , f/n \in \Omega & \text{(functional term)}
\end{array}
$$

By $\mathrm{T}_\Sigma(X)$ we denote the set of Σ-terms (over X).

A term not containing any variable is called a ground term.

By T_Σ we denote the set of Σ-ground terms.

Terms

In other words, terms are formal expressions with well-balanced brackets which we may also view as marked, ordered trees.
The markings are function symbols or variables.
The nodes correspond to the subterms of the term.
A node v that is marked with a function symbol f of arity n has exactly n subtrees representing the n immediate subterms of v.

Atoms

Atoms (also called atomic formulas) over Σ are formed according to this syntax:

$$
\begin{array}{lcll}
A, B & ::= & p(s_1, \dots, s_m) & , \ p/m \in \Pi \\
\Big[& \mid & (s \approx t) & \text{(equation)} \quad \Big]
\end{array}
$$

Whenever we admit equations as atomic formulas we are in the realm of first-order logic with equality. Admitting equality does not really increase the expressiveness of first-order logic. But deductive systems where equality is treated specifically can be much more efficient.

Literals

$$
\begin{array}{lcll}
L & ::= & A & \text{(positive literal)} \\
 & | & \neg A & \text{(negative literal)}
\end{array}
$$

Clauses

$$
\begin{array}{rclr}
C, D & ::= & \bot & \text{(empty clause)} \\
 & \mid & L_1 \vee \ldots \vee L_k, \;\; k \geq 1 & \text{(non-empty clause)}
\end{array}
$$

General First-Order Formulas

$\mathrm{F}_\Sigma(X)$ is the set of first-order formulas over Σ defined as follows:

$$
\begin{array}{rcll}
F, G, H & ::= & \bot & \text{(falsum)} \\
& \mid & \top & \text{(verum)} \\
& \mid & A & \text{(atomic formula)} \\
& \mid & \neg F & \text{(negation)} \\
& \mid & (F \wedge G) & \text{(conjunction)} \\
& \mid & (F \vee G) & \text{(disjunction)} \\
& \mid & (F \rightarrow G) & \text{(implication)} \\
& \mid & (F \leftrightarrow G) & \text{(equivalence)} \\
& \mid & \forall x\, F & \text{(universal quantification)} \\
& \mid & \exists x\, F & \text{(existential quantification)}
\end{array}
$$

Notational Conventions

We omit brackets according to the following rules:

- $\neg \quad >_p \quad \wedge \quad >_p \quad \vee \quad >_p \quad \rightarrow \quad >_p \quad \leftrightarrow$
  (binding precedences)
- $\vee$ and $\wedge$ are associative and commutative
- $\rightarrow$ is right-associative

$Qx_1, \ldots, x_n\, F$ abbreviates $Qx_1 \ldots Qx_n\, F$.

Notational Conventions

We use infix-, prefix-, postfix-, or mixfix-notation with the usual operator precedences.

Examples:

$$
\begin{array}{ccc}
s + t * u & \text{for} & +(s, *(t, u)) \\
s * u \leq t + v & \text{for} & \leq(*(s, u), +(t, v)) \\
-s & \text{for} & -(s) \\
0 & \text{for} & 0()
\end{array}
$$