Non-classical logics

Lecture 1: Classical logic

Viorica Sofronie-Stokkermans

sofronie@uni-koblenz.de

Winter Semester 2013/2014

Part 1: Propositional Logic

Literature (also for first-order logic)

Schöning: Logik für Informatiker, Spektrum

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

1.1 Syntax

- propositional variables
- logical symbols
  $\Rightarrow$ Boolean combinations

Propositional Variables

Let Π be a set of propositional variables.

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

Propositional Formulas

F_Π is the set of propositional formulas over Π defined as follows:

$$
\begin{array}{llll}
F, G, H & ::= & \bot & \text{(falsum)} \\
& \mid & \top & \text{(verum)} \\
& \mid & P, \quad P \in \Pi & \text{(atomic formula)} \\
& \mid & \neg F & \text{(negation)} \\
& \mid & (F \wedge G) & \text{(conjunction)} \\
& \mid & (F \vee G) & \text{(disjunction)} \\
& \mid & (F \rightarrow G) & \text{(implication)} \\
& \mid & (F \leftrightarrow G) & \text{(equivalence)}
\end{array}
$$

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

1.2 Semantics

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

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

Valuations

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

A Π-valuation is a map

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

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

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

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

$$\begin{aligned}
\mathcal{A}^*(\bot) &= 0 \\
\mathcal{A}^*(\top) &= 1 \\
\mathcal{A}^*(P) &= \mathcal{A}(P) \\
\mathcal{A}^*(\neg F) &= 1 - \mathcal{A}^*(F) \\
\mathcal{A}^*(F \rho G) &= \mathrm{B}_\rho(\mathcal{A}^*(F), \mathcal{A}^*(G)) \\
&\quad \text{with } \mathrm{B}_\rho \text{ the Boolean function associated with } \rho
\end{aligned}$$

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

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

Example: Let's evaluate the formula

$$(P \to Q) \land (P \land Q \to R) \to (P \to R)$$

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

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

(On the blackboard)

1.3 Models, Validity, and Satisfiability

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

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

F is valid (or is a tautology):

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

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

Otherwise F is called unsatisfiable (or contradictory).

1.3 Models, Validity, and Satisfiability

Examples:

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

Obviously, every valid formula is also satisfiable

$F \wedge \neg F$ is unsatisfiable

The formula P is satisfiable, but not valid

Entailment and Equivalence

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

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

Proposition 1.1:
F entails G iff $(F \rightarrow G)$ is valid

Proposition 1.2:
F and G are equivalent iff $(F \leftrightarrow G)$ is valid.

Entailment and Equivalence

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

Definition A set N of formulae is satisfiable if there exists a Π-valuation $\mathcal{A}$ which makes true all formulae in N.

If there is no Π-valuation $\mathcal{A}$ which makes true all formulae in N we say that N is unsatisfiable

Remark: N unsatisfiable iff $N \models \perp$.

Validity vs. Unsatisfiability

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

Proposition 1.3:

$$F \text{ valid} \quad \Leftrightarrow \quad \neg F \text{ unsatisfiable}$$

$$N \models F \quad \Leftrightarrow \quad N \cup \neg F \text{ unsatisfiable}$$

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

Checking Unsatisfiability

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

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

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

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

Some Important Equivalences

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

$$
\begin{array}{rl}
(F \wedge F) \leftrightarrow F & \\
(F \vee F) \leftrightarrow F & \text{(Idempotency)} \\
(F \wedge G) \leftrightarrow (G \wedge F) & \\
(F \vee G) \leftrightarrow (G \vee F) & \text{(Commutativity)} \\
(F \wedge (G \wedge H)) \leftrightarrow ((F \wedge G) \wedge H) & \\
(F \vee (G \vee H)) \leftrightarrow ((F \vee G) \vee H) & \text{(Associativity)} \\
(F \wedge (G \vee H)) \leftrightarrow ((F \wedge G) \vee (F \wedge H)) & \\
(F \vee (G \wedge H)) \leftrightarrow ((F \vee G) \wedge (F \vee H)) & \text{(Distributivity)}
\end{array}
$$

Some Important Equivalences

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

$(F \wedge (F \vee G)) \leftrightarrow F$	
$(F \vee (F \wedge G)) \leftrightarrow F$	(Absorption)
$(\neg\neg F) \leftrightarrow F$	(Double Negation)
$\neg(F \wedge G) \leftrightarrow (\neg F \vee \neg G)$	
$\neg(F \vee G) \leftrightarrow (\neg F \wedge \neg G)$	(De Morgan's Laws)
$(F \wedge G) \leftrightarrow F$, if G is a tautology	
$(F \vee G) \leftrightarrow \top$, if G is a tautology	(Tautology Laws)
$(F \wedge G) \leftrightarrow \bot$, if G is unsatisfiable	
$(F \vee G) \leftrightarrow F$, if G is unsatisfiable	(Tautology Laws)

1.4 Normal Forms

We define conjunctions of formulas as follows:

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

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

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

and analogously disjunctions:

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

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

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

Literals and Clauses

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

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

Literals and Clauses

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

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

Example of clauses:

$\perp$	the empty clause
P	positive unit clause
$\neg P$	negative unit clause
$P \vee Q \vee R$	positive clause
$P \vee \neg Q \vee \neg R$	clause
$P \vee P \vee \neg Q \vee \neg R \vee R$	allow repetitions/complementary literals

CNF and DNF

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

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

Warning: definitions in the literature differ:

are complementary literals permitted?
are duplicated literals permitted?
are empty disjunctions/conjunctions permitted?

Conversion to CNF/DNF

Proposition 1.4:

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

Proof:

We consider the case of CNF.

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

Step 1: Eliminate equivalences:

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

Conversion to CNF/DNF

Step 2: Eliminate implications:

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

Step 3: Push negations downward:

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

Step 4: Eliminate multiple negations:

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

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

Conversion to CNF/DNF

Step 5: Push disjunctions downward:

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

Step 6: Eliminate $\top$ and $\bot$:

$$\begin{aligned}
(F \wedge \top) &\;\Rightarrow_K\; F \\
(F \wedge \bot) &\;\Rightarrow_K\; \bot \\
(F \vee \top) &\;\Rightarrow_K\; \top \\
(F \vee \bot) &\;\Rightarrow_K\; F \\
\neg\bot &\;\Rightarrow_K\; \top \\
\neg\top &\;\Rightarrow_K\; \bot
\end{aligned}$$

Conversion to CNF/DNF

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

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

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

Complexity

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

Satisfiability-preserving Transformations

The goal

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

is unpractical.

But if we relax the requirement to

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

we can get an efficient transformation.

Satisfiability-preserving Transformations

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

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

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

Optimized Transformations

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

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

Optimized Transformations

Proposition 1.5:
Let $F[F']$ be a formula containing neither $\rightarrow$ nor $\leftrightarrow$; let P be a propositional variable not occurring in $F[F']$.

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

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

Proof:
Exercise.

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

Optimized Transformations

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

The following are equivalent:

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

Decision Procedures for Satisfiability

- Simple Decision Procedures

  truth table method

- The Resolution Procedure

- Tableaux

...

1.5 The Propositional Resolution Calculus

Resolution inference rule:

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

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

(Positive) factorisation inference rule:

$$\frac{C \lor A \lor A}{C \lor A}$$

The Resolution Calculus *Res*

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

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

Sample Refutation

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

Resolution with Implicit Factorization *RIF*

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

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

Soundness and Completeness

Theorem 1.6. Propositional resolution is sound.

for both the resolution rule and the positive factorization rule

the conclusion of the inference is entailed by the premises.

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

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

Theorem 1.11:

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

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

c) $\succ$ total $\Rightarrow$ $\succ_{\mathsf{mul}}$ 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	$\ldots \vee B$ $\ldots \vee B \vee B$ $\neg B \vee \ldots$	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 $\max(C) = A$

Closure of Clause Sets under *Res*

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

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.