# Decision Procedures for Verification

Part 1. Propositional Logic (2)

31.10.2013

Viorica Sofronie-Stokkermans

sofronie@uni-koblenz.de

# Last time

## 1.1 Syntax

- Language
  - propositional variables
  - logical symbols
    $\Rightarrow$ Boolean combinations
- Propositional Formulae

## 1.2 Semantics

- Valuations
- Truth value of a formula in a valuation
- Models, Validity, and Satisfiability

## 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).

# 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 $\Pi$-valuations $\mathcal{A}$: if $\mathcal{A} \models G$ for all $G \in N$, then $\mathcal{A} \models F$.

# Validity vs. Unsatisfiability

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

**Proposition 1.3:**

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

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

$Q$: In a similar way, entailment $N \models F$ can be reduced to unsatisfiability. How?

# Validity vs. Unsatisfiability

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

**Proposition 1.4:**

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

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)

# Checking Unsatisfiability

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

For sets of propositional formulae of a certain type, satisfiability can be checked in polynomial time:

**Examples:** 2SAT, Horn-SAT (will be discussed in the exercises)

Dichotomy theorem. Schaefer [Schaefer, STOC 1978] identified six classes of sets $S$ of Boolean formulae for which $SAT(S)$ is in PTIME. He proved that all other types of sets of formulae yield an NP-complete problem.

# Substitution Theorem

**Proposition 1.5:**

Let $F$ and $G$ be equivalent formulas, let $H$ be a formula in which $F$ occurs as a subformula.

Then $H$ is equivalent to $H'$ where $H'$ is obtained from $H$ by replacing the occurrence of the subformula $F$ by $G$.
(Notation: $H = H[F]$, $H' = H[G]$.)

Proof: By induction over the formula structure of $H$.

# Structural Induction

**Goal:** Prove a property $P$ of propositional formulae

Prove that for every formula $F$, $P(F)$ holds.

**Induction basis:** Show that $P(F)$ holds for all $F \in \Pi \cup \{\top, \bot\}$

Let $F$ be a formula (not in $\Pi \cup \{\top, \bot\}$).

**Induction hypothesis:** We assume that $P(G)$ holds for all strict subformulae $G$ of $F$.

**Induction step:** Using the induction hypothesis, we show that $P(F)$ holds as well.
In order to prove that $P(F)$ holds we usually need to consider various cases (reflecting the way the formula $F$ is built):

Case 1: $F = \neg G$
Case 2: $F = G_1 \wedge G_2$
Case 3: $F = G_1 \vee G_2$
Case 4: $F = G_1 \rightarrow G_2$
Case 5: $F = G_1 \leftrightarrow G_2$

# Some Important Equivalences

**Proposition 1.6:**

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

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

$$(F \vee F) \leftrightarrow F \qquad \text{(Idempotency)}$$

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

$$(F \vee G) \leftrightarrow (G \vee F) \qquad \text{(Commutativity)}$$

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

$$(F \vee (G \vee H)) \leftrightarrow ((F \vee G) \vee H) \qquad \text{(Associativity)}$$

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

$$(F \vee (G \wedge H)) \leftrightarrow ((F \vee G) \wedge (F \vee H)) \qquad \text{(Distributivity)}$$

# Some Important Equivalences

**Proposition 1.7:**

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

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

$$(F \vee (F \wedge G)) \leftrightarrow F \qquad \text{(Absorption)}$$

$$(\neg\neg F) \leftrightarrow F \qquad \text{(Double Negation)}$$

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

$$\neg(F \vee G) \leftrightarrow (\neg F \wedge \neg G) \qquad \text{(De Morgan's Laws)}$$

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

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

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

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

## 1.4 Normal Forms

We define conjunctions of formulas as follows:

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

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

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

and analogously disjunctions:

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

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

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

# 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:**

| | |
|---|---|
| $\bot$ | 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?

# CNF and DNF

Checking the validity of CNF formulas or the unsatisfiability of DNF formulas is easy:

A formula in CNF is valid, if and only if each of its disjunctions contains a pair of complementary literals $P$ and $\neg P$.

Conversely, a formula in DNF is unsatisfiable, if and only if each of its conjunctions contains a pair of complementary literals $P$ and $\neg P$.

On the other hand, checking the unsatisfiability of CNF formulas or the validity of DNF formulas is known to be coNP-complete.

# Conversion to CNF/DNF

**Proposition 1.8:**

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

Proof:

We consider the case of CNF.

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

**Step 1:** Eliminate equivalences:

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

# Conversion to CNF/DNF

**Step 2:** Eliminate implications:

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

**Step 3:** Push negations downward:

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

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

**Step 4:** Eliminate multiple negations:

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

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

# Conversion to CNF/DNF

**Step 5:** Push disjunctions downward:

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

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

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

# Conversion to CNF/DNF

Proving termination is easy for most of the steps; only steps 1, 3 and 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 1.9:**

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:

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

# Decision Procedures for Satisfiability

- Simple Decision Procedures

  truth table method

- The Resolution Procedure

- The Davis-Putnam-Logemann-Loveland Algorithm

## 1.5 Inference Systems and Proofs

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

$$(F_1, \ldots, F_n, F_{n+1}), \;\; n \geq 0,$$

called inferences or inference rules, and written

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

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

# Proofs

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

(i) $F_k = F$,

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

# Soundness and Completeness

Provability $\vdash_\Gamma$ of $F$ from $N$ in $\Gamma$:

$N \vdash_\Gamma F \quad :\Leftrightarrow$ there exists a proof $\Gamma$ of $F$ from $N$.

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

$$\frac{F_1 \;\; \ldots \;\; F_n}{F} \in \Gamma \quad \Rightarrow \quad F_1, \ldots, F_n \models F$$

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

$$N \models F \quad \Rightarrow \quad N \vdash_\Gamma F$$

$\Gamma$ is called refutationally complete $\quad :\Leftrightarrow$

$$N \models \bot \quad \Rightarrow \quad N \vdash_\Gamma \bot$$

# 1.6 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:

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

# 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. | $\bot$ | (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. | $\bot$ | (Res. 4. into 7.) |

# Soundness of Resolution

**Theorem 1.10.** Propositional resolution is sound.

Proof:

Let $\mathcal{A}$ valuation. To be shown:

(i) for resolution: $\mathcal{A} \models C \vee A,\ \mathcal{A} \models D \vee \neg A \ \Rightarrow\ \mathcal{A} \models C \vee D$

(ii) for factorization: $\mathcal{A} \models C \vee A \vee A \ \Rightarrow\ \mathcal{A} \models C \vee A$

(i): Assume $\mathcal{A}^*(C \vee A) = 1, \mathcal{A}^*(D \vee \neg A) = 1$.
Two cases need to be considered: (a) $\mathcal{A}^*(A) = 1$, or (b) $\mathcal{A}^*(\neg A) = 1$.

(a) $\mathcal{A} \models A \Rightarrow \mathcal{A} \models D \Rightarrow \mathcal{A} \models C \vee D$

(b) $\mathcal{A} \models \neg A \Rightarrow \mathcal{A} \models C \Rightarrow \mathcal{A} \models C \vee D$

(ii): Assume $\mathcal{A} \models C \vee A \vee A$. Note that $\mathcal{A}^*(C \vee A \vee A) = \mathcal{A}^*(C \vee A)$, i.e. the conclusion is also true in $\mathcal{A}$.

# Soundness of Resolution

Note: In propositional logic we have:

1. $\mathcal{A} \models L_1 \vee \ldots \vee L_n \;\Leftrightarrow\;$ there exists $i$: $\mathcal{A} \models L_i$.
2. $\mathcal{A} \models A$ or $\mathcal{A} \models \neg A$.

# 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 $\succ_C$.

# Multi-Set Orderings

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

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

**Theorem 1.11:**

a) $\succ_{\mathsf{mul}}$ 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.

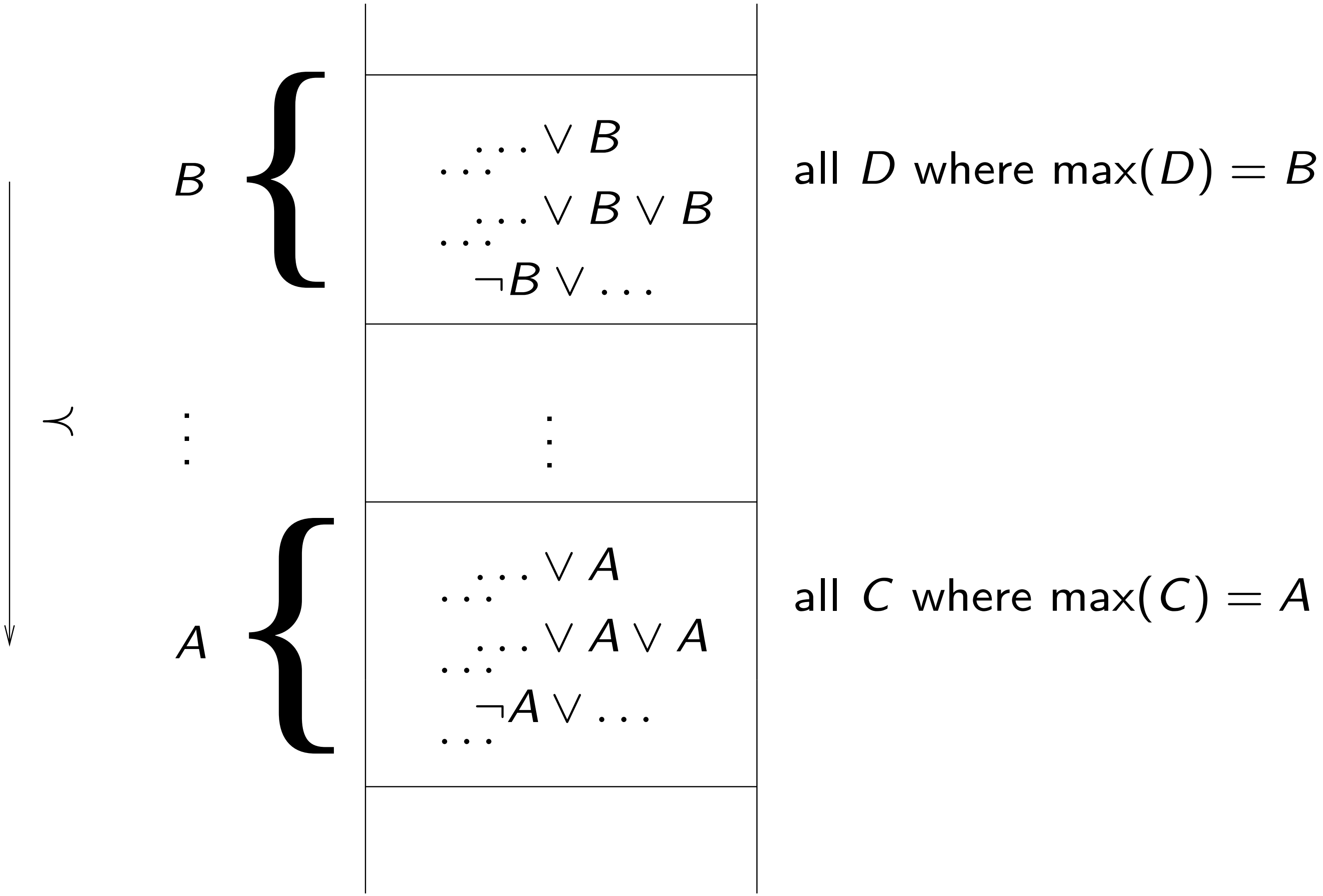
# Example

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:



# Closure of Clause Sets under *Res*

$$Res(N) = \{C \mid C \text{ is concl. of a rule in } Res \text{ w/ premises in } N\}$$

$$Res^0(N) = N$$

$$Res^{n+1}(N) = Res(Res^n(N)) \cup Res^n(N), \text{ for } n \geq 0$$

$$Res^*(N) = \bigcup_{n \geq 0} Res^n(N)$$

$N$ 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 $C$ | $I_C = \mathcal{A}_C^{-1}(1)$ | $\Delta_C$ | 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 $C$ | $I_C = \mathcal{A}_C^{-1}(1)$ | $\Delta_C$ | 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 $C$ | $I_C = \mathcal{A}_C^{-1}(1)$ | $\Delta_C$ | 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 $C$ | $I_C = \mathcal{A}_C^{-1}(1)$ | $\Delta_C$ | 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 $C$ | $I_C = \mathcal{A}_C^{-1}(1)$ | $\Delta_C$ | 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 $C$ | $I_C = \mathcal{A}_C^{-1}(1)$ | $\Delta_C$ | 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 $C$ | $I_C = \mathcal{A}_C^{-1}(1)$ | $\Delta_C$ | 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 $C$ | $I_C$ | $\Delta_C$ | 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<br>*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 $C$ | $I_C$ | $\Delta_C$ | 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 $\Delta_C$ for all ground clauses $C$ over the given signature inductively over $\succ$:

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

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

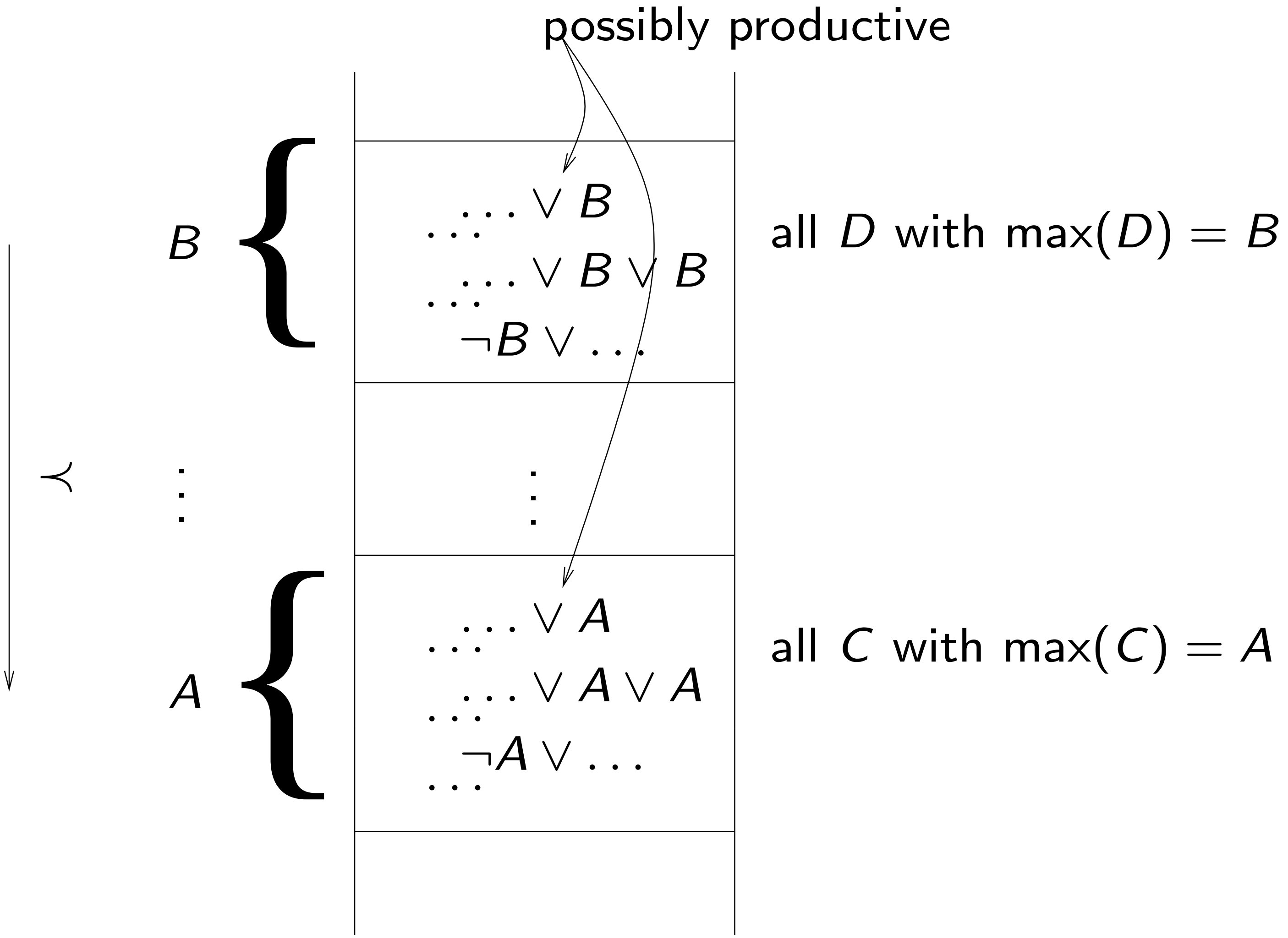
We say that $C$ produces $A$, if $\Delta_C = \{A\}$.

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.



# Some Properties of the Construction

**Proposition 1.13:**

(i) $C = \neg A \vee C' \;\Rightarrow\;$ no $D \succeq C$ produces $A$.

(ii) $C$ productive $\Rightarrow I_C \cup \Delta_C \models C$.

(iii) Let $D' \succ D \succeq C$. Then

$$I_D \cup \Delta_D \models C \Rightarrow I_{D'} \cup \Delta_{D'} \models C \text{ and } I_N \models C.$$

If, in addition, $C \in N$ or $\max(D) \succ \max(C)$:

$$I_D \cup \Delta_D \not\models C \Rightarrow I_{D'} \cup \Delta_{D'} \not\models C \text{ and } I_N \not\models C.$$

# Some Properties of the Construction

(iv) Let $D' \succ D \succ C$. Then

$$I_D \models C \Rightarrow I_{D'} \models C \text{ and } I_N \models C.$$

If, in addition, $C \in N$ or $\max(D) \succ \max(C)$:

$$I_D \not\models C \Rightarrow I_{D'} \not\models C \text{ and } I_N \not\models C.$$

(v) $D = C \vee A$ produces $A \Rightarrow I_N \not\models C$.

# 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 | $A \vee B$ | |
| 2 | $A \vee \boxed{\neg B}$ | |
| 3 | $\neg A \vee B$ | |
| 4 | $\neg A \vee \boxed{\neg B}$ | |
| 5 | $B \vee B$ | Res 1, 3 |
| 6 | $B$ | Fact 5 |
| 7 | $\neg A$ | Res 6, 4 |
| 8 | $A$ | Res 6, 2 |
| 9 | $\bot$ | 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.