M.Lupea

Resolution Proof Method

IN PROPOSITIONAL LOGIC

Resolution Proof Method

- It was proposed by J.A. Robinson in 1965 as a proof method for classical logics
- Dedicated theorem provers based on resolution:
  OTTER, PCPROOVE, AMPHION, Jape
- It was easily adapted to *nonstandard logics* (modal, temporal, many-valued, non-monotonic).
- Its basic aim is to check the ***consistency/inconsistency*** of a set of clauses.
- It is based on syntactic considerations => ***syntactic method***
- The ***validity*** of a formula **is proved by contradiction**=>
  => ***refutation method***

DECISION PROBLEMS IN PROPOSITIONAL/PREDICATE LOGIC

M.Lupea

1. Is a propositional/predicate formula a tautology/theorem?

$$\overset{?}{\models} V \quad \text{or} \quad \overset{?}{\vdash} V$$

2. Is a propositional/predicate formula a logical/syntactic consequence of a set of hypotheses?

$$U_1,\ldots,U_n \overset{?}{\models} V \quad \text{or} \quad U_1,\ldots,U_n \overset{?}{\vdash} V$$

In order to solve these two decision problems, **proof methods** are applied.

Resolution method
- formal system for propositional logic -

M.Lupea

$\text{Res} = (\Sigma_{\text{Res}}, F_{\text{Res}}, A_{\text{Res}}, R_{\text{Res}})$, where:

- $\Sigma_{\text{Res}} = \Sigma_P - \{\rightarrow, \leftrightarrow, \wedge\}$ - the alphabet;
- $F_{\text{Res}} \cup \{\square\}$
  - F_{Res} - the set of all clauses built using the alphabet Σ_{Res};
  - $\square$ is the empty clause, does not contain any literal,

    it symbolizes inconsistency;
- $A_{\text{Res}} = \varnothing$ is the set of axioms ;
- R_{Res} is the set of inference rules containing the ***resolution rule*** *(res)*:

  $f \vee l, g \vee \neg l \vdash_{res} f \vee g$, where l is a literal and $f, g \in F_{\text{Res}}$.

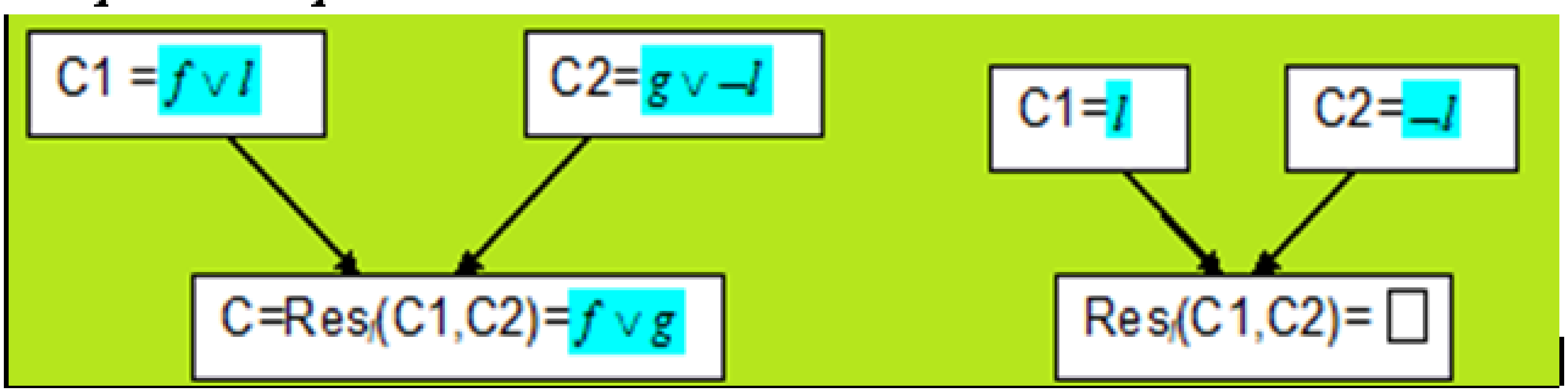
Definitions

M.Lupea

Let l be a literal and $f, g \in F_{\text{Res}}$.

- the clauses $C_1 = f \vee l$ and $C_2 = g \vee \neg l$ are called ***clashing clauses*** and they ***resolve upon the literal*** l.
- <u>notation</u>: $C = \text{Res}_l(C_1, C_2) = f \vee g$, where C is called the ***resolvent*** of the ***parent clauses*** C_1 and C_2 .
- if $C_1 = l$ and $C_2 = \neg l$, then $\text{Res}_l(C_1, C_2) = \square$ (empty clause) which is inconsistent.

Graphical representation:



Algorithm: propositional_resolution

M.Lupea

input: S – a set of propositional clauses
output: message "S is *consistent*" or "S is *inconsistent*"
begin
    $S_0 := S$; $i := 0$;
    do
        choose two clashing clauses: $C_1, C_2 \in S_i$;
        $C := \mathrm{Res}(C_1, C_2)$;
        $S_{i+1} := S_i \cup \{C\}$;
        if ($C = \square$) **then** write "S is *inconsistent*"; **exit;**
            else $i := i + 1$;
        end_if
    until ($S_i = S_{i-1}$) // no new clauses can be derived
    write "S is *consistent*";
end

Theorems

***Soundness* theorem**

If the empty clause is derived from the set S of propositional clauses using the resolution algorithm ($S \vdash_{\text{Res}} \square$), *then* S is an inconsistent set.

***Completeness* theorem**

If the set S of propositional clauses is inconsistent, *then* the empty clause can be derived from S using the resolution algorithm ($S \vdash_{\text{Res}} \square$).

***Soundness and completeness* theorem**

A set S of propositional clauses is inconsistent *if and only if* $S \vdash_{\text{Res}} \square$.

Resolution - a refutation Proof method

Theorem

A propositional formula U is a **theorem** (tautology) *if and only if* the empty clause can be derived from the conjunctive normal form of $\neg U$, using the resolution algorithm.

U is a theorem (tautology) *if and only if* $\mathrm{CNF}(\neg U) \vdash_{\mathrm{Res}} \square$.

Theorem

Let $U_1, U_2, \ldots, U_n, V$ be propositional formulas.

$U_1, U_2, \ldots, U_n \vdash V$ *if and only if*

$U_1, U_2, \ldots, U_n \models V$ *if and only if*

$\mathrm{CNF}(U_1 \wedge U_2 \wedge \ldots \wedge U_n \wedge \neg V) \vdash_{\mathrm{Res}} \square$.

Example 1

Using the resolution method prove that the set S of clauses is inconsistent.

$$S = \{p \vee q,\ \neg p \vee q,\ p \vee \neg q,\ \neg p \vee \neg q\}$$

We denote the propositional clauses as follows:

$$C_1 = p \vee q\,,\quad C_2 = \neg p \vee q\,,\quad C_3 = p \vee \neg q\,,\quad C_4 = \neg p \vee \neg q\,.$$

The process of deriving the empty clause is symbolized by the binary tree:

$$C_1 = \underline{p} \vee q \qquad C_2 = \underline{\neg p} \vee q \qquad C_3 = \underline{p} \vee \neg q \qquad C_4 = \underline{\neg p} \vee \neg q$$

$$C_5 = \mathrm{Res}_p(C_1, C_2) = q \qquad C_6 = \mathrm{Res}_p(C_3, C_4) = \neg q$$

$$C_7 = \mathrm{Res}_q(C_5, C_6) = \square$$

$S \vdash_{\mathrm{Res}} \square$ and according to the *soundness theorem* we conclude that *the set* S ***of propositional clauses is inconsistent.***

Theorem
(based on Davis-Putman procedure)

M.Lupea

A set S of propositional clauses can be simplified, preserving its **consistency/inconsistency**, by applying the following transformations:

- ***Delete the clauses that are tautologies*** because they are not useful in the derivation of the empty clause (from **true** we cannot derive **false**).
- ***Delete the clauses subsumed by other clauses of*** S (based on absorption).
  C_1 ***is subsumed by*** C_2 and C_2 ***subsumes*** C_1 if there exists a clause C_3 such that $C_1 = C_2 \vee C_3$.
- ***Delete every clause that contains a pure literal.***
  A ***pure literal*** in the set S of clauses is a literal that appears in a clause of S, but its negation does not appear in any clause of S.
- ***Let*** $C = l$ ***be a unit clause of*** S.
  Delete every clause containing l ***and delete*** $\neg l$ ***from every remaining clause.***

Strategies and Refinements of Resolution

M.Lupea

- **Strategies:**
  - assure that all the possible clauses to be derived are generated
  - try to avoid the derivation of redundant and irrelevant clauses in order to obtain the empty clause.
  - **level-saturation** strategy, **deletion** strategy,
    set-of-support strategy
- **Refinements**
  - make the resolution process more efficient by imposing restrictions on the clashing clauses
  - **lock** resolution, **linear** resolution,
    semantic resolution

Remarks

- All the refinements and strategies of resolution **preserve** the **soundness** and the **completeness** properties.
- All the combinations of these refinements and strategies **preserve** the **soundness** property.
- The **completeness is not preserved in some combinations.**
- *Incompleteness:* the initial set of clauses is **inconsistent, but the empty clause cannot be derived** because there are too many restrictions imposed by those strategies and refinements.

general resolution + *deletion strategy* is **sound** and **complete**

general resolution + *set-of-support strategy* is **sound and complete**

general resolution + *deletion strategy* + *set-of-support strategy* is **sound and complete**

Strategies of Resolution

M.Lupea

- The ***level saturation strategy*** generates levels of resolvents corresponding to the exploration of the whole search space which contains all the possible resolvents.
- The ***deletion strategy***: the **resolvents that are tautologies or are subsumed by other clauses** in the set S of clauses **are eliminated** and they will not be used further in the resolution process because they produce redundant clauses.

Algorithm: level-saturation-strategy

M.Lupea

input: S **-** the initial set of clauses;
output: message "S is *inconsistent*" or "S is *consistent*"
begin

// we generate the sequence $S^0, S^1, \ldots, S^k$ containing sets of clauses (levels)

$S^0 := S$; $k := 0$; // initial level

do

{ $k := k+1$;

$S^k := \{\mathrm{Res}(C_i, C_j) \mid C_i \in S^{k-1}, C_j \in S^0 \cup S^1 \cup \ldots \cup S^{k-1}\}$;

// we eliminate the resolvents obtained in the current level,
// but which appear already in the previous levels

$S^k := S^k \setminus (S^0 \cup S^1 \cup \ldots \cup S^{k-1})$;

} **until** ($\square \in S^k$ **or** $S^k = \varnothing$).

if ($\square \in S^k$)

then write "S is inconsistent"

else //no more resolvents can be derived and $\square \notin S^k$

write "S is consistent"

end_if
end

Example 2

M.Lupea

Check the consistency/inconsistency of the set $S=\{p\vee q,\ \neg p\vee q\vee\neg r,\ \neg q\vee r\}$.

- level saturation strategy + deletion strategy
- the sequence $S^0, S^1, S^2, \ldots$ represents levels of resolvents

  $S^k := \{\mathrm{Res}(C_i,C_j) \mid C_i \in S^{k-1}, C_j \in S^0 \cup S^1 \cup \ldots \cup S^{k-1}\}, k=1,2,\ldots$
- initial level: $S^0 = S = \{C_1 = p\vee q,\ C_2 = \neg p\vee q\vee\neg r,\ C_3 = \neg q\vee r\}$

First level

$C_4 = \mathrm{Res}_p(C_1,C_2) = q\vee\neg r$

$C_5 = \mathrm{Res}_q(C_1,C_3) = p\vee r$

$C_6 = \mathrm{Res}_q(C_2,C_3) = \neg p\vee\neg r\vee r \equiv T$ (tautology)

$C_7 = \mathrm{Res}_r(C_2,C_3) = \neg p\vee q\vee\neg q \equiv T$ (tautology)

C_6 and C_7 are not included in the first level

$S^1 = \{C_4 = q\vee\neg r, C_5 = p\vee r\}$

Second level

$C_8 = \mathrm{Res}_r(C_4,C_5) = q\vee p = C_1$

$C_9 = \mathrm{Res}_q(C_4,C_3) = r\vee\neg r \equiv T$

$C_{10} = \mathrm{Res}_r(C_4,C_3) = q\vee\neg q \equiv T$

$C_{11} = \mathrm{Res}_p(C_5,C_2) = q\vee r\vee\neg r \equiv T$

$C_{12} = \mathrm{Res}_r(C_5,C_2) = \neg p\vee p\vee q \equiv T$

$S^2 = \varnothing$

$S^2 = \varnothing$ with the meaning that no more resolvents can be generated. *After a complete search, the empty clause was not derived from S and thus S* ***is a consistent set.***

Strategies of Resolution (contd.)

M.Lupea

- The ***set-of-support strategy*** **avoids resolving two clauses belonging to a consistent subset** of the initial set of clauses, because the resolvents derived from a consistent set are irrelevant in the process of deriving □ (inconsistency).

Definition:

Let S be a set of clauses. A subset Y of S is called the ***support set of*** S, if the set $S \setminus Y$ is consistent. The ***set-of-support resolution*** is the resolution of two clauses that are not both from the set $S \setminus Y$. The resolvents generated during the resolution process are added to Y.

Example 3. Modeling reasoning

M.Lupea

$H_1, H_2, H_3 \vdash^{?} C$	$\mathrm{CNF}(H_1 \wedge H_2 \wedge H_3 \wedge \neg C) \vdash_{\mathrm{Res}}^{?} \square$
H_1. If it is sunny, Diane and Alice go to the swimming pool.	$H_1 : S \rightarrow D \wedge A \equiv \neg S \vee (D \wedge A) \equiv$ $\equiv (\neg S \vee D) \wedge (\neg S \vee A) : C_1 \wedge C_2$
H_2. Ben goes to the swimming pool on Thursdays.	$H_2 : Th \rightarrow B \equiv \neg Th \vee B : C_3$
H_3. It was sunny last Thursday.	$H_3 : S \wedge Th : C_4 \wedge C_5$
C. Did Diane meet Ben at the swimming pool last Thursday?	$C : Th \wedge D \wedge B$ $\neg C : \neg(Th \wedge D \wedge B) \equiv \neg Th \vee \neg D \vee \neg B : C_6$ $X = \{C_1, C_2, C_3, C_4, C_5, C_6\}$
Resolution process for the set of clauses X: the set-of- support strategy is applied (see lecture)	
We proved that $\mathrm{CNF}(H_1 \wedge H_2 \wedge H_3 \wedge \neg C) \vdash_{\mathrm{Res}} \square$, so C is deducible from the hypotheses, therefore: "*Diane met Ben at the swimming pool last Thursday*".	

Example 3. the set-of-support strategy

M.Lupea

$X = \{C_1, C_2, C_3, C_4, C_5, C_6\}$ $C_1 = \neg S \vee D$ $C_2 = \neg S \vee A$ $C_3 = \neg Th \vee B$ $C_4 = S$ $C_5 = Th$ $C_6 = \neg Th \vee \neg D \vee \neg B$ $X' = \{C_1, C_3, C_4, C_5, C_6\}$	• The clause C_2 is eliminated because A is a pure literal in the set X and we obtain X'. • The support set of X' is $Y = \{C_6\}$ and corresponds to the negation of the conclusion. • The set $X' \setminus Y = \{C_1, C_3, C_4, C_5\}$ is consistent and contains the clauses provided by the hypotheses. • In the set-of-support strategy, we do not resolve two clauses belonging to the consistent subset of clauses $X' \setminus Y = \{C_1, C_3, C_4, C_5\}$. • In each resolution step at least one parent clause belongs to Y, and the resolvents are added to Y.

A refutation from X' (the derivation of $\square$ from X') is provided below:

$C_7 = \text{Res}(C_5, C_6) = \neg D \vee \neg B,$	$Y = \{C_6, C_7\}$
$C_8 = \text{Res}(C_7, C_3) = \neg D \vee \neg Th$	$Y = \{C_6, C_7, C_8\}$
$C_9 = \text{Res}(C_8, C_5) = \neg D$	$Y = \{C_6, C_7, C_8, C_9\}$
$C_{10} = \text{Res}(C_9, C_1) = \neg S$	$Y = \{C_6, C_7, C_8, C_9, C_{10}\}$
$C_{11} = \text{Res}(C_{10}, C_4) = \square$	

$\text{CNF}(H_1 \wedge H_2 \wedge H_3 \wedge \neg C) \vdash_{\text{Res}} \square$, so C is deducible from the hypotheses, therefore:

"Diane met Ben at the swimming pool last Thursday".

Lock Resolution

M.Lupea

- This refinement of resolution was introduced by R.S. Boyer in 1971.
- It is very efficient and easily to be implemented.
- Each occurrence of a literal from a set of clauses is arbitrarily indexed with an integer.
- ***Restriction***: the literals resolved upon must have the lowest indices in their clauses.
- The literals from resolvents inherit the indices from their parent clauses.

  If the parent clauses have a common literal, in the resolvent this literal will have the lowest index from the inherited indices.
- At the implementation level we must combine lock resolution with the level saturation strategy in order to check all the possible ways of deriving □.

Theorems – Lock resolution

M.Lupea

Soundness:

Let S be a set of clauses having each literal *arbitrarily* indexed with an integer.

If from S the empty clause can be derived using lock resolution ($S \vdash_{\text{Res}}^{lock} \square$),

then S is ***inconsistent***.

Completeness:

Let S be a set of clauses having each literal *arbitrarily* indexed with an integer.

If S is ***inconsistent***, *then* there is a lock derivation of the empty clause from S ($S \vdash_{\text{Res}}^{lock} \square$).

Note: The ***completeness*** property is not preserved if lock resolution is combined with the deletion strategy or with the set-of-support strategy.

Remarks

- In order to prove the **consistency** of a set S of clauses, **lock resolution must be combined with the level saturation strategy**:

  if $S^k = \varnothing$ (the last level of lock resolvents is empty)

  then the set S **is consistent.**

- In order to prove **inconsistency** of a set S of clauses **we do not need to use a strategy**, it is enough to find a lock derivation of $\square$ from S.

  If we combine lock resolution with the level saturation strategy the decision is:

  if $\square \in S^k$ ($\square$ was derived as a lock resolvent in the k level)

  then the set S **is inconsistent.**

Example 4

M.Lupea

Check the consistency/inconsistency of $S=\{\neg r, p\vee\neg q, r\vee\neg p, q\}$

Indexing of the literals:	Another indexing of the literals from clauses
$C_1={}_{(1)}\neg r,\ C_2={}_{(3)}p\vee{}_{(2)}\neg q,\ C_3={}_{(4)}r\vee{}_{(5)}\neg p,\ C_4={}_{(6)}q$	$C_1={}_{(1)}\neg r,\ C_2={}_{(2)}p\vee{}_{(3)}\neg q,\ C_3={}_{(5)}r\vee{}_{(4)}\neg p,\ C_4={}_{(6)}q$
• lock resolution + level saturation strategy	• lock resolution • no strategy
$S^0=S$	The derivation of the empty clause:
$S^1=\{\mathrm{Res}^{lock}(C_i,C_j) \mid C_i\in S^0, C_j\in S^0\}$	$C'_5=\mathrm{Res}_p^{lock}(C_2,C_3)={}_{(3)}\neg q\vee{}_{(5)}r$
$C_5=\mathrm{Res}_r^{lock}(C_1,C_3)={}_{(5)}\neg p$	$C'_6=\mathrm{Res}_q^{lock}(C_4,C'_5)={}_{(5)}r$
$C_6=\mathrm{Res}_q^{lock}(C_2,C_4)={}_{(3)}p$	$C'_7=\mathrm{Res}_r^{lock}(C_1,C'_6)=\square.$
$S^1=\{C_5,C_6\}$	
$S^2=\{\mathrm{Res}^{lock}(C_i,C_j) \mid C_i\in S^1, C_j\in S^0\cup S^1\}$	
$C_7=\mathrm{Res}_p^{lock}(C_5,C_6)=\square$	
$S^2=\{C_7\}$, $\square\in S^2$ => S is ***inconsistent***	

Example 5

M.Lupea

Using lock resolution and the level saturation strategy check the consistency/inconsistency of the following set of clauses:

$$S = \{p \vee q, p \vee \neg q \vee \neg r, \neg p \vee r\}.$$

The literals from the clauses are indexed as follows:

$C_1 = {}_{(1)}p \vee {}_{(2)}q$,

$C_2 = {}_{(3)}p \vee {}_{(4)}\neg q \vee {}_{(5)}\neg r$,

$C_3 = {}_{(7)}\neg p \vee {}_{(6)}r$,

$S^0 = S = \{C_1, C_2, C_3\}$ - the initial set of clauses

No lock resolvents can be generated ($S^1 = \varnothing$), the empty clause cannot be derived from S and we conclude that *the set S is consistent*.

Examples (Homework)

M.Lupea

Example:

Prove that ***lock resolution + the deletion strategy is not complete***

using the inconsistent set $S = \{p \vee q,\ \neg p \vee q,\ p \vee \neg q,\ \neg p \vee \neg q\}$ and the indexing:

$$C_1 = {}_{(2)}p \vee {}_{(1)}q,\quad C_2 = {}_{(3)}\neg p \vee {}_{(4)}q,\quad C_3 = {}_{(5)}p \vee {}_{(6)}\neg q,\quad C_4 = {}_{(8)}\neg p \vee {}_{(7)}\neg q$$

Example:

Prove that ***lock resolution + the set-of-support strategy is not complete***

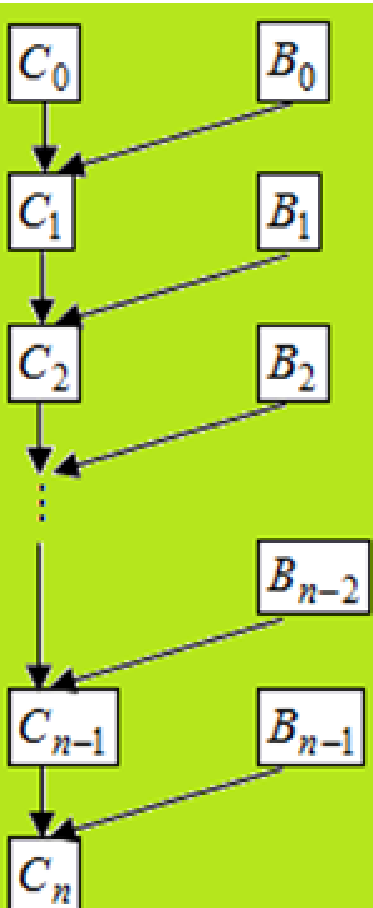
using the deduction:

$$p \to (q \to r),\ r \wedge s \to t,\ u \to s \wedge \neg t \vdash p \wedge q \to \neg u.$$

Linear Resolution (Loveland 1971)

M.Lupea

For a set S of clauses, a ***linear deduction*** of C_n from S with $C_0 \in S$ as the top clause is symbolized graphically as follows:



C_0 is the ***top clause***

$C_1, C_2, \ldots C_{n-1}, C_n$ are ***central clauses***

$B_0, B_1, \ldots, B_{n-1}$ are ***side clauses***

$\forall i = 1, \ldots, n : C_i = \mathrm{Res}(C_{i-1}, B_{i-1})$

$\forall i = 0, \ldots, n-1 : B_i \in S \cup \{C_0, C_1, \ldots, C_{i-2}\}$

Theoretical results

<u>Soundness and completeness theorem</u>

The set S of clauses is **inconsistent** *if and only if* $S \vdash_{\text{Res}}^{lin} \square$.

linear resolution + deletion strategy is complete.

This refinement of resolution provides a **strategy** at the implementation level:

- **backtracking** algorithm.

The algorithm stops in two cases:

- the empty clause was derived and the conclusion is that S is inconsistent.
- for the top clause all the possible side clauses were used, but the empty clause was not derived, then we conclude that the set S is consistent.

The consistency of a set of clauses is proved after a complete search without the derivation of the empty clause.

Special cases of linear resolution

M.Lupea

- ***unit resolution***:

  - the central clauses have at least a unit clause as a parent clause.

- ***input resolution***:

  - all side clauses are initial (input) clauses.

Theorem **(equivalence of *unit resolution* and *input resolution*)**

Let S be a set of propositional clauses. $S \vdash_{\text{Res}}^{input} \square$ ***if and only if*** $S \vdash_{\text{Res}}^{unit} \square$.

These two refinements of resolution are **sound, but** they are **not complete**:

- **soundness**: If $S \vdash_{\text{Res}}^{input/unit} \square$ then S is inconsistent;
- **incompleteness**: there exist inconsistent sets of clauses from which the empty clause cannot be derived using input or unit resolution.

Theoretical results

Definitions:

A clause is called a *positive clause* if it contains only positive literals.

A clause is called a *negative clause* if it contains only negative literals.

A clause is called *Horn clause* if it contains exactly one positive literal, all the other literals are negative.

Theorem:

The input resolution is complete on a set of Horn clauses with a negative top clause. (PROLOG).

Theoretical results

M.Lupea

Knowledge base

From the hypotheses:

$H_1: U_1 \wedge U_2 \wedge ... \wedge U_n \to V$

$H_2: X_1 \wedge X_2 \wedge ... \wedge X_l \to Y$

...

$H_j: W_1 \wedge W_2 \wedge ... \wedge W_m \to R$

is the conclusion $C = Z_1 \wedge Z_2 \wedge ... \wedge Z_m$

deducible?

A formula of type: $U_1 \wedge U_2 \wedge ... \wedge U_n \to V$ **provides**

a Horn clause: $\neg U_1 \vee \neg U_2 \vee ... \vee \neg U_n \vee V$

The hypothesis H_i **provides the Horn clause**

$$C_i, i = 1,2,...,j.$$

The negation of C **is** $\neg(Z_1 \wedge Z_2 \wedge ... \wedge Z_m)$

and provides a negative clause:

$$C_{j+1} = \neg Z_1 \vee \neg Z_2 \vee ... \vee \neg Z_m$$

We have to apply the input resolution to the set $S = \{C_1, C_2, ..., C_j, C_{j+1}\}$ **with the top clause** C_{j+1}**.**

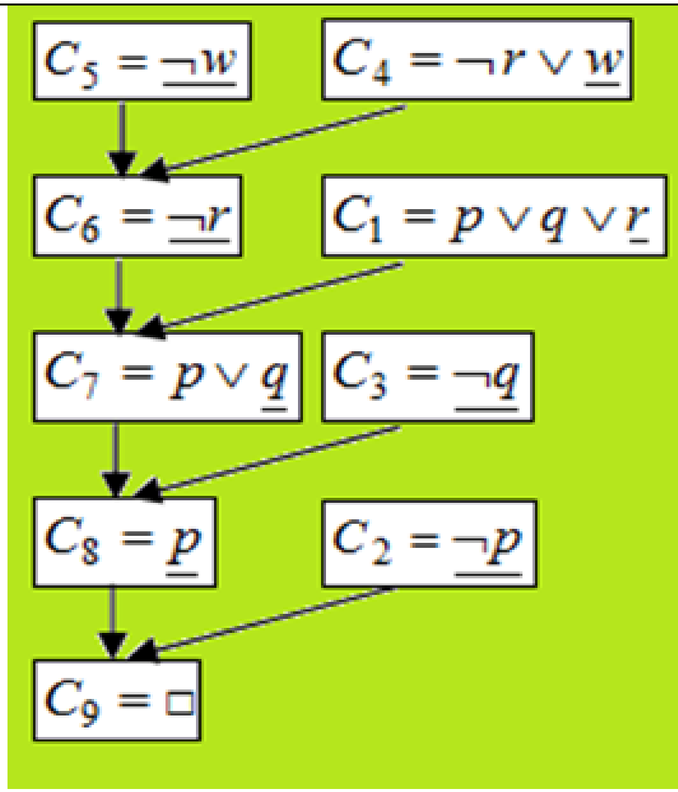
$$H_1, H_2, ..., H_j \vdash C \quad \textbf{if and only if} \quad S \vdash_{Res}^{input} \Box$$

Example 6

M.Lupea

Prove the inconsistency of the set S of clauses: $S = \{C_1, C_2, C_3, C_4, C_5\}$.

$C_1 = p \vee q \vee r$, $C_2 = \neg p$, $C_3 = \neg q$, $C_4 = \neg r \vee w$, $C_5 = \neg w$



- This linear refutation is also a **unit** and an **input** refutation

- $S \vdash^{lin}_{Res} \square$, therefore S is an ***inconsistent set***.

Example 7. Check the consistency/inconsistency of $S=\{C_1=p\vee q, C_2=\neg p\vee q, C_3=\neg p\vee\neg q\}$

The linear search of the derivation of □ using the backtracking strategy:

C_1 : **top clause**

I) C_2 **is used as a side clause for** C_1

$C_4=\text{Res}_p(C_1,C_2)=q$

$C_5=\text{Res}_q(C_4,C_3)=\neg p$

$C_6=\text{Res}_p(C_5,C_1)=q=C_4$. Process blocked.

II) C_3 **is used as a side clause for** C_1

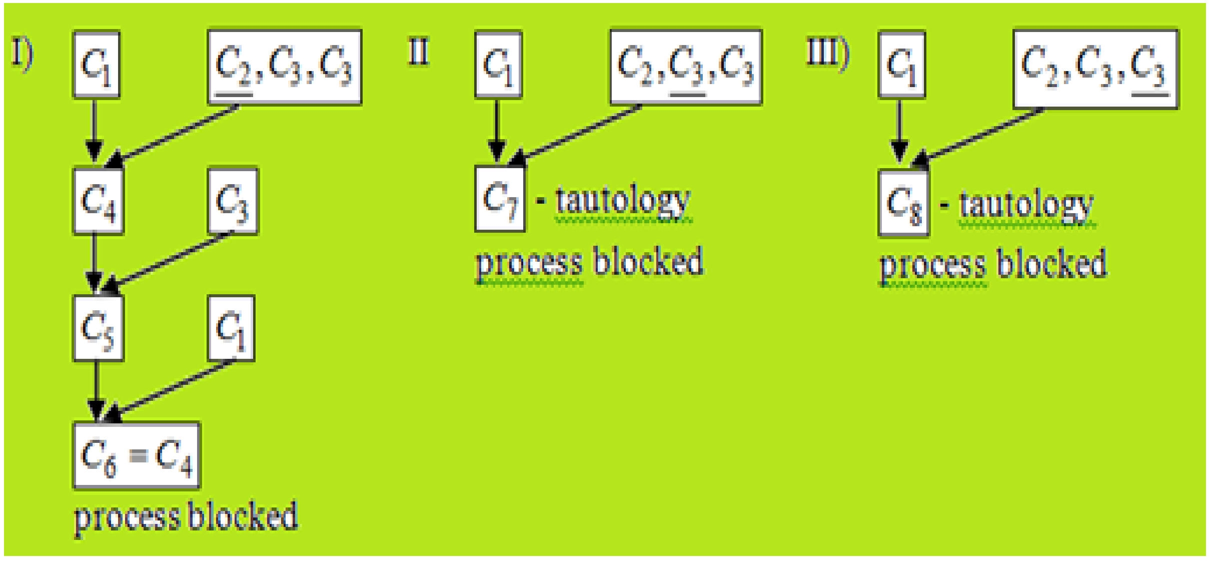
$C_7=\text{Res}_p(C_1,C_3)=q\vee\neg q\equiv T$ (tautology)

The search process is blocked.

III) C_3 **is used as a side clause for** C_1

$C_8=\text{Res}_q(C_1,C_3)=p\vee\neg p\equiv T$ (tautology)

The search process is blocked.



A complete linear derivation search was performed, but □ was not derived, so S is a **consistent** set.

Example 8. Modeling reasoning

M.Lupea

By applying linear resolution prove that the conclusion is derivable from the set of hypotheses.

Hypotheses:

H1. It is not sunny this afternoon and it is colder than yesterday.

H2. We will go swimming only if it is sunny.

H3. If we do not go swimming, then we will take a canoe trip.

H4. If we take a canoe trip, then we will be home by sunset.

Conclusion:

C. We will be home by sunset.