

Resolution

History

Resolution is a proof system for DNF formulas or a refutational Systems for CNF formulas

It was introduced by **Blake in 1937** and then became important by a work **Davis-Putnam** and Robinson in the 60s in the field of automated theorem Proving.

In the last 20 years it was subject of deep investigations in the field of **Proof Complexity**. There are hundreds of papers with subject Resolution.

At present it is still matter of strong investigations

Plan

- Definition of the Resolution system
- Soundness and Completeness
- Examples
- Restrictions and Refinements of Resolution
- Complexity measures for Resolution
- Interpolation for Resolution
- Search Problems and Resolution.
- DPLL algorithm and Treelike Resolution

Definitions

Resolution rule

Clauses are disjunctions of literals ($x_1 \vee x_2 \vee \neg x_3$).

CNF are conjunctions of clauses

Resolution Rule

$$\frac{C \vee x_i \quad D \vee \neg x_i}{C \vee D}$$

We can assume that both x_i and $\neg x_i$ do not occur in C and D .

x_i is the **resolved** variables

Assignments

An assignment satisfies a clause if satisfies at least one of its literals

Property [Exercise 1]

Resolution rule is sound: if an assignment satisfies the premises of the rule then it satisfies the conclusion

Resolution refutations

Let F be a CNF, $F = C_1, \dots, C_m$.

A **Resolution proof** P of a clause C from F $F \xrightarrow[P]{Res} C$

is a sequence of

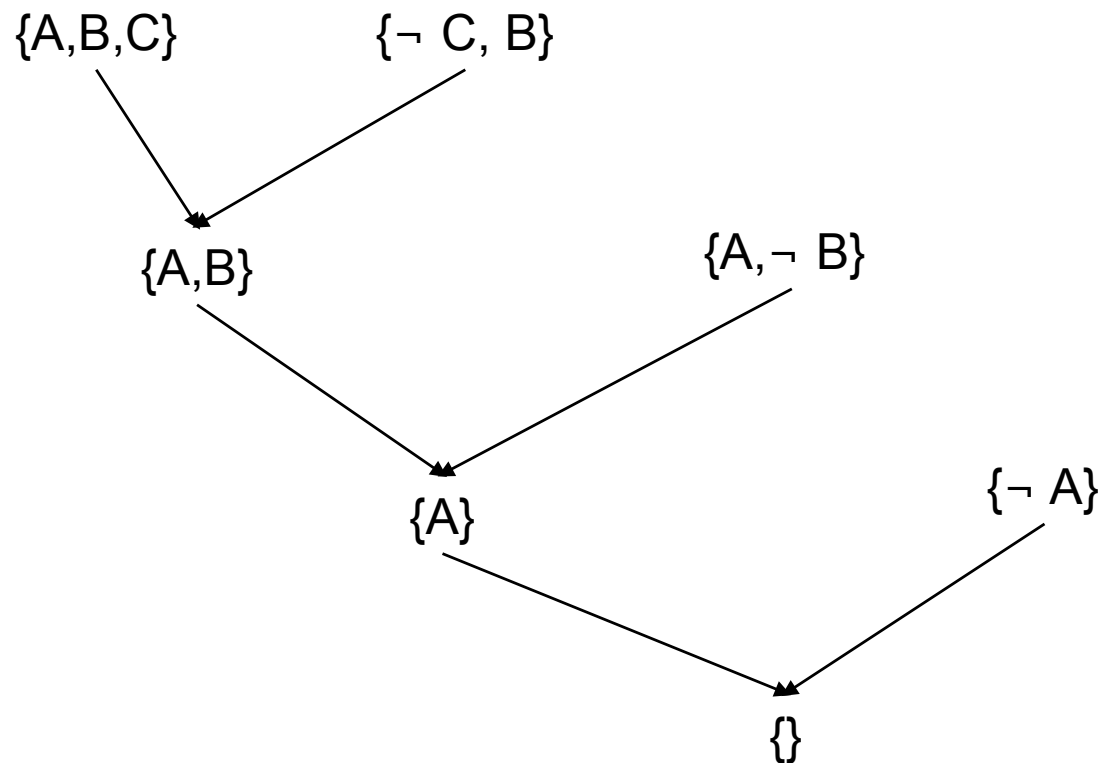
clauses D_1, \dots, D_l s.t.

1. $D_l = C$
2. D_i is either one of the C_i 's or is inferred by Resolution rule from two previous clauses D_j and D_k , $j, k < i$ in the sequence

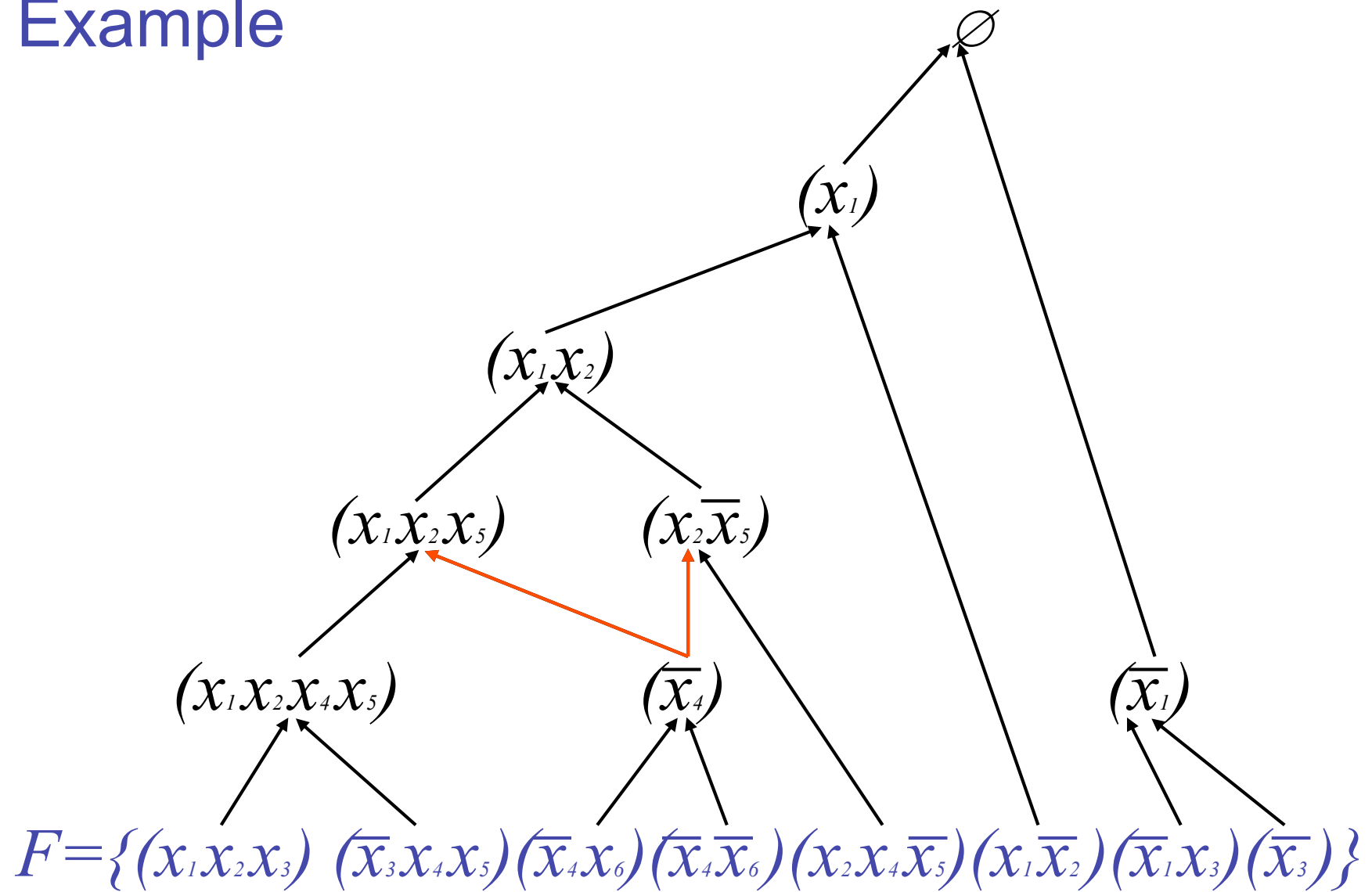
If $C = []$ the empty clause, we speak of **Refutation** of F

Examples

$F = \{A, B, C\}, \{\neg C, B\} \{A, \neg B\} \{\neg A\}$



Example

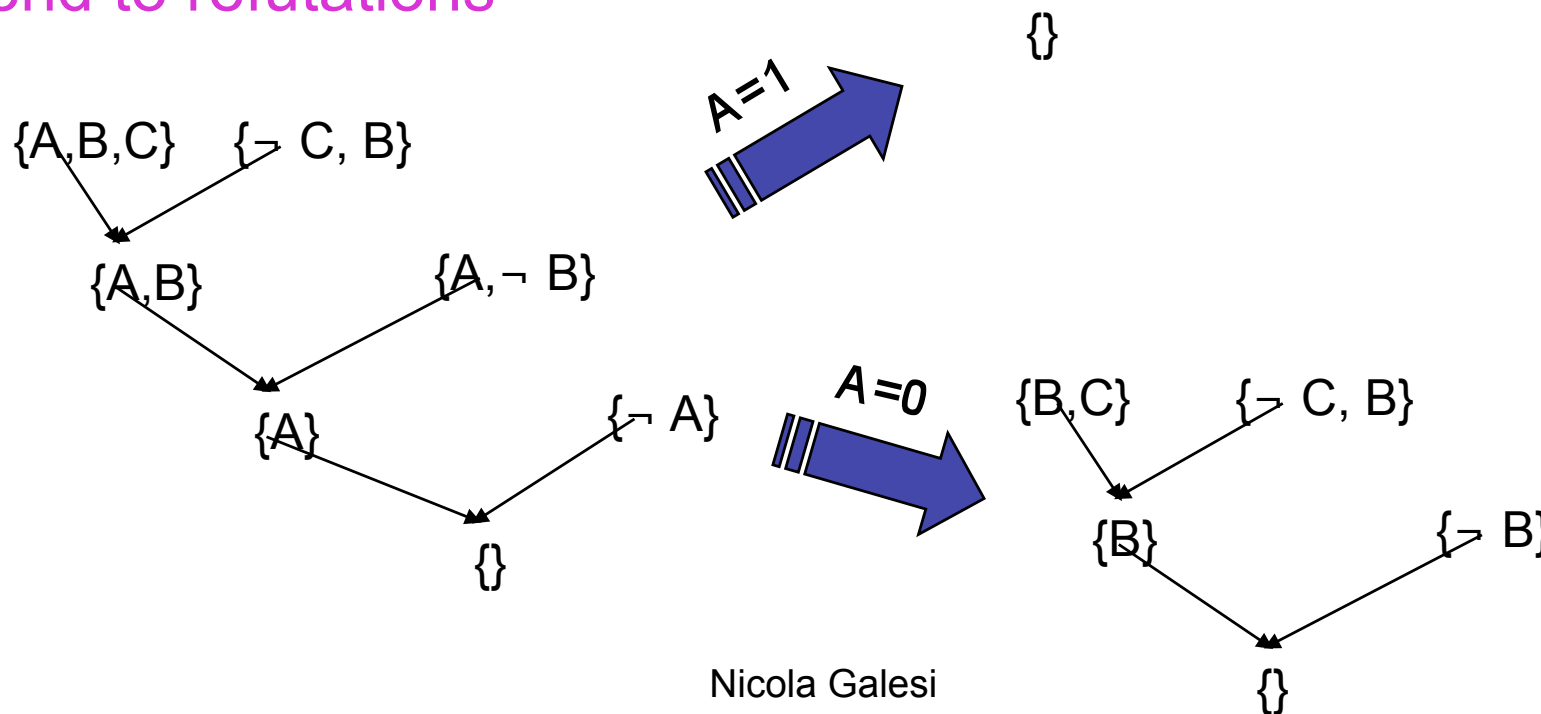


Assignments and refutations

Let C be a clause and α a partial assignment to variables of C . $C[\alpha]$ acts as follows:

- if some variables of C is set to 1 then $C = 1$
- All variables set to 0 are deleted from C

Extend to refutations



A method for UNSAT

Resolution is a method to prove unsatisfiability of formulas in CNF. If a proof of F in Resolution ends with the $[\]$ then F is UNSAT.

Soundness

If $F \xrightarrow[P]{Res} [\]$ then F is UNSAT.

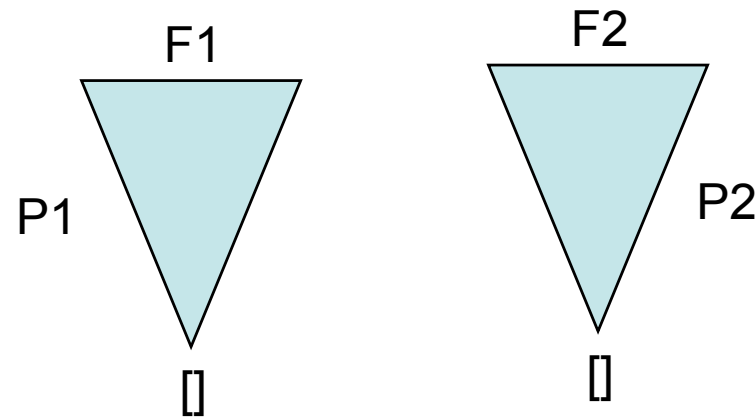
Assume by the contrary that F is SAT. Then there exists an assignment which satisfies the whole proof, in particular $[\]$.

A method for UNSAT

Completeness. Induction on $n = \#$ of variables of F
 $n=0$. Then $F = []$. Ok

$n \rightarrow n+1$. Choose a var x in F

$F1 = F[x=1]$ and $F2 = F[x=0]$. $F1$ and $F2$ are UNSAT, then by HI



A method for UNSAT

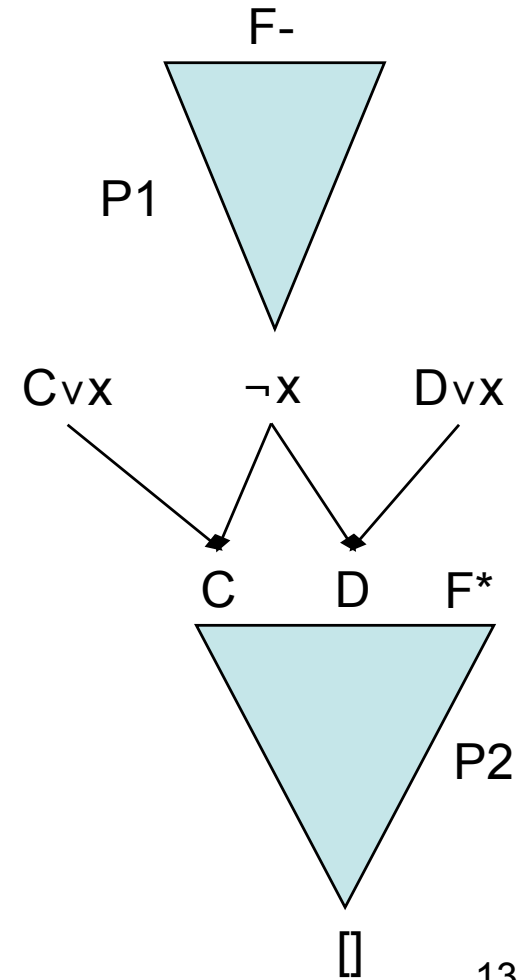
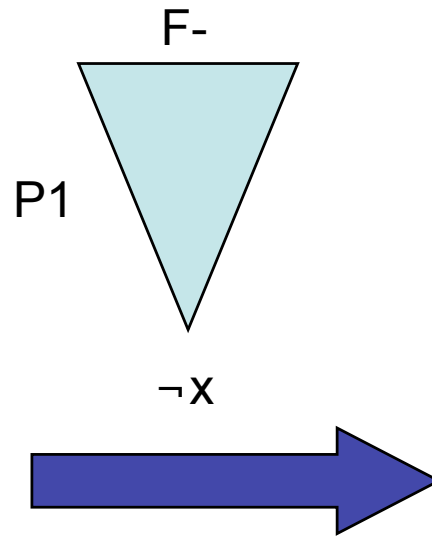
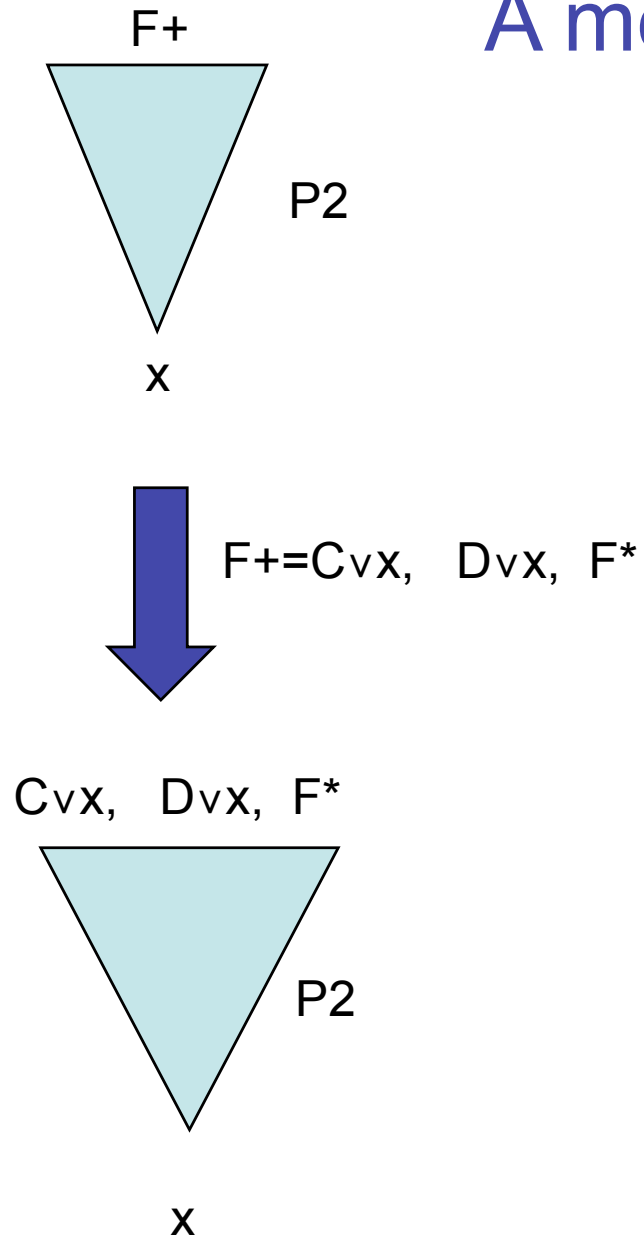
$F+$ obtained from F as follows

- keep all clauses containing x
- Delete all clauses containing $\neg x$
- Keep all other clauses

$F-$ obtained from F as follows

- keep all clauses containing $\neg x$
- Delete all the clauses containing x
- Keep all other clauses

A method for UNSAT



Resolution as an algorithm

Assume S is a set of clauses.

$\text{Res}(S) = S \cup \{C \mid C \text{ is obtained by Resolution from } A, B \in S\}$

Define

$$\text{Res}^0(S) = S$$

$$\text{Res}^{n+1}(S) = \text{Res}(\text{Res}^n(S))$$

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

Thm. S is a set of clauses is UNSAT iff $\square \in \text{Res}^*(S)$.

[Exercise 2]

Resolution as an algorithm

An algorithm to test if a formula A is a TAUT

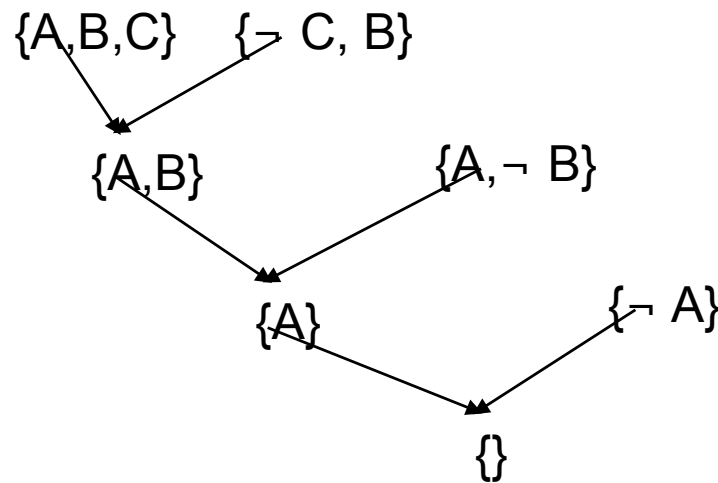
1. Take $\neg A$
2. Transform $\neg A$ in CNF formula S
3. Repeat
4. $F=S$
5. $S= \text{Res}(S)$
6. While ($\square \notin S$ or $F=S$)
7. Output($\square \in S$)

Refinements of Resolution

Treelike Resolution (TLR)

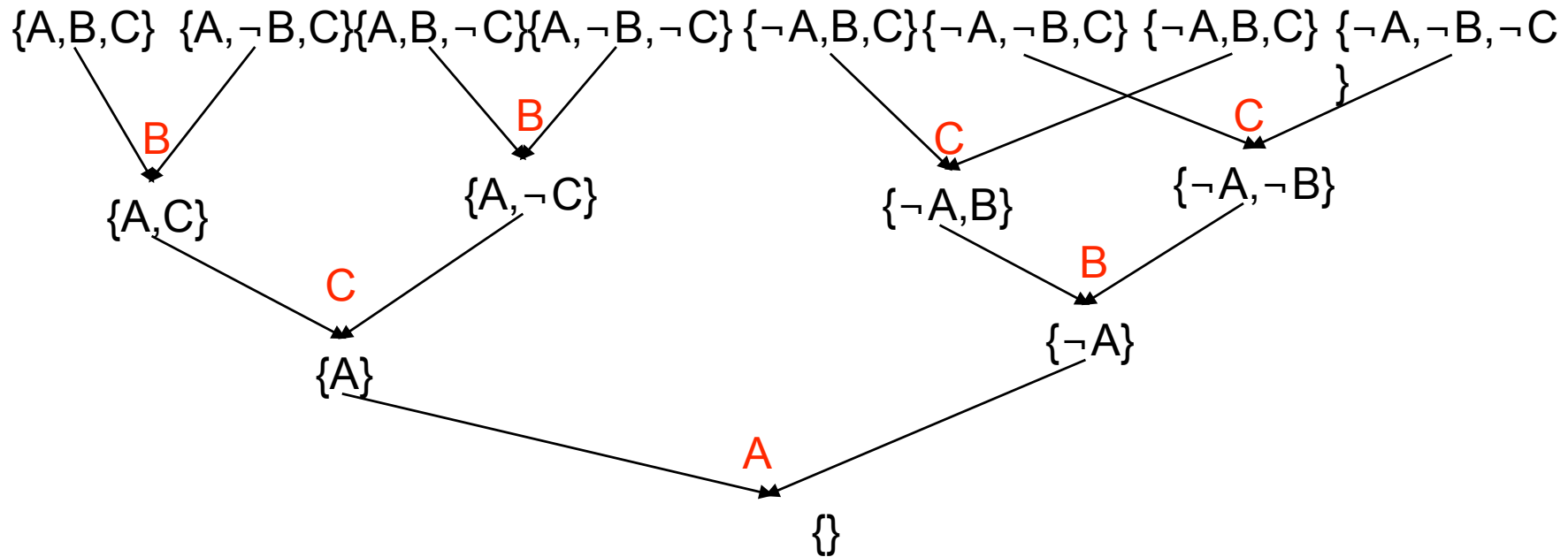
A Resolution refutation P is **treelike** if each clause in the proof is **used at most once** as a **premise** in a resolution rule

Said otherwise: a refutation is treelike if the proof graph is a **tree**.



Regular Resolution

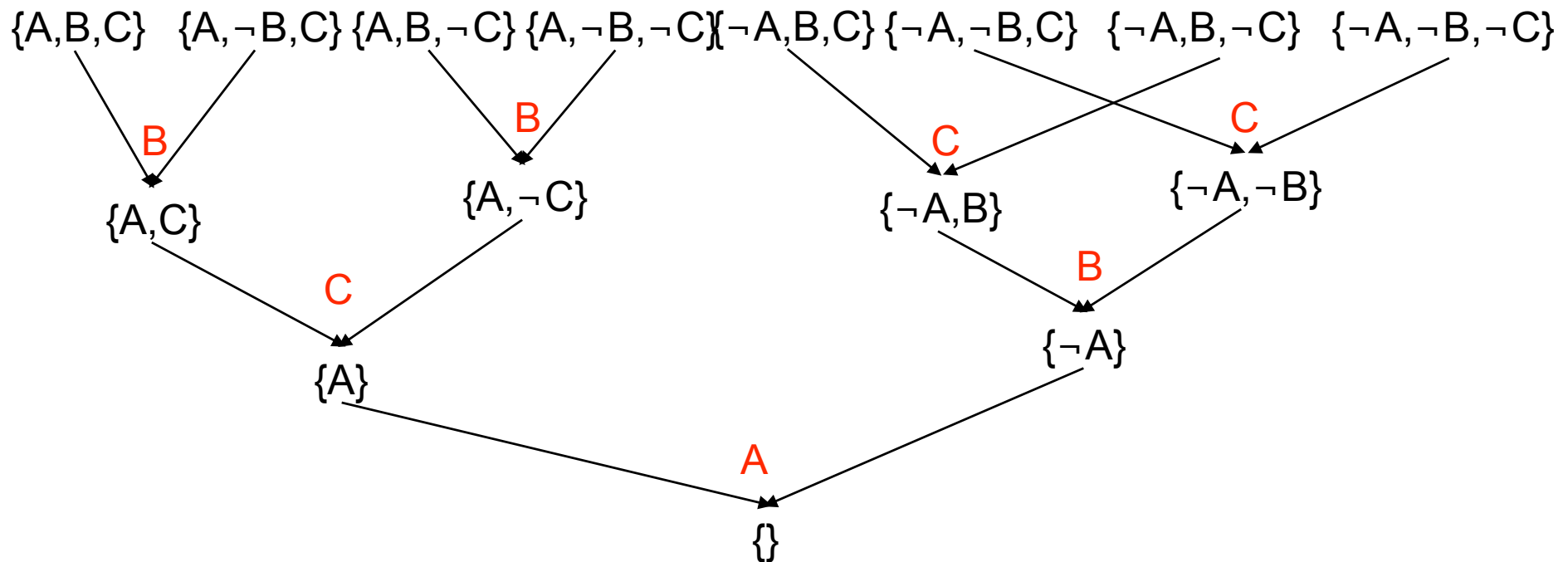
A Resolution refutation P is **regular** if along all paths from the empty clause to a leaf, each variable is resolved at most once.



Regularity important on daglike proofs

Ordered Resolution

A Resolution refutation P of is **Ordered** if there is an elimination order of the variables which is respected along all Paths.

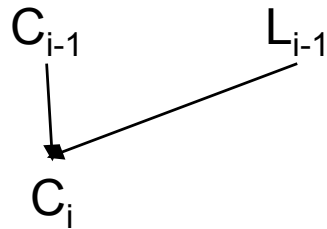


Ordered is a case of Regular [Transform the proof in the example in ordered].

Ordered Resolution important on daglike proofs

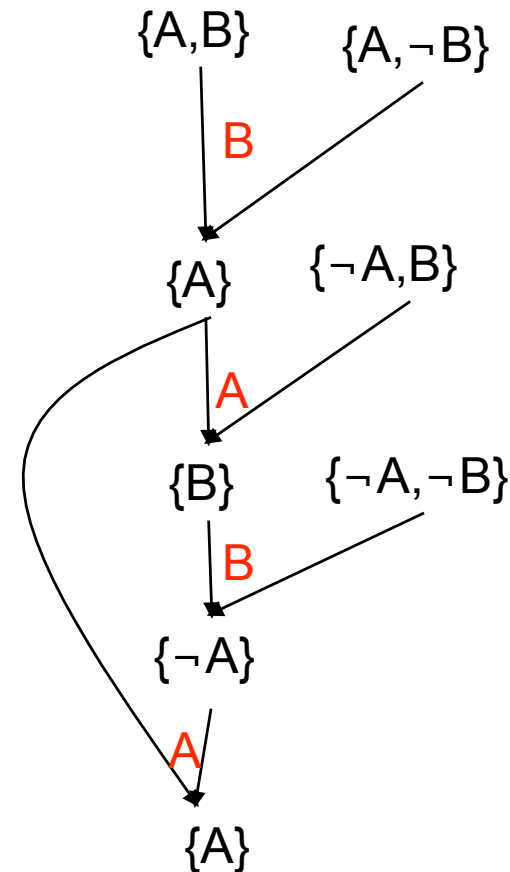
Linear Resolution

A Linear Resolution refutation of a formula $F = F_1, \dots, F_r$ is a sequence C_1, \dots, C_m s.t. $C_1 \in F$, $C_m = []$ and each step is of the form



Where L_{i-1} is either a clause of F or is C_j for $j < i$

Example



Complexity measures For Resolution

Size

Let P be a refutation $C_1, C_2, \dots, C_m = []$ of a CNF $F = F_1, \dots, F_r$.

The **size** of P is m , i.e. the **number of clauses in the proof** or equivalently the **number of nodes** in the proof graph.

Given a CNF formula F

Size of refuting F in X -Resolution (where $X = \text{daglike, treelike, Regular, ecc}$)

$$S_X(F) = \min\{|P| : P \text{ is a } X\text{-Resolution refutations of } F\}$$

Notice for the same F it could be that $S_{TLR}(F) \geq \exp(|F|^\epsilon)$ but

$$S_{DLR}(F) \leq |F|^{O(1)}$$

Resolution Space

Memory configuration: A set of clauses M

Refutation: $P = M_0, M_1, \dots, M_k$

* M_0 is empty

* M_k contains the empty clause.

* M_{t+1} is obtained from M_t by:

1. Axiom Download: $M_{t+1} = M_t + C \in F.$

2. Inference step: $M_{t+1} = M_t +$ some C derived by resolution from a pair of clauses in $M_t.$

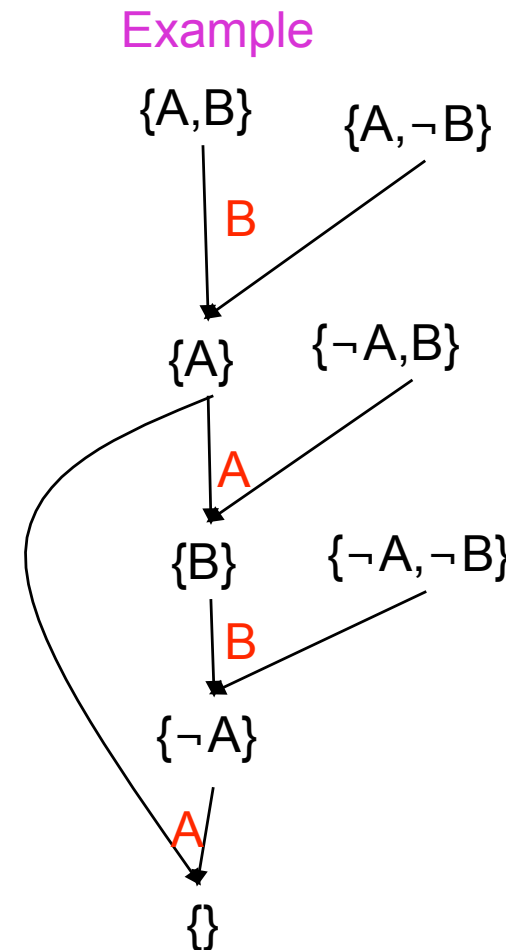
3. Memory Erasure: M_{t+1} is a subset of $M_t.$

$$Sp(P) = \max_{t \in [k]} \{ |M_t| \}.$$

$$Sp_R(F) = \min \{ Sp(P) : P \text{ refutation of } F \}.$$

Resolution Space: Example

Time		Memory		
0				
1	{A,B}			
2	{A,B}	{A,¬B}		
3	{A,B}	{A,¬B}	{A}	
4	{A,¬B}	{A}		
5	{A}			
6	{¬A,¬B}	{A}		
7	{¬A,¬B}	{A}	{B}	
8	{A}	{B}		
9	{A}	{B}	{¬A,¬B}	
10	{A}	{B}	{¬A,¬B}	{¬A}
11	{A}	{B}	{¬A}	
12	{A}	{B}	{¬A}	{}



Resolution width

C a clause. The **width of C**,

$$w(C) = \# \text{ literals in } C$$

F a CNF, the **width of F**

$$w(F) = \max\{w(C) : C \text{ a clause in } F\}$$

P a Refutation of a CNF F, **the width of P**

$$w(P) = \max\{w(C) : C \text{ a clause in } P\}$$

F UNSAT CNF. The **width of refuting F in Resolution**

$$w_R(F) = \min \{w(P) : P \text{ is a Resolution of } F\}$$

Relationships between size and width

[BenSasson, Wigderson 99] Proved in **Chapter II**

Let F be a UNSAT k -CNF defined over n variables

Size-width tradeoffs for TLR

$$w_R(F) \leq \log(S_{TLR}(F)) + k$$

$$S_{TLR}(F) \geq 2^{(w_R(F)-k)}$$

Size-width tradeoffs for DLR

$$w_R(F) \leq O\left(\sqrt{n \log S_{DLR}(F)}\right) + k$$

$$S_{DLR}(F) \geq 2^{\Omega\left(\frac{(w_R(F)-k)^2}{n}\right)}$$

Relationships between space and width

[Atserias-Dalmau 03] Proved in **Chapter VI**

Let F be a UNSAT k -CNF defined over n variables

Space width tradeoffs for Resolution

$$Sp_R(F) \geq w_R(F) - k + 1$$

Interpolation for Resolution

Interpolation and Complexity

Let $A(\mathbf{p},\mathbf{q}) \rightarrow B(\mathbf{p},\mathbf{r})$ be a TAUT formula where \mathbf{q},\mathbf{r} are sets of private variables and \mathbf{p} are commons to the two formulas.

An **Interpolant** $C(\mathbf{p})$ is a formula such that

$$A(\mathbf{p},\mathbf{q}) \rightarrow C(\mathbf{p})$$

$$C(\mathbf{p}) \rightarrow B(\mathbf{p},\mathbf{r}).$$

Interpolant and complexity

[Mundici 82] proved that if the formula size (circuit size) of the interpolant is polynomial in the size of the implication then

$$NP \cap \text{co-NP} \subseteq NC1/\text{poly} \text{ (resp } NP \cap \text{co-NP} \subseteq P/\text{poly})$$

Interpolation and Complexity

[Krajicek 94] Estimate the size of the circuit of the interpolant in terms of the length of the proof for the implicant.

Let $A(\mathbf{p}, \mathbf{q}) \wedge B(\mathbf{p}, \mathbf{r})$ a UNSAT CNF

An **Interpolant** $C(\mathbf{p})$ is a circuit s.t.

$$C(\mathbf{a}) = \begin{cases} 0 & A(\mathbf{a}, \mathbf{q}) \text{ UNSAT} \\ 1 & B(\mathbf{a}, \mathbf{r}) \text{ UNSAT} \end{cases}$$

Interpolation and Complexity

Thm [Krajicek 94, Pudlak 96] **proved in Chapter III**

Let P be a DLR refutations of $A(\mathbf{p},\mathbf{q}) \wedge B(\mathbf{p},\mathbf{r})$. Then there exists a boolean circuit $C(\mathbf{p})$ s.t.

1. for every truth assignment \mathbf{a} to the common variables p

$$C(\mathbf{a}) = \begin{cases} 0 & A(\mathbf{a},\mathbf{q}) \text{ UNSAT} \\ 1 & B(\mathbf{a},\mathbf{r}) \text{ UNSAT} \end{cases}$$

2. C is of size $O(|P|)$ (#gates).

3. If the common variables \mathbf{p} occur only positively in A and negatively in B , then C is monotone

4. If P is TLR, then C is a formula (treelike circuit)

Cor (informal)

Lower bounds on (monotone) circuit size give lower bounds on length of Resolution refutations

Search Problems And Resolution

Branching Program

Branching Programs

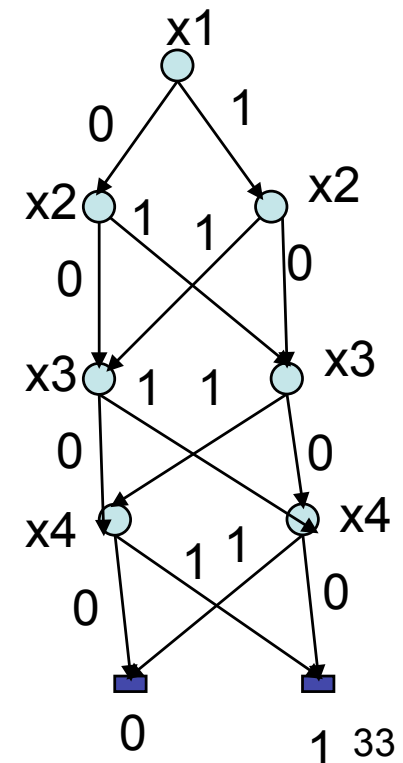
A 2-regular dag with one source node and two target nodes

- Every node labelled with a variable
- The two edges leaving a node are labelled with element of a set X

A branching program **computes a function**

$f: \{0,1\}^n \rightarrow X$ as follows:

$f(a_1, \dots, a_n) = x$ if the path starting at the root and following the edges according to \mathbf{a} ends in x



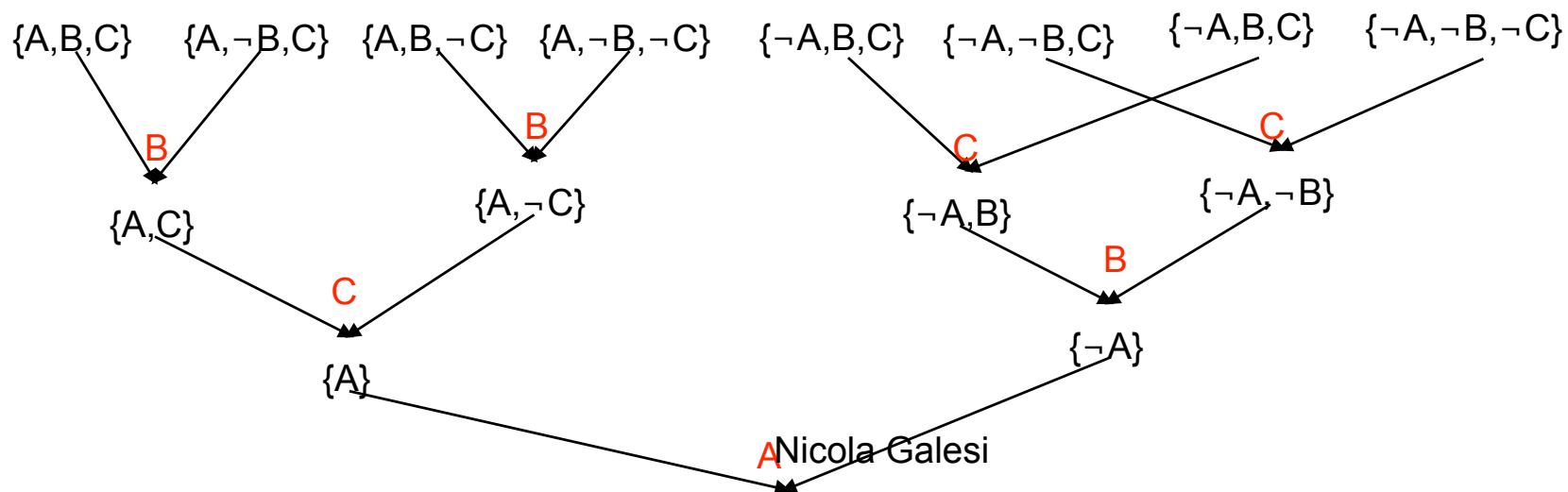
A False Clause Search Problem

Let F be a UNSAT CNF $F = C_1, \dots, C_m$

Search Problem

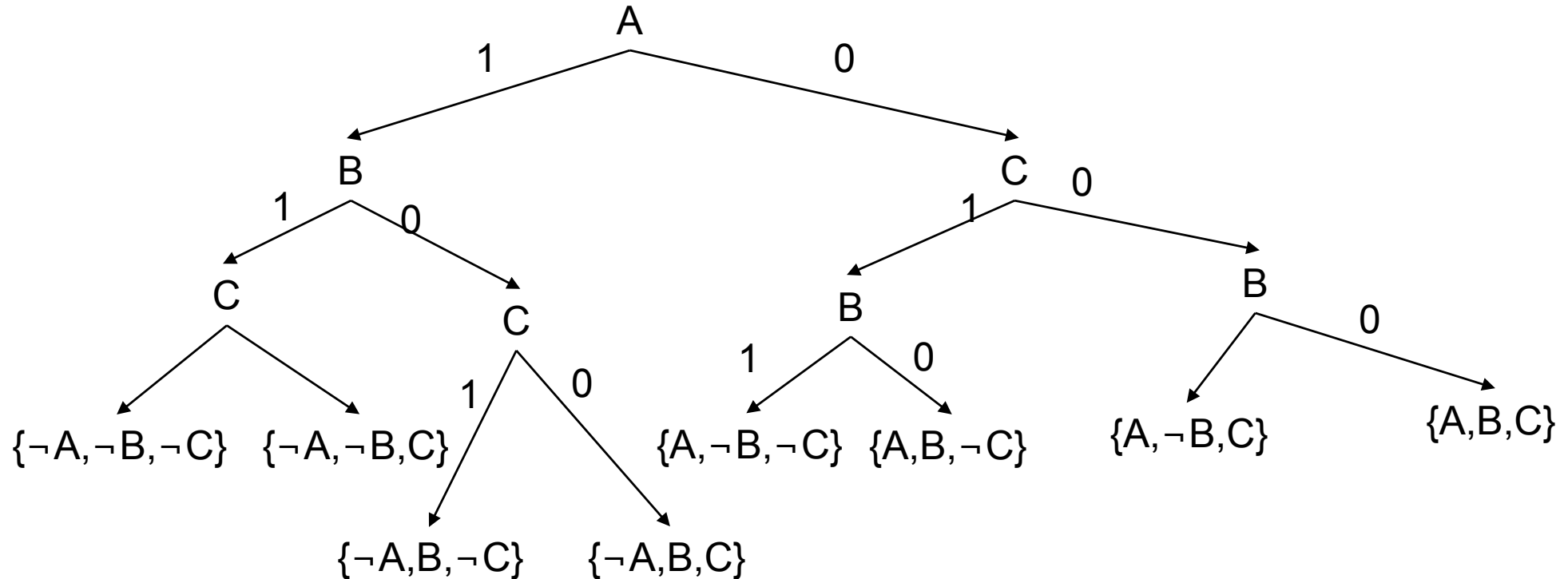
Given an assignment α to variables F find a clause C in F falsified under α

Refutation into decision trees



A False Clause Search Problem

Refutation into decision trees



Thm. A TLR refutation for F defines a decision tree that solves the FCLP for F . [Exercise 5. Make the statement and its proof precise]

Regular resolution define ordered branching programs to solve FCSP

DPLL and treelike Resolution

A SAT algorithm

Let F be a CNF. DPLL is an algorithm for the SAT of F

DPLL (F)

if F is empty

F is satisfiable and report the assignment

If F contains the empty clause

then return

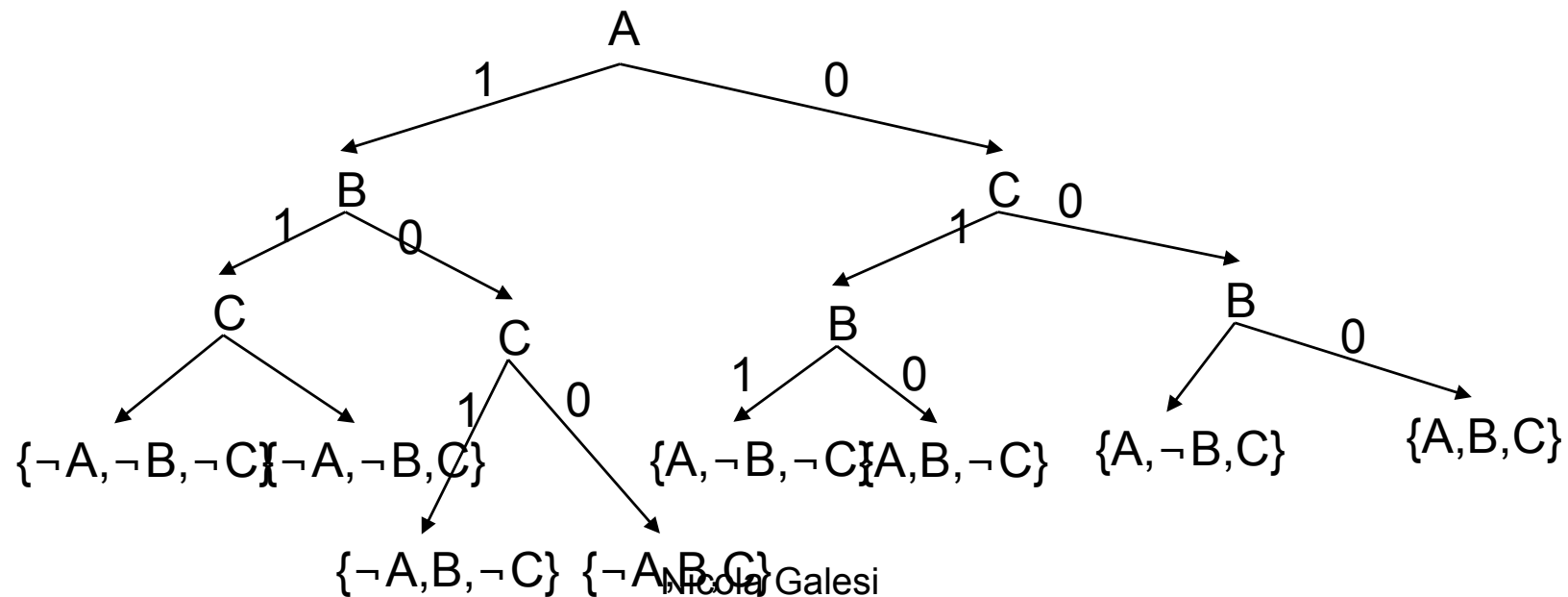
else

- choose a literal x
- DPLL($F[x=1]$)
- DPLL($F[x=0]$)

Search tree on DPLL

Let F be a UNSAT CNF. DPLL is producing a Treelike Resolution refutation of F .

$$F = \{A, B, C\} \{A, \neg B, C\} \{A, B, \neg C\} \{A, \neg B, \neg C\} \\ \{\neg A, B, C\} \{\neg A, \neg B, C\} \{\neg A, B, \neg C\} \{\neg A, \neg B, \neg C\}$$



DPLL and Proof Complexity

DPLL vs TLR refutations.

DPLL is an algorithm to recover TLR refutations on UNSAT CNF formulas. This will have some consequences on Automatizabilty of TLR (Chapter III)

TLR Refutations vs DPLL performances

Since TLR is not polynomially bounded, i.e. There are families of formulas requiring exponential size TRL refutations, then DPPL performs bad on this formula.

DLR and Proof Search

Extending similar considerations to DLR and find good algorithms giving DLR refutations is matter of research