

# *Artificial Intelligence*

## *Automated Symbolic Calculus*

Marco Piastra

# Semantic Tableaux

# Semantic Tableaux, alpha and beta rules

*Semantic tableaux* is a method that can be implemented as a Turing machine

- It is a decision algorithm for the problem  
“is  $\Sigma$  satisfiable?”  
where  $\Sigma$  is a set of wffs in  $L_P$

In spite of its name, it is a **symbolic** method: it works on the structure of wffs only  
No explicit assignments of (semantic) values are involved

# Semantic Tableaux, alpha and beta rules

- A tableau is a set of wffs in  $L_P$

The method starts from an *initial* tableau

(i.e. the set  $\Sigma$  whose satisfiability is to be determined)

It is based on rules that transform each one wff into two wffs

- Alpha rules (i.e. expansion)

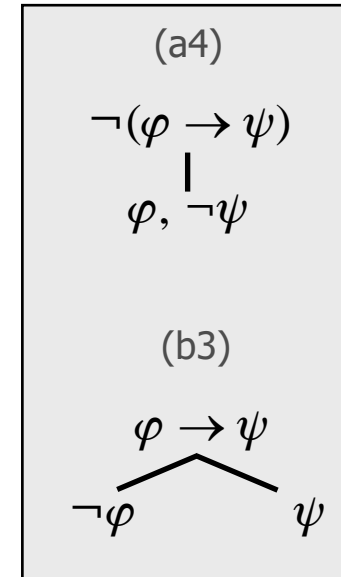
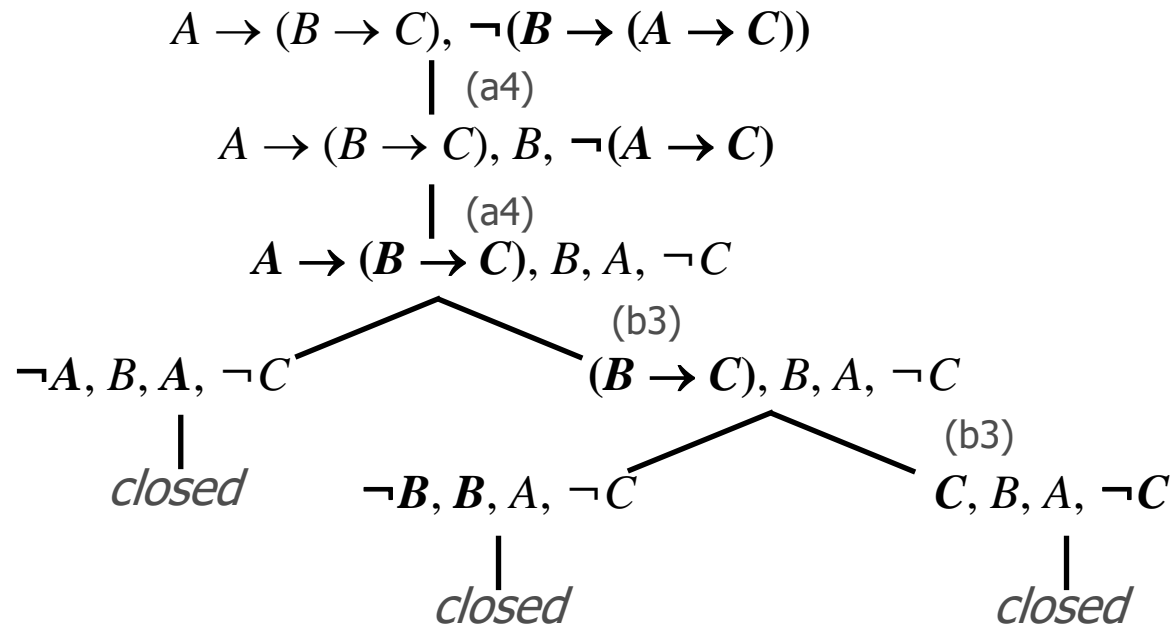
(a1)	(a2)	(a3)	(a4)
$\neg(\neg\varphi)$	$\varphi \wedge \psi$	$\neg(\varphi \vee \psi)$	$\neg(\varphi \rightarrow \psi)$
$\varphi$	$\varphi, \psi$	$\neg\varphi, \neg\psi$	$\varphi, \neg\psi$

- Beta rules (i.e. bifurcation)

(b1)	(b2)	(b3)	(b4)	(b5)
$\varphi \vee \psi$	$\neg(\varphi \wedge \psi)$	$\varphi \rightarrow \psi$	$\varphi \leftrightarrow \psi$	$\neg(\varphi \leftrightarrow \psi)$
$\varphi$ $\psi$	$\neg\varphi$ $\neg\psi$	$\neg\varphi$ $\psi$	$\neg\varphi, \neg\psi$ $\varphi, \psi$	$\neg\varphi, \psi$ $\varphi, \neg\psi$

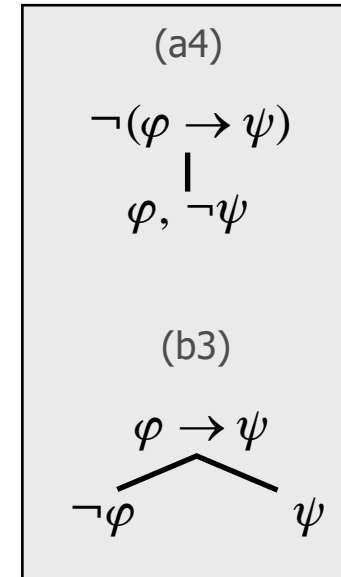
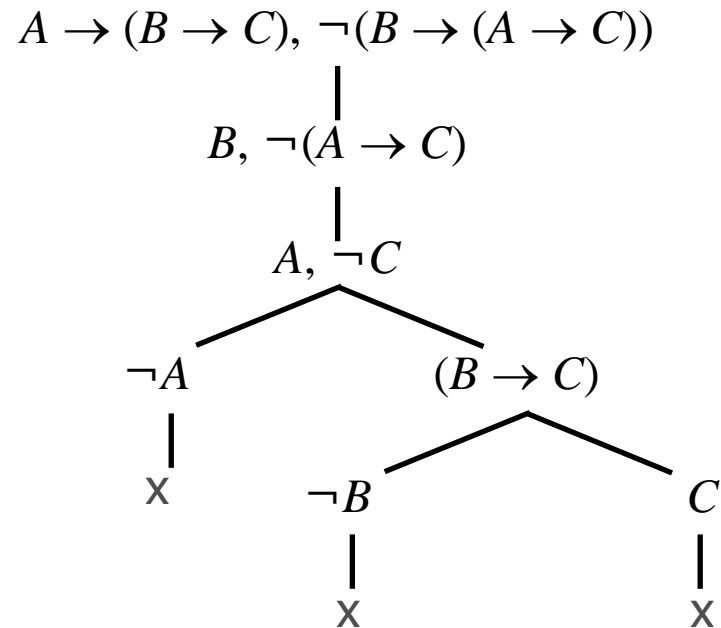
# Semantic Tableaux – a working example

- Original problem: “ $\Gamma \models \varphi$  ?”  
Example input:  $A \rightarrow (B \rightarrow C) \models B \rightarrow (A \rightarrow C)$  ?
- Transformed problem: “is  $\Gamma \cup \{ \neg\varphi \}$  satisfiable?”  
Hence the initial tableau is  $\Gamma \cup \{ \neg\varphi \}$



# Semantic Tableaux – a working example

- Original problem: “ $\Gamma \models \varphi$  ?”  
Example input:  $A \rightarrow (B \rightarrow C) \models B \rightarrow (A \rightarrow C)$  ?
- Transformed problem: “is  $\Gamma \cup \{ \neg\varphi \}$  satisfiable?”  
Hence the initial tableau is  $\Gamma \cup \{ \neg\varphi \}$



The usual notation in textbooks is even more concise:  
only those wffs that are *added* to the initial tableau in each branch are shown in the tree

# Semantic Tableaux – algorithm recap

## ■ Algorithm:

The input problem “ $\Gamma \models \varphi ?$ ” is transformed into “is  $\Gamma \cup \{ \neg \varphi \}$  satisfiable?”

Methods of this type are also called ‘*by refutation*’

Set  $\Gamma \cup \{ \neg \varphi \}$  as the first *active* tableau

For each *active* tableau, there will be two cases:

1) The tableau contains only *literals*

**If** the tableau contains a *complementary pair of literals*

**then** declare it *closed*

**else** declare it *open*

2) The tableau contains one or more *composite* wff

First try to apply an *alpha* rule, generating a new tableau

otherwise, if this is not possible, try to apply a *beta* rule generating two new tableaux

Mark the tableau as *inactive*, mark the new tableau(x) as *active*

Continue until there are no more *active* tableaux

Output: the tree structure of tableaux

Result: either all the leaves in the tree are closed (*success*)

or any of them are open (*failure*)

# Semantic Tableaux – (required) algorithm properties

## ■ Termination

The algorithm never *diverges* (i.e. it never enters an infinite loop)

Each application of either alpha or beta rule *simplifies* a wff (i.e. it makes it *less* composite):  
so the application of rules cannot continue forever

## ■ Symbolic derivation

As already stated, in spite of its name, this is a *symbolic* method

We write

$$\Gamma \vdash_{ST} \varphi$$

iff the *Semantic Tableau* method is successful (i.e. all leaves are *closed*) for  $\Gamma \cup \{\neg\varphi\}$

How do we know that  $\Gamma \vdash_{ST} \varphi \Rightarrow \Gamma \models \varphi$  ?

(*Soundness* - also *correctness* - of the method)

Exercise: prove it

(*hint*: consider the condition on  $\Gamma \cup \{\neg\varphi\}$  and think about how it relates to each *rule*)

How do we know that  $\Gamma \models \varphi \Rightarrow \Gamma \vdash_{ST} \varphi$  ?

(*Completeness* of the method)

Proving it is a bit more difficult: see textbook (i.e. Ben-Ari's book)



# Semantic Tableaux – (required) algorithm properties

- **Termination**

The algorithm never *diverges* (i.e. it never enters an infinite loop)

Each application of either alpha or beta rule *simplifies* a wff (i.e. it makes it *less* composite):  
so the application of rules cannot continue forever

- **Soundness**

$$\Gamma \vdash_{ST} \varphi \Rightarrow \Gamma \models \varphi$$

- **Completeness**

$$\Gamma \models \varphi \Rightarrow \Gamma \vdash_{ST} \varphi$$

- **Termination + Soundness + Completeness = *Decision Algorithm***

(for propositional logic)

# Which method is faster?

- Time complexity (remember, consider the *worst case*)  
The 'brute-force search' and *Semantic Tableaux* have the same complexity:  $O(2^n)$

- *How well do these method perform in practice?*

*It depends*

## **Example 1** (try it):

$$A \wedge B \wedge C \wedge \neg A$$

The 'brute-force search' requires  $2^3 = 8$  attempts

The Semantic Tableau method requires applying the same alpha rule 3 times

## **Example 2** (try it):

$$(A \vee B) \wedge (A \vee \neg B) \wedge (\neg A \vee B) \wedge (\neg A \vee \neg B)$$

The 'brute-force search' requires  $2^2 = 4$  attempts

The Semantic Tableaux method requires applying the same alpha rule 3 times; then the same beta rule is applied exhaustively producing a tree with 4 levels, with each node in a tree with a branching factor 2

At the end, the tree has  $2^4 = 16$  leaves (all *closed* tableau)

# Resolution by Refutation

# Inference rule: Resolution

$$\varphi \vee \chi, \neg\chi \vee \psi \vdash \varphi \vee \psi$$

$\varphi \vee \psi$  is also called the *resolvent* of  $\varphi \vee \chi$  e  $\neg\chi \vee \psi$

The resolution rule is *correct*

$$\text{In fact } \varphi \vee \chi, \neg\chi \vee \psi \vdash \varphi \vee \psi \Rightarrow \varphi \vee \chi, \neg\chi \vee \psi \models \varphi \vee \psi$$

$\varphi$	$\psi$	$\chi$	$\varphi \vee \chi$	$\neg\chi \vee \psi$	$\varphi \vee \psi$
0	0	0	0	1	0
0	0	1	1	0	0
0	1	0	0	1	1
0	1	1	1	1	1
1	0	0	1	1	1
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	1

# Normal forms

= translation of each wff into an equivalent wff having a specific structure

## ▪ **Conjunctive Normal Form (CNF)**

A wff with a structure

$$\alpha_1 \wedge \alpha_2 \wedge \dots \wedge \alpha_n$$

where each  $\alpha_i$  has a structure

$$(\beta_1 \vee \beta_2 \vee \dots \vee \beta_n)$$

where each  $\beta_j$  is a *literal* (i.e. an atomic symbol or the negation of an atomic symbol)

Examples:

$$(B \vee D) \wedge (A \vee \neg C) \wedge C$$

$$(B \vee \neg A \vee \neg C) \wedge (\neg D \vee \neg A \vee \neg C)$$

## ▪ **Disjunctive Normal Form (DNF)**

A wff with a structure

$$\beta_1 \vee \beta_2 \vee \dots \vee \beta_n$$

where each  $\beta_i$  has a structure

$$(\alpha_1 \wedge \alpha_2 \wedge \dots \wedge \alpha_n)$$

where each  $\alpha_j$  is a *literal*

# Conjunctive Normal Form

- Translation into CNF (it can be automated)

Exhaustive application of the following rules:

1) Rewrite  $\rightarrow$  and  $\leftrightarrow$  using  $\wedge$ ,  $\vee$ ,  $\neg$

2) Move  $\neg$  inside composite formulae

“De Morgan laws”:

$$\neg(\varphi \wedge \psi) \equiv (\neg\varphi \vee \neg\psi)$$
$$\neg(\varphi \vee \psi) \equiv (\neg\varphi \wedge \neg\psi)$$

3) Eliminate double negations:  $\neg\neg$

4) Distribute  $\vee$

$$((\varphi \wedge \psi) \vee \chi) \equiv ((\varphi \vee \chi) \wedge (\psi \vee \chi))$$

## Examples:

$$\begin{aligned} &(\neg B \rightarrow D) \vee \neg(A \wedge C) \\ &B \vee D \vee \neg(A \wedge C) && \text{(rewrite } \rightarrow \text{)} \\ &B \vee D \vee \neg A \vee \neg C && \text{(De Morgan)} \end{aligned}$$

$$\begin{aligned} &\neg(B \rightarrow D) \vee \neg(A \wedge C) \\ &\neg(\neg B \vee D) \vee \neg(A \wedge C) && \text{(rewrite } \rightarrow \text{)} \\ &(B \wedge \neg D) \vee (\neg A \vee \neg C) && \text{(De Morgan)} \\ &(B \vee \neg A \vee \neg C) \wedge (\neg D \vee \neg A \vee \neg C) && \text{(distribute } \vee \text{)} \end{aligned}$$

# Clausal Forms

= each wff is translated into an equivalent set of wffs having a specific structure

- **Clausal Form (CF)**

Starting from a wff in CNF

$$\alpha_1 \wedge \alpha_2 \wedge \dots \wedge \alpha_n$$

the clausal form is simply the set of all *clauses*

$$\{\alpha_1, \alpha_2, \dots, \alpha_n\}$$

Examples:

$$(B \vee D) \wedge (A \vee \neg C) \wedge C$$
$$\{(B \vee D), (A \vee \neg C), C\}$$

- **Special notation**

Each clause is usually written as a *set*

$$\beta_1 \vee \beta_2 \vee \dots \vee \beta_n$$
$$\{\beta_1, \beta_2, \dots, \beta_n\}$$

Example:

$$\{\{B, D\}, \{A, \neg C\}, \{C\}\}$$

**A set of *literals*:**  
ordering is irrelevant  
no multiple copies

# Resolution by refutation

- The same example as before

$$B \vee D \vee \neg A \vee \neg C, B \vee C, A \vee D, \neg B \vdash D$$

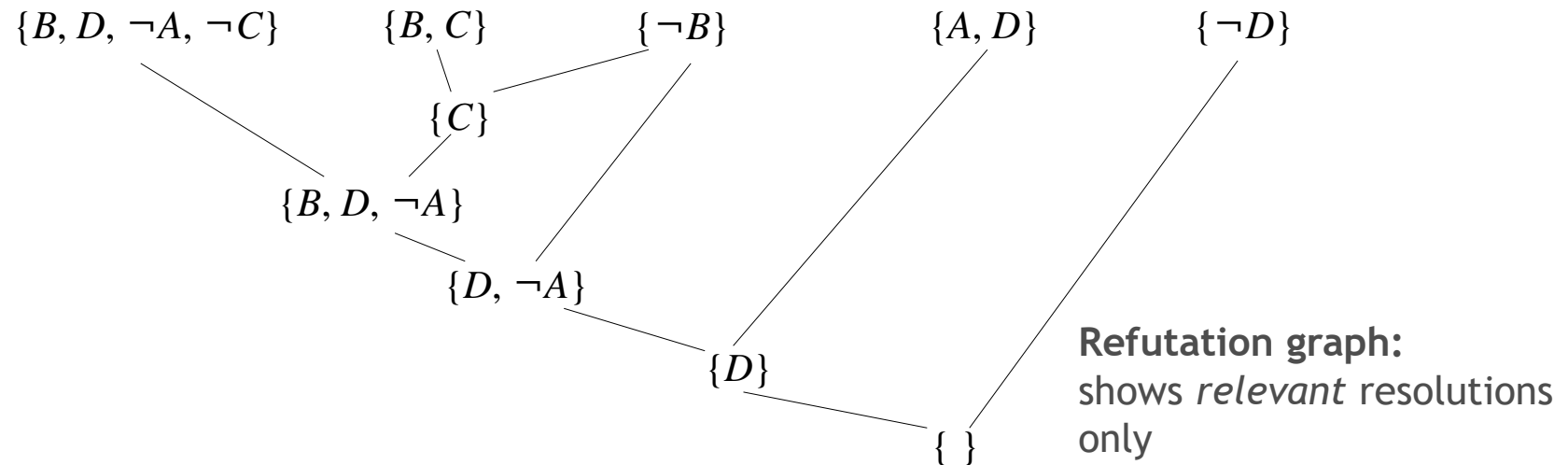
Refutation + rewrite in CNF:

$$B \vee D \vee \neg A \vee \neg C, B \vee C, A \vee D, \neg B, \neg D$$

Rewrite in CF:

$$\{B, D, \neg A, \neg C\}, \{B, C\}, \{A, D\}, \{\neg B\}, \{\neg D\}$$

Applying the resolution rule, one pair of literals at time:





# Resolution by refutation

- The same example as before

$$B \vee D \vee \neg A \vee \neg C, B \vee C, A \vee D, \neg B \vdash D$$

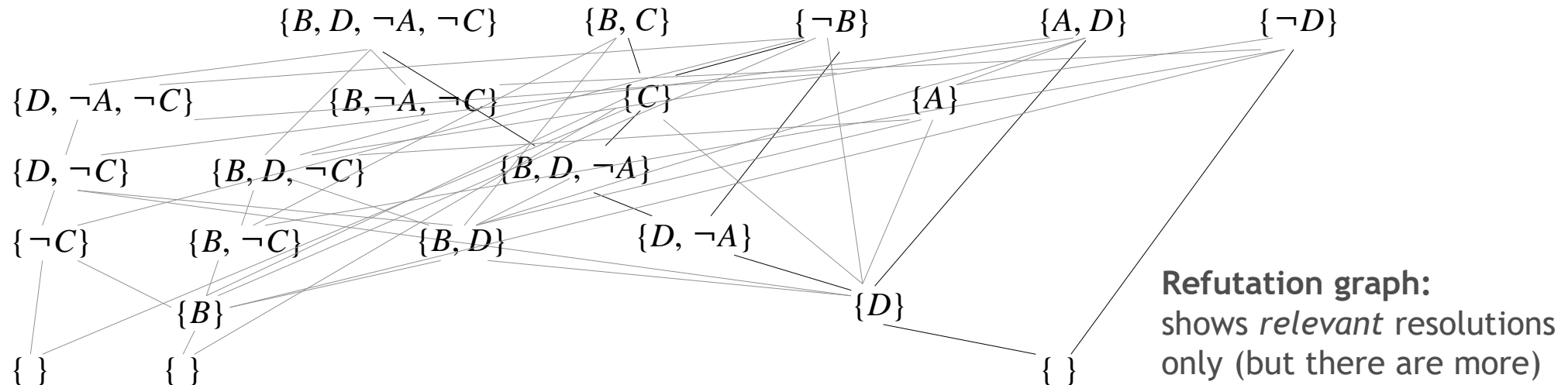
Refutation + rewrite in CNF:

$$B \vee D \vee \neg A \vee \neg C, B \vee C, A \vee D, \neg B, \neg D$$

Rewrite in CF:

$$\{B, D, \neg A, \neg C\}, \{B, C\}, \{A, D\}, \{\neg B\}, \{\neg D\}$$

Applying the resolution rule:



# Resolution by refutation

- Algorithm

Problem: “ $\Gamma \vdash \varphi$ ” ?

The problem is transformed into: is “ $\Gamma \cup \{\neg\varphi\}$ ” *coherent*?

If  $\Gamma \vdash \varphi$  then  $\Gamma \cup \{\neg\varphi\}$  is incoherent and therefore a contradiction can be derived

$\Gamma \cup \{\neg\varphi\}$  is translated into CNF hence in CF

The resolution algorithm is applied to the set of *clauses*  $\Gamma \cup \{\neg\varphi\}$

At each step:

- Select a pair of clauses  $\{C_1, C_2\}$  containing a pair of *complementary literals* making sure that such combination has never been selected before
- Compute  $C_r$  as the *resolvent* of  $\{C_1, C_2\}$  according to the resolution rule.
- Add  $C_r$  to the set of clauses

Termination:

When  $C_r$  is the empty clause  $\{ \}$  (*success*)

or there are no more combinations to be selected in step a) (*failure*)

# Resolution by refutation

- Resolution by refutation for propositional logic

Is correct:  $\Gamma \vdash_{RES} \varphi \Rightarrow \Gamma \models \varphi$

Is complete:  $\Gamma \models \varphi \Rightarrow \Gamma \vdash_{RES} \varphi$

In this sense: iff  $\Gamma \models \varphi$  then there exists a refutation graph

- Algorithm

It is a decision procedure for the problem  $\Gamma \models \varphi$

It has time complexity  $O(2^n)$

where  $n$  is the number of propositional symbols in  $\Gamma \cup \{\neg\varphi\}$