Artificial Intelligence

A course about foundations

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 Σ satisfiable? "

  where Σ is a set of wffs in L_P

  Despite 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

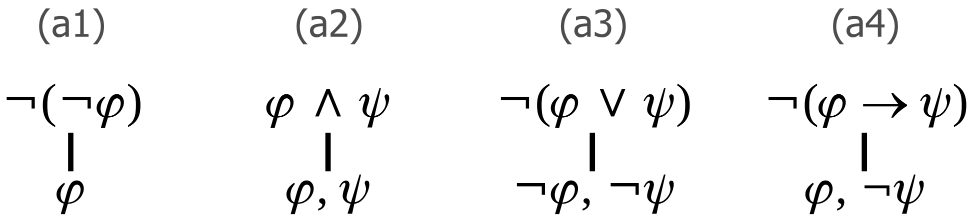
- A tableau is a set of wffs in L_P

  The method starts from an *initial* tableau

  (the set Σ whose satisfiability is to be determined)

  It is based on rules that transform wffs

- Alpha rules (*expansion*)



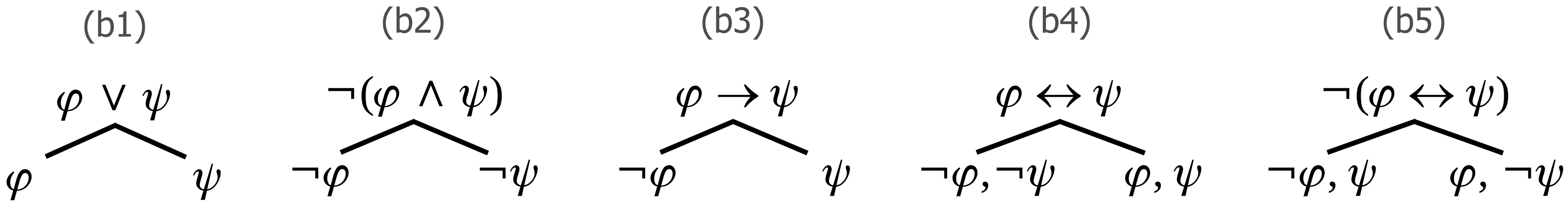
(a1) $\neg(\neg\varphi)$ | φ

(a2) $\varphi \land \psi$ | φ, ψ

(a3) $\neg(\varphi \lor \psi)$ | $\neg\varphi, \neg\psi$

(a4) $\neg(\varphi \to \psi)$ | $\varphi, \neg\psi$

- Beta rules (*bifurcation*)



(b1) $\varphi \lor \psi$ / φ \ ψ

(b2) $\neg(\varphi \land \psi)$ / $\neg\varphi$ \ $\neg\psi$

(b3) $\varphi \to \psi$ / $\neg\varphi$ \ ψ

(b4) $\varphi \leftrightarrow \psi$ / $\neg\varphi, \neg\psi$ \ φ, ψ

(b5) $\neg(\varphi \leftrightarrow \psi)$ / $\neg\varphi, \psi$ \ $\varphi, \neg\psi$

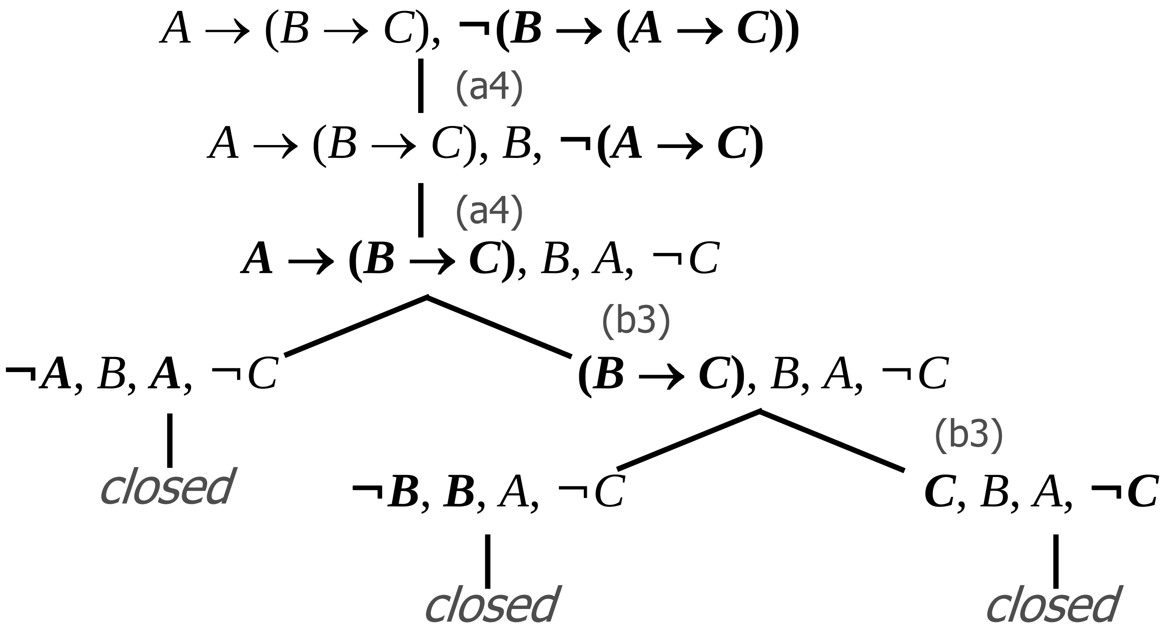
Semantic Tableaux – a working example

- Original problem: “ $\Gamma \models \varphi$? ”

  Example input: $\{ A \to (B \to C) \} \models B \to (A \to C)$?

- Transformed problem: “ is $\Gamma \cup \{\neg\varphi\}$ satisfiable? ”

  Hence, the initial tableau is $\Gamma \cup \{\neg\varphi\}$



$A \to (B \to C), \boldsymbol{\neg(B \to (A \to C))}$

(a4)

$A \to (B \to C), B, \boldsymbol{\neg(A \to C)}$

(a4)

$\boldsymbol{A \to (B \to C)}, B, A, \neg C$

(b3)

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

closed

$\boldsymbol{(B \to C)}, B, A, \neg C$

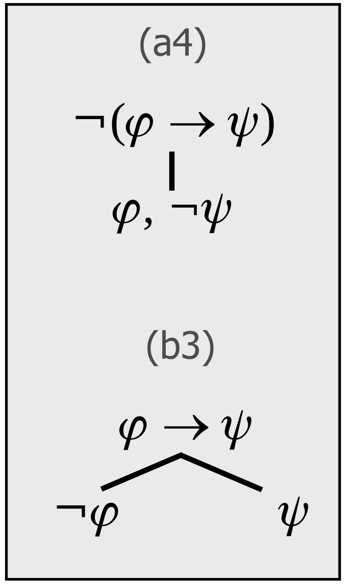
(b3)

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

closed

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

closed



(a4)

$\neg(\varphi \to \psi)$

$\varphi, \neg\psi$

(b3)

$\varphi \to \psi$

$\neg\varphi$ ψ

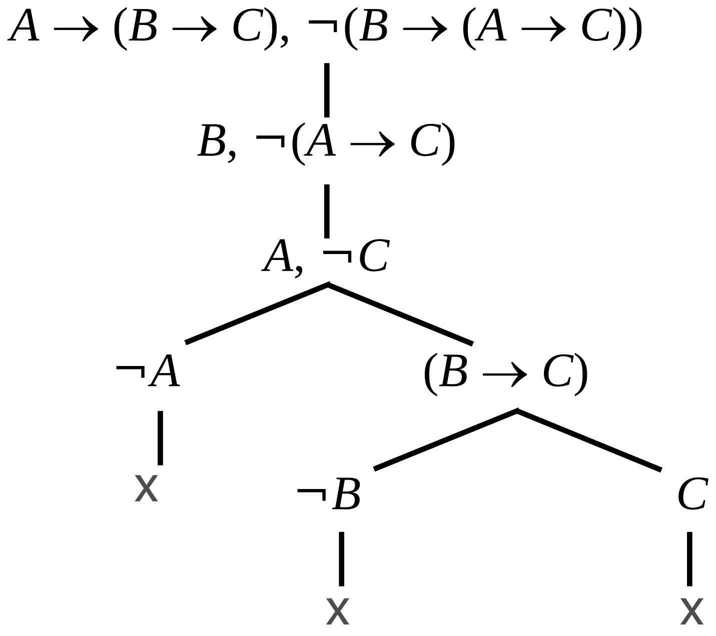
Semantic Tableaux – a working example

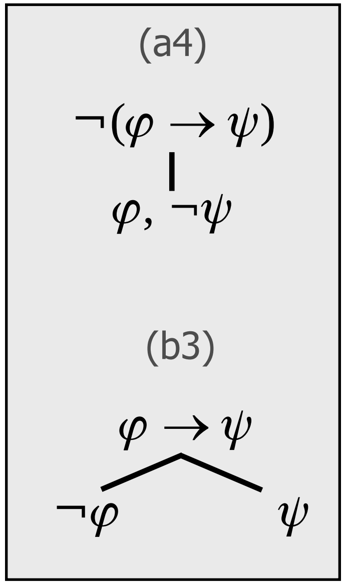
- Original problem: “ $\Gamma \models \varphi$? ”

  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 $\Gamma \cup \{\neg\varphi\}$





NOTE: the usual notation in textbooks is even more concise:
only those wffs that are <u>added</u> 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? "

  Any methods of this type are deemed '*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 <u>all</u> the leaves in the tree are closed *(success)*
  or <u>any</u> of them are open *(failure)*

Semantic Tableaux - (required) algorithm properties

- **Termination**

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

  Each application of either alpha or beta rule *simplifies* a wff (it makes it *less* composite): therefore, the process of applying rules cannot go on forever

- ***Symbolic derivation***

  As already stated, despite its name, this is a *symbolic* method

  We write

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

  iff the *Semantic Tableau* method is successful (= 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 (Ben-Ari's)

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*)

  Both `brute-force search’ and *Semantic Tableaux* methods 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 just 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

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

$\varphi \lor \psi$ is also called the *resolvent* of $\varphi \lor \chi$ and $\neg\chi \lor \psi$

The resolution rule is *correct*

That is, $\varphi \lor \chi, \neg\chi \lor \psi \vdash \varphi \lor \psi \quad \Rightarrow \quad \varphi \lor \chi, \neg\chi \lor \psi \models \varphi \lor \psi$

Proof:

φ	ψ	χ	$\varphi \lor \chi$	$\neg\chi \lor \psi$	$\varphi \lor \psi$
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

  $\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n$

  where each α_i has a structure

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

  where each β_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

  $\beta_1 \vee \beta_2 \vee \ldots \vee \beta_n$

  where each β_i has a structure

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

  where each α_j is a *literal*

Conjunctive Normal Form

- Translation into CNF (it can be automated)

Exhaustive application of the following rules:

1) Rewrite → and ↔ 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:

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

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

$B \vee D \vee \neg A \vee \neg C$ (De Morgan)

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

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

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

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

Clausal Forms

= each wff is translated into an equivalent <u>set</u> of wffs having a specific structure

- **Clausal Form** (CF)

  Starting from a wff in CNF

  $\alpha_1 \land \alpha_2 \land \ldots \land \alpha_n$

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

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

  Examples:

  $(B \lor D) \land (A \lor \neg C) \land C$

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

- **Special notation**

  Each clause is usually written as a *set*

  $\beta_1 \lor \beta_2 \lor \ldots \lor \beta_n$

  $\{\beta_1, \beta_2, \ldots, \beta_n\}$

  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

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

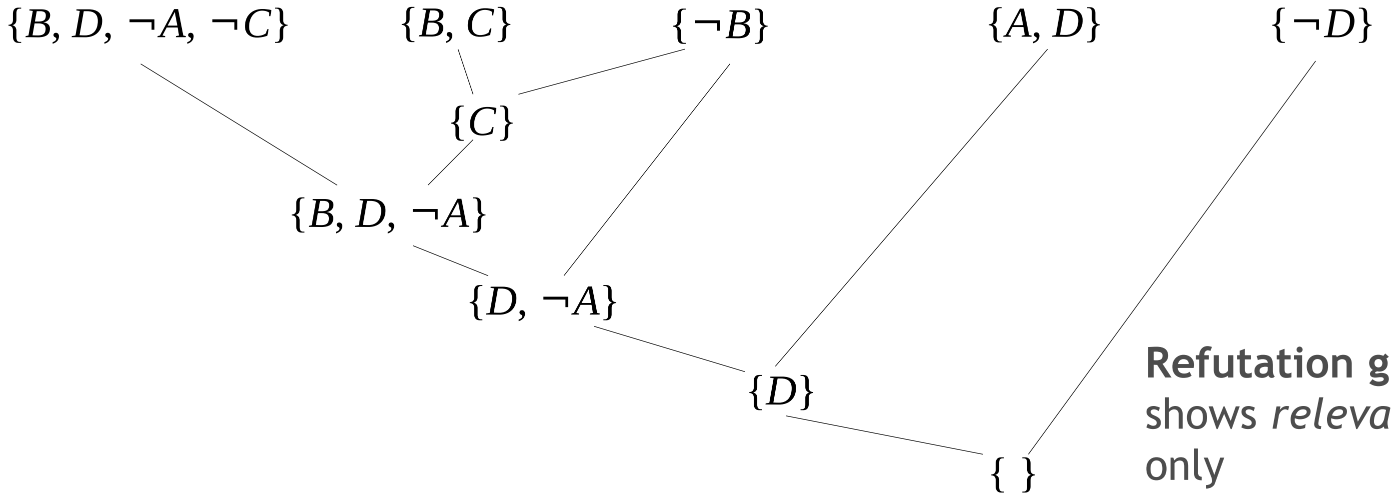
  Refutation + rewrite in CNF:

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

  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*:



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

$\{C\}$

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

$\{D, \neg A\}$

$\{D\}$

$\{\ \}$

Refutation graph:
shows *relevant* resolutions only

Resolution by refutation

- The same example as before

  $B \lor D \lor \neg A \lor \neg C, B \lor C, \; A \lor D, \; \neg B \vdash D$

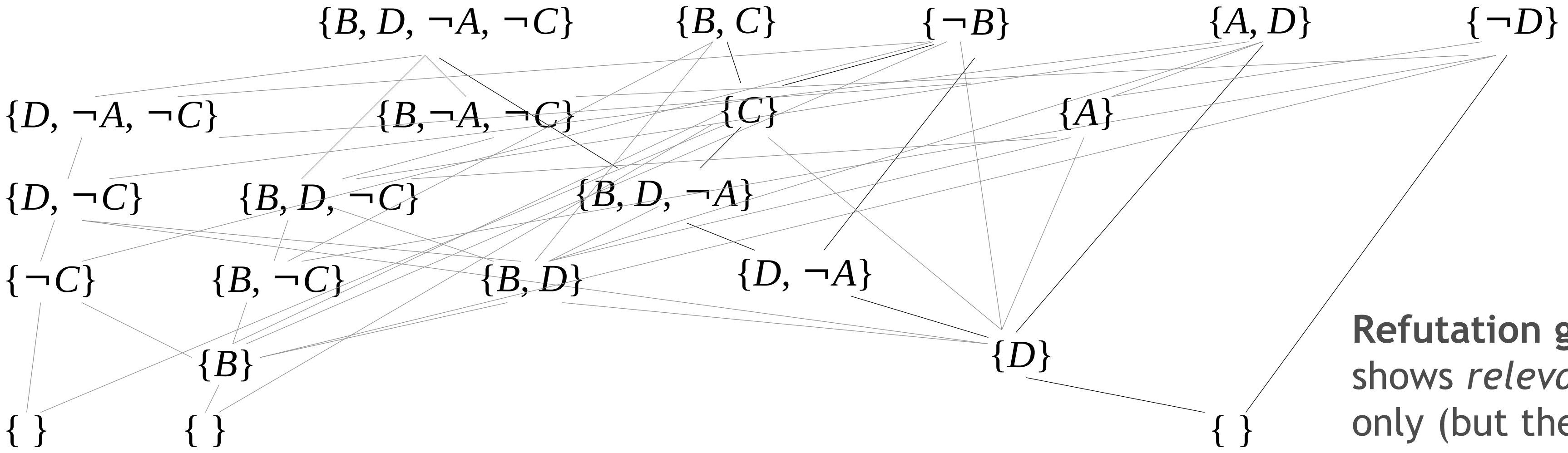
  Refutation + rewrite in CNF:

  $B \lor D \lor \neg A \lor \neg C, B \lor C, \; A \lor D, \; \neg B, \neg D$

  Rewrite in CF:

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

  Applying the resolution rule:



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

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

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

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

$\{B\}$ $\{D\}$

$\{\,\}$ $\{\,\}$ $\{\,\}$

Refutation graph:
shows *relevant* resolutions only (but there are more)

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:

  a) 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

  b) Compute C_r as the *resolvent* of $\{C_1,C_2\}$ according to the resolution rule.

  c) 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 $\Gamma \cup \{\neg\varphi\}$