Artificial Intelligence

An Advanced Course About Foundations



Automated Symbolic Calculus

Marco Piastra

Resolution by Refutation

One inference rule for everything: Resolution

- **Resolution Rule of Inference**

means 'symbolic derivability'

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

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

The resolution rule is *correct*

means 'entailment'

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

Proof:

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

Correctness can be checked via truth table

Normal Forms

= *translation of each* wff *into an equivalent* wff *having a specific* <u>*structure*</u>

- **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 $\rightarrow$ 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:

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

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

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

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

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

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

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

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

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

  Examples:

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

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

- **Set Notation**

  Each clause is transformed into a *set of literals*

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

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

  Example:

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

***Literals*:**
each β_i is either an *atom* or the *negation* of an atom

A <u>set</u> of literals:
ordering is irrelevant, no multiple copies

Resolution by Refutation

- An example:

  $\neg(\neg(\neg B \rightarrow D) \wedge \neg(A \rightarrow \neg C)), \neg B \rightarrow C, A \vee D, \ \neg B \vdash D$

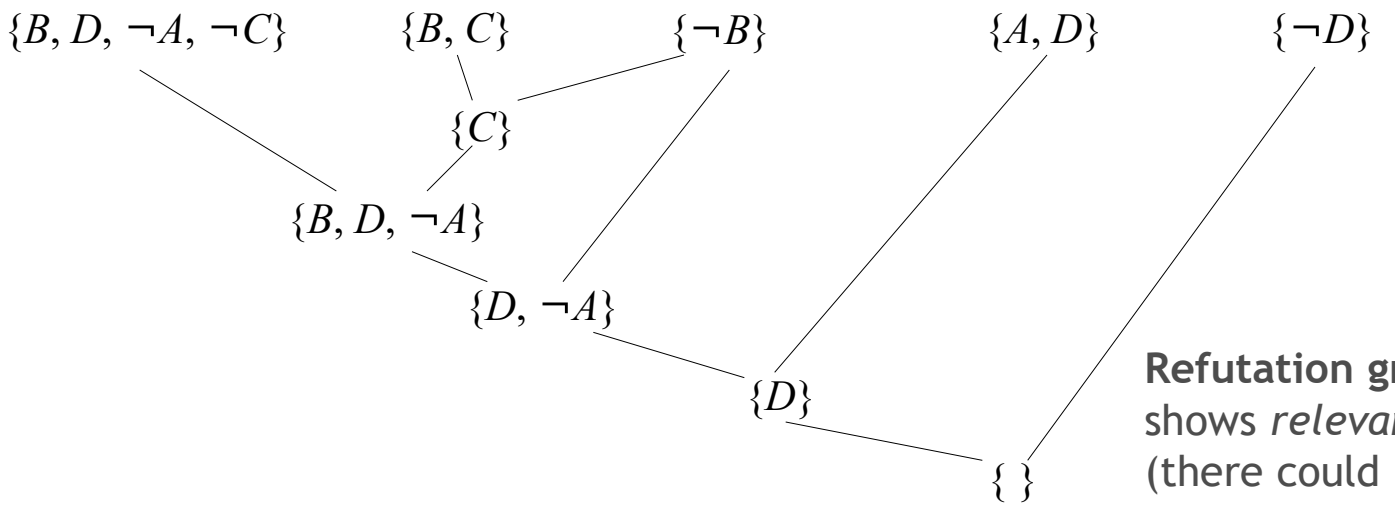
  Refutation + rewrite in CNF:

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

  Rewrite in CF:

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

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



Refutation graph:
shows *relevant* resolutions only
(there could be more)

Resolution by Refutation

- An example:

$\neg(\neg(\neg B \rightarrow D) \wedge \neg(A \rightarrow \neg C)), \neg B \rightarrow C, A \vee D, \neg B \vdash D$

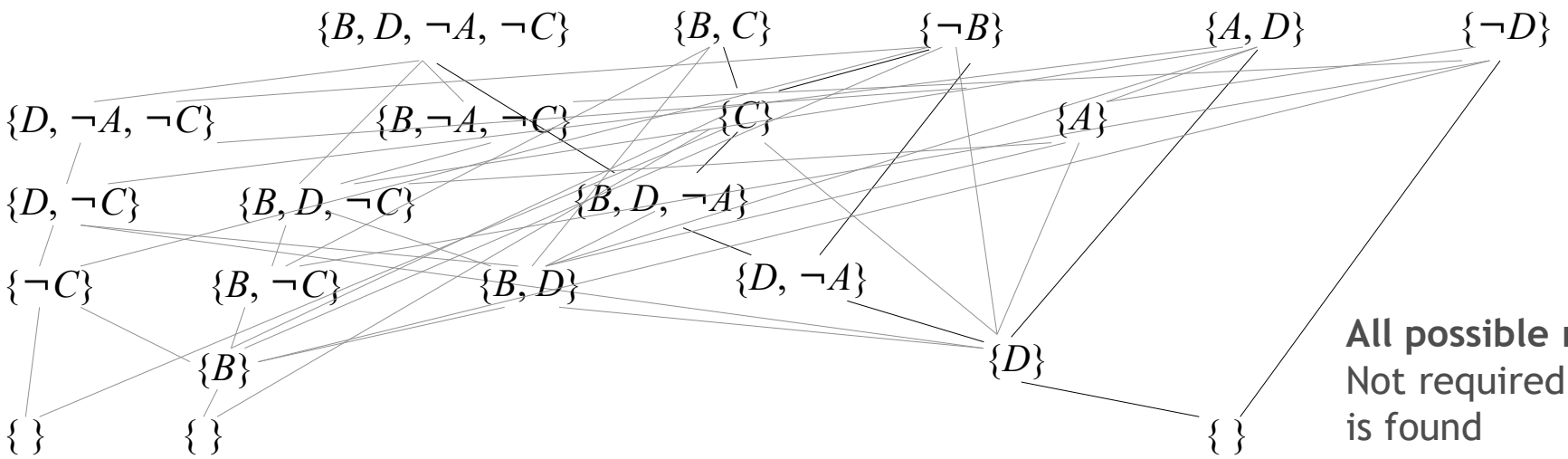
Refutation + rewrite in CNF:

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

Rewrite in CF:

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

Applying the resolution rule:



All possible resolutions
Not required, once an *empty clause* is found

Resolution by Refutation

- Algorithm

  Problem: "$\Gamma \vdash \varphi$" ?

  - The problem is transformed into: is "$\Gamma \cup \{\neg\varphi\}$" *coherent (=satisfiable)*?

    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 into CF, in set notation

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 clauses 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

d) Go back to step a)

Termination:

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

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

Resolution by Refutation – Algorithm properties

- **Termination**

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

  Each application of the resolution rule 'uses' a pair of clauses in a finite set, each resolvent is smaller than the union of the resolved clauses. Therefore, it cannot go on forever

- ***Symbolic derivation***

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

  We write

  $$\Gamma \vdash \varphi$$

  iff the resolution method is successful (= the empty clause is derived) for $\Gamma \cup \{\neg\varphi\}$

How do we know that $\Gamma \vdash \varphi \Rightarrow \Gamma \models \varphi$?

(*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 the *inference rule*)

How do we know that $\Gamma \models \varphi \Rightarrow \Gamma \vdash \varphi$?

(*Completeness* of the method)

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

Resolution by Refutation - Algorithm properties

- **Termination**

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

  Each application of the resolution rule 'uses' a pair of clauses in a finite set, each resolvent is smaller than the union of the resolved clauses. Therefore, it cannot go on 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)

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

Can the Resolution Algorithm be made *faster*?

Horn Clauses (in L_P)

- Definition

  A ***Horn Clause*** is a wff in CF
  that contains at most <u>one</u> literal in positive form

- Three types of *Horn Clauses*:

  Rule: two or more literals, one positive

  Examples: $\{B, \neg D, \neg A, \neg C\}$, $\{A, \neg B\}$ (equivalent to: $(D \wedge A \wedge C) \rightarrow B,\ B \rightarrow A$)

  Facts: just one positive literal

  Examples: $\{B\}$, $\{A\}$

  Goal: one or more literals, all negative

  Examples: $\{\neg B\}$, $\{\neg A, \neg B\}$

  More terminology:

  Rules and facts are also called *definite clauses*

  Goals are allo called *negative clauses*

Lost in Translation…

Many wffs can be translated into Horn clauses:

$(A \wedge B) \rightarrow C$
$\neg(A \wedge B) \vee C$ (rewriting $\rightarrow$)
$\neg A \vee \neg B \vee C$ (De Morgan - CF – it is a rule)

$A \rightarrow (B \wedge C)$
$\neg A \vee (B \wedge C)$ (rewriting $\rightarrow$)
$(\neg A \vee B) \wedge (\neg A \vee C)$ (distributing $\vee$)
$(\neg A \vee B), (\neg A \vee C)$ (CF – two rules)

$(A \vee B) \rightarrow C$
$\neg(A \vee B) \vee C$ (rewriting $\rightarrow$)
$(\neg A \wedge \neg B) \vee C$ (De Morgan)
$(\neg A \vee C) \wedge (\neg B \vee C)$ (distributing $\vee$)
$(\neg A \vee C), (\neg B \vee C)$ (CF – two rules)

But not all of them:

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

$A \rightarrow (B \vee C)$
$\neg A \vee B \vee C$ (rewriting $\rightarrow$)

SLD Resolution

Linear resolution with Selection function for Definite clauses

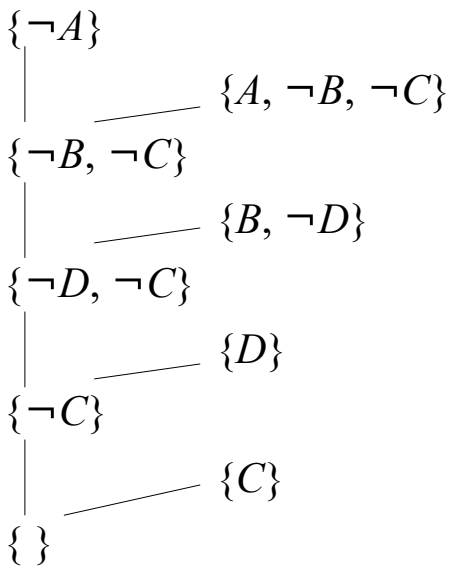
- Algorithm
  - Input *(Horn clauses)*
    - A set of *rules* and *facts* (KB) plus one *goal*
  - Resolution Loop
    - Select a *literal* from the current *goal* clause *(with a deterministic strategy)*
    - Find a clause in the KB *(with a deterministic strategy)* that contains the *complementary literal*
    - Apply *binary resolution* to derive a *resolvent*
    - Replace the *selected literal* in the goal clause with the *resolvent*
  - Termination
    - If the empty clause is derived → success
    - If all resolution paths fail → failure

Example:

Selection function: ***leftmost subgoal first***

Definite clauses: $\{C\}$, $\{D\}$, $\{B, \neg D\}$, $\{A, \neg B, \neg C\}$

Goal: $\{\neg A\}$



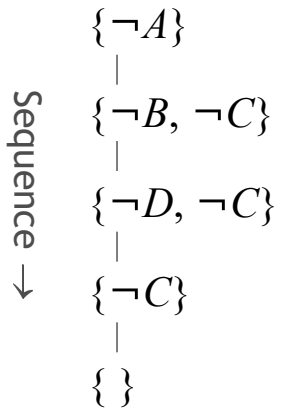
SLD Trees: *backtracking*

SLD derivations

Example: $\{C\}$, $\{D\}$, $\{B, \neg D\}$, $\{A, \neg B, \neg C\}$ goal $\{\neg A\}$

In this example each subgoal can be resolved in one mode only

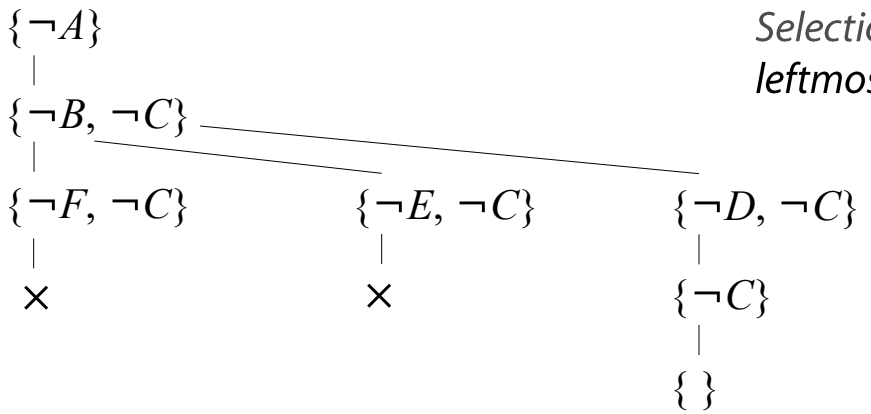
This is not true in general



- SLD trees (= trace of all SLD derivations from a goal)

  Example: $\{C\}$, $\{D\}$, $\{B, \neg F\}$, $\{B, \neg E\}$, $\{B, \neg D\}$, $\{A, \neg B, \neg C\}$ goal $\{\neg A\}$

  A few new rules have been added: there are now different possibilities



Selection function:
leftmost subgoal first

Each branch correspond to a possible resolution for a *subgoal*

SLD Resolution

Linear resolution with Selection function for Definite clauses

- Algorithm

  Input (Horn clauses)

  A set of rules and facts (KB) plus one goal

  Resolution Loop

  Select a *literal* from the current *goal* clause
  (with a deterministic strategy)

  Find a clause in the KB *(with a deterministic strategy)*
  that contains the *complementary literal*

  Apply *binary resolution* to derive a *resolvent*

  Replace the *selected literal* in the goal clause with the *resolvent*

  Backtracking

  If no clause can resolve the selected literal

  - Retract the last resolution step
  - Select an alternative clause from KB
  - Repeat until a literal is resolved or fail if no alternatives found

  Termination

  If the empty clause is derived → success

  If all resolution paths fail → failure

SLD Resolution

A resolution algorithm <u>for Horn clauses</u> in L_P

It always terminates
It is *correct*: $\Gamma \vdash \varphi \Rightarrow \Gamma \models \varphi$
It is *complete*: $\Gamma \models \varphi \Rightarrow \Gamma \vdash \varphi$

- ***Computationally efficient***

  It has polynomial time complexity (w.r.t the # of propositional symbols occurring in Γ and φ)

- ***Reduced scope***

  Not all entailment problems $\Gamma \models \varphi$ can be reduced to Horn Clauses

  In this sense, SLD Resolution is a tradeoff