Propositional Resolution

Marco Piastra

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*

In fact $\varphi \vee \chi, \neg\chi \vee \psi \vdash \varphi \vee \psi \quad \Rightarrow \quad \varphi \vee \chi, \neg\chi \vee \psi \models \varphi \vee \psi$

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

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 $\rightarrow$ 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 set 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:

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

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

- **Special notation**

  Each clause is usually written as a *set*

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

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

  Example:

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

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

Resolution by refutation

- 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 this combination has never been selected before

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

  c) Add C to the set of clauses

  Termination:

  When C is the empty clause $\{\ \}$

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

Resolution by refutation

- The same example as before

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

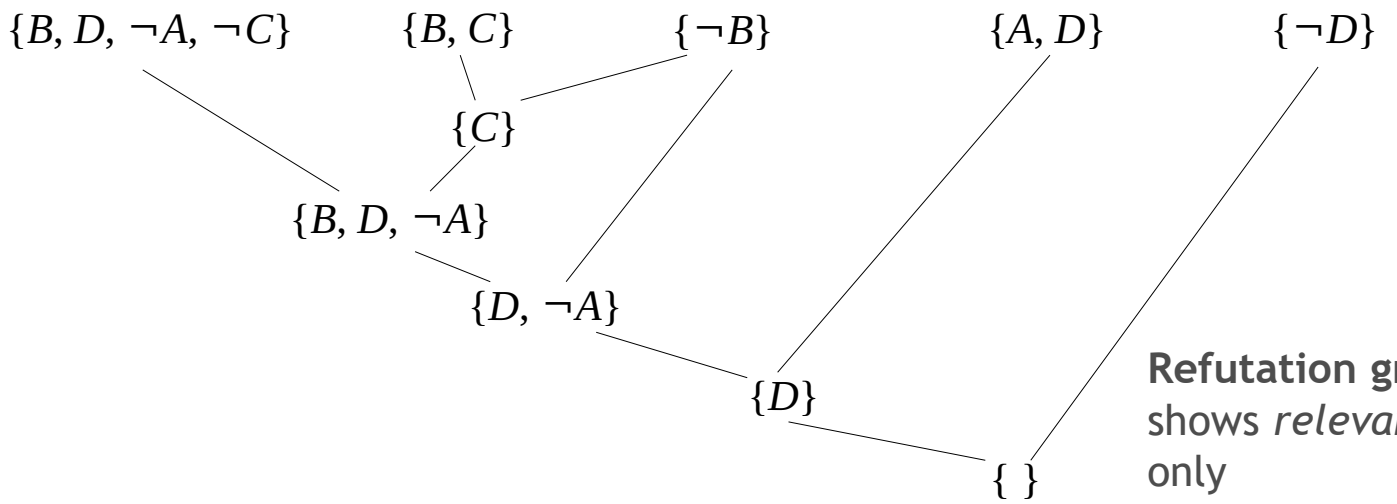
  Refutation + rewrite in CNF:

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

  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*:



Refutation graph:
shows *relevant* resolutions only

Resolution by refutation

- The same example as before

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

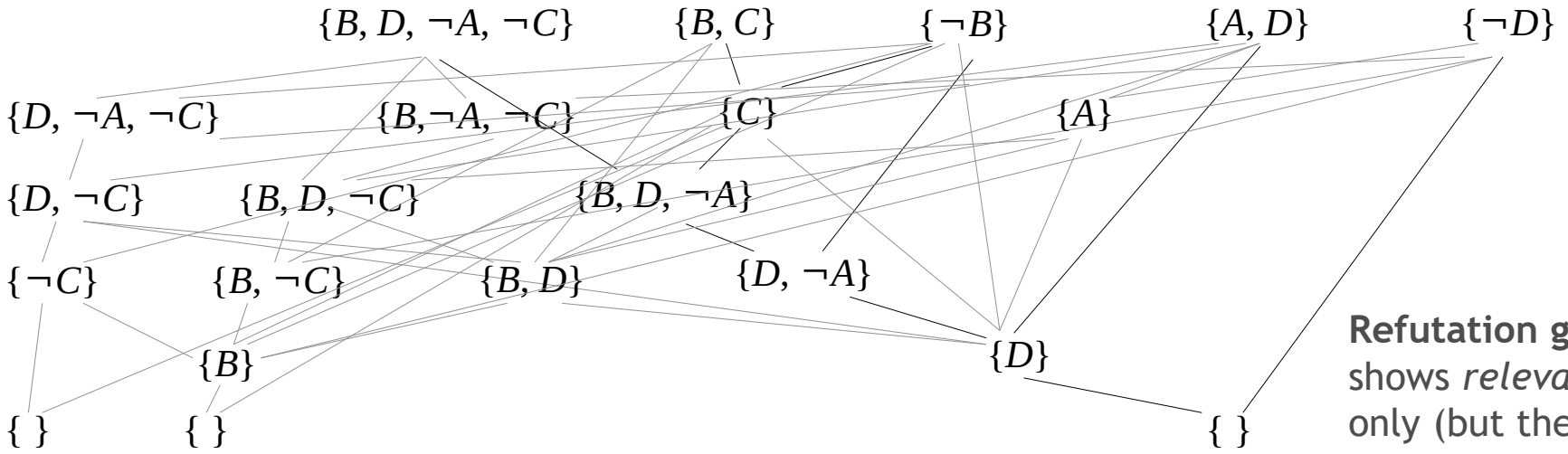
  Refutation + rewrite in CNF:

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

  Rewrite in CF:

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

  Applying the resolution rule:



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

Resolution by refutation

- Resolution by refutation for propositional logic

  Is correct: $\Gamma \vdash \varphi \Rightarrow \Gamma \models \varphi$

  Is complete: $\Gamma \models \varphi \Rightarrow \Gamma \vdash \varphi$

  In this sense: if $\Gamma \models \varphi$ 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 $\Gamma \cup \{\neg\varphi\}$

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

  Starts from a set of *definite clauses* (also the *program*) + a *goal*

  1) At each step, the *selection function* identifies a *literal* in the *goal* (i.e. *subgoal*)
  2) All *definite clause* applicable to the *subgoal* is selected
  3) The resolution rule is applied generating the *resolvent*

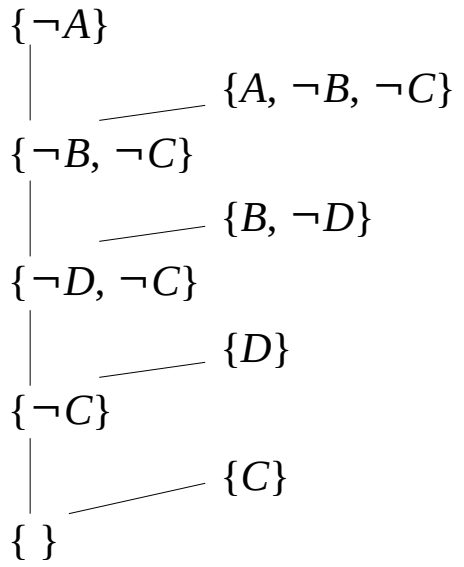
  Termination: either the empty clause { } is obtained or step 2) fails.

Example:

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

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

Goal: $\{\neg A\}$



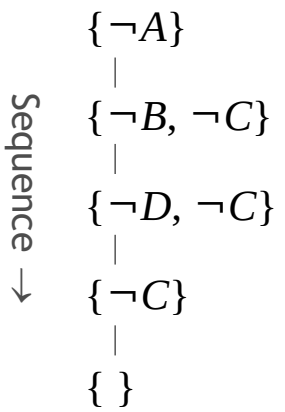
SLD trees

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

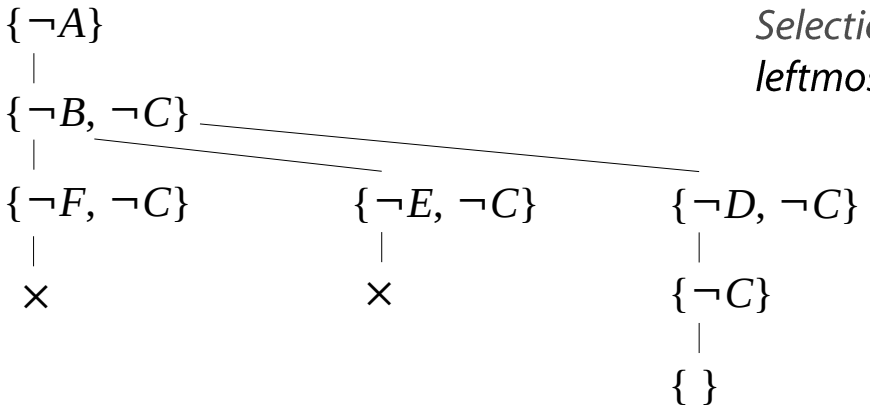
This is not true in general



- 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

- A resolution method <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)