First-Order Resolution

Marco Piastra

Propositional Resolution

A decision method for $\Gamma \models \varphi$

a) Refutation $\Gamma \cup \{\neg\varphi\}$ and translation into *conjuctive normal form* (CNF)

$\beta_1 \wedge \beta_2 \wedge \ldots \wedge \beta_n$ where each β_i is a disjuction of literals (i.e. A or $\neg A$)

b) Translation of $\Gamma \cup \{\neg\varphi\}$ in *clausal form* (CF)

$\{\beta_1, \beta_2, \ldots, \beta_n\}$ where each β_i is a *clause* (i.e. a set of literals, representing a disjunction)

c) Exhaustive application of the resolution rule

1) Selection of two clauses $\{\beta_1, \beta_2, \ldots, \beta_n, \alpha\}, \{\neg\alpha, \gamma_1, \gamma_2, \ldots, \gamma_m\}$
2) Generation of the *resolvent*
   $\{\beta_1, \beta_2, \ldots, \beta_n, \alpha\}, \{\neg\alpha, \gamma_1, \gamma_2, \ldots, \gamma_m\} \vdash \{\beta_1, \beta_2, \ldots, \beta_n, \gamma_1, \gamma_2, \ldots, \gamma_m\}$

Termination conditions:

1) The empty clause has been derived (*success*)
2) No further resolutions are possible – *fixed point* (*failure*)

Clausal Form in L_{PO}

a) Refutation: $\Gamma \cup \{\neg\varphi\}$

b) Translation into PNF and *skolemization* $sko(\Gamma \cup \{\neg\varphi\})$:

All wff are now in the form:

$\forall x_1 \forall x_2 \ldots \forall x_n \psi$ (the *matrix* ψ does not contain quantifiers)

Given that all wffs are universal sentences, the universal quantifiers can just be omitted

c) Removal of all universal quantifiers in $sko(\Gamma \cup \{\neg\varphi\})$:

At this point, all wffs in $sko(\Gamma \cup \{\neg\varphi\})$ contain only *atoms* (possibly with *variables*), connectives and parenthesis

Example:

1: $\forall x\, (P(x) \to (\exists y\, Q(x,y) \wedge R(y)))$

2: $\forall x\, \exists y\, (P(x) \to (Q(x,y) \wedge R(y)))$ (PNF)

3: $\forall x\, (P(x) \to (Q(x, k(x)) \wedge R(k(x))))$ (Skolemization, with a new function $k/1$)

4: $P(x) \to (Q(x, k(x)) \wedge R(k(x)))$ (removal of universal quantifiers)

Just atoms, connectives and parentheses…

Clausal Form in L_{PO}

a) Refutation: $\Gamma \cup \{\neg\varphi\}$

b) Translation into PNF and *skolemization* $sko(\Gamma \cup \{\neg\varphi\})$:

All wff are now in the form:

$\forall x_1 \forall x_2 \ldots \forall x_n \psi$ (the *matrix* ψ does not contain quantifiers)

Given that all wffs are universal sentences, the universal quantifiers can just be omitted

c) Removal of all universal quantifiers in $sko(\Gamma \cup \{\neg\varphi\})$:

The *clausal form* can be obtained by just treating atoms as propositions and applying the rules seen in the propositional case

Example:

4: $P(x) \rightarrow (Q(x, k(x)) \wedge R(k(x)))$ (from before)

5: $\neg P(x) \vee (Q(x, k(x)) \wedge R(k(x)))$ (removing $\rightarrow$)

6: $(\neg P(x) \vee Q(x, k(x))) \wedge (\neg P(x) \vee R(k(x)))$ (CNF, by distributing $\vee$)

7: $\{\neg P(x), Q(x, k(x))\}, \ \{\neg P(x), R(k(x))\}$ (*Clausal Form*)

Unificare necesse est, for resolution

- Problem: $\Gamma \models \varphi$?

  $\Gamma \equiv \{\forall x\ (Philosopher(x) \rightarrow Uman(x)), \forall x\ (Uman(x) \rightarrow Mortal(x)), Philosopher(socrates)\}$

  $\varphi \equiv Mortal(socrates)$

  Refutation, translation, clausal form:

  1: $\{\forall x\ (Philosopher(x) \rightarrow Uman(x)), \forall x\ (Uman(x) \rightarrow Mortal(x)),$
  $Philosopher(socrates), \neg Mortal(socrates)\}$

  ($\Gamma \cup \{\neg\varphi\}$ is already in PNF, no skolemization is needed)

  2: $\{\{Uman(x), \neg Philosopher(x)\}, \{Mortal(x), \neg Uman(x)\}, \{Philosopher(socrates)\},$
  $\{\neg Mortal(socrates)\}\}$

  (*Clausal Form*)

  Resolution method (first attempt):

  3: $\{Uman(x), \neg Philosopher(x)\}, \{Mortal(x), \neg Uman(x)\}\{\neg Philosopher(x)\ , Mortal(x)\}$

  4: Try resolving: $\{Uman(socrates)\}, \{Mortal(x), \neg Uman(x)\}$

  ???

Unification

Replacing variables with terms may render two atoms *identical*

- **Unifier**

  A substitution of variables with terms $\sigma = [x_1/t_1,\ x_2/t_2\ \ldots\ x_n/t_n]$
  that makes two complementary literals α and $\neg\beta$ *resolvable*

  That is, it makes the two atoms *identical*: $\sigma(\alpha) = \sigma(\beta)$

  - *Recursive* substitutions are not allowed: in x_i/t_i, x_i **cannot** occur in t_i
  - Obviously, a unifier does not necessarily exist:
    for instance $P(g(x, f(a)), a)$ and $\neg P(g(b, f(w)), k(w))$ are not unifiable

- MGU - *most general unifier*

  It is the minimal *unifier* of α and $\neg\beta$

  $$\text{MGU}\ \mu \Leftrightarrow \forall\, \sigma\ \exists\, \sigma' : \sigma = \mu \cdot \sigma'$$

  Any other unifier can be obtained as a composition of μ

  Esiste un algoritmo che trova μ (se la coppia α e $\neg\beta$ è unificabile, ovviamente)

Constructing the MGU

- Martelli and Montanari's algorithm

  Input: $\{s_1 = t_1, s_2 = t_2 \ldots s_n = t_n\}$ (a system of *symbolic* equations)

  Procedure:

  Exhaustive application to the system of symbolic equations
  (each rule *transforms* the original system)

(1)	$f(s_1, ..., s_n) = f(t_1, ..., t_n)$	*replace by the equations* $s_1 = t_1, ..., s_n = t_n$,
(2)	$f(s_1, ..., s_n) = g(t_1, ..., t_m)$ where $f \neq g$	*halt with failure*,
(3)	$x = x$	*delete the equation*,
(4)	$t = x$ where t is not a variable	*replace by the equation* $x = t$,
(5)	$x = t$ where x does not occur in t and x occurs elsewhere	*apply the substitution* $\{x/t\}$ *to all other equations*
(6)	$x = t$ where x occurs in t and x differs from t	*halt with failure.*

Unless an explicit failure occurs (i.e. by rules (2) or (6)), the procedure terminates with success if no further rule is applicable

Constructing the MGU: examples

Example: $\{f(x, a) = f(g(z), y), h(u) = h(d)\}$

$\{x = g(z), y = a, h(u) = h(d)\}$	Rule (1) on $f(x, a) = f(g(z), y)$
$\{x = g(z), y = a, u = d\}$	Rule (1) on $h(u) = h(d)$, MGU

Example: $\{f(x, a) = f(g(z), y), h(x, z) = h(u, d)\}$

$\{x = g(z), y = a, h(x, z) = h(u, d)\}$	Rule (1) on $f(x, a) = f(g(z), y)$
$\{x = g(z), y = a, h(g(z), z) = h(u, d)\}$	Rule (5) on $x = g(z)$
$\{x = g(z), y = a, u = g(z), z = d\}$	Rule (1) on $h(g(z), z) = h(u, d)$
$\{x = g(d), y = a, u = g(d), z = d\}$	Rule (5) on $z = d$, MGU

Example: $\{f(x, a) = f(g(z), y), h(x, z) = h(d, u)\}$

$\{x = g(z), y = a, h(x, z) = h(d, u)\}$	Rule (1) on $f(x, a) = f(g(z), y)$
$\{x = g(z), y = a, h(g(z), z) = h(d, u)\}$	Rule (5) on $x = g(z)$
$\{x = g(z), y = a, g(z) = d, z = u\}$	Rule (2) on $g(z) = d$ FAILURE

Resolution with unification for L_{FO}

A correct procedure for $\Gamma \models \varphi$ in L_{FO}

a) Refutation $\Gamma \cup \{\neg\varphi\}$,

b) Prenex normal form and skolemization $sko(\Gamma \cup \{\neg\varphi\})$

c) Translation of $sko(\Gamma \cup \{\neg\varphi\})$ into CNF hence into CF

d) Repeat application of the resolution method:

   1) Selection of two clauses $\{\beta_1, \beta_2, \ldots, \beta_n, \alpha\}, \{\neg\alpha', \gamma_1, \gamma_2, \ldots, \gamma_m\}$
   2) *Standardization* of variables
      (i.e. create new copies of the two clauses having new and unique variables)
   3) Construction of the MGU μ (if it exists) for the two literals α e α'
   4) Application generation of the resolvent with the application of μ
      $\{\beta_1, \beta_2, \ldots, \beta_n, \alpha\}[\mu], \{\neg\alpha', \gamma_1, \gamma_2, \ldots, \gamma_m\}[\mu] \vdash \{\beta_1, \beta_2, \ldots, \beta_n, \gamma_1, \gamma_2, \ldots, \gamma_m\}[\mu]$

e) Until

   1) The empty clause has been derived (*success*)
   2) No further resolutions are possible – *fixed point* (*failure*)

   But the method is not guaranteed to terminate (i.e. it might *diverge*)

The method might diverge…

Problem: $\forall x\,(Q(f(x)) \rightarrow P(x)) \models \exists x\,(P(f(x)) \wedge \neg Q(f(x)))$

Refutation:

$\{\ \forall x\,(Q(f(x)) \rightarrow P(x))\ \} \cup \{\neg \exists x\,(P(f(x)) \wedge \neg Q(f(x)))\ \}$

Prenex normal form:

$\{\ \forall x\,(Q(f(x)) \rightarrow P(x))\ \} \cup \{\ \forall x\ \neg(P(f(x)) \wedge \neg Q(f(x)))\ \}$

(no *skolemizzation* required)

Clausal form:

$\{\ Q(f(x)) \rightarrow P(x)\ \} \cup \{\neg(P(f(x)) \wedge \neg Q(f(x)))\ \}$

$\{\neg Q(f(x)) \vee P(x)\ \} \cup \{\ \neg P(f(x)) \vee Q(f(x))\ \}$

$\{\{\neg Q(f(x)) \vee P(x)\}, \{\neg P(f(x)) \vee Q(f(x))\}\}$

Resolution:

1: $\{\neg Q(f(x_1)), P(x_1)\}, \{\neg P(f(x_2)), Q(f(x_2))\}, [x_1/f(x_2)] \vdash \{\neg Q(f(f(x_2))), Q(f(x_2))\}$

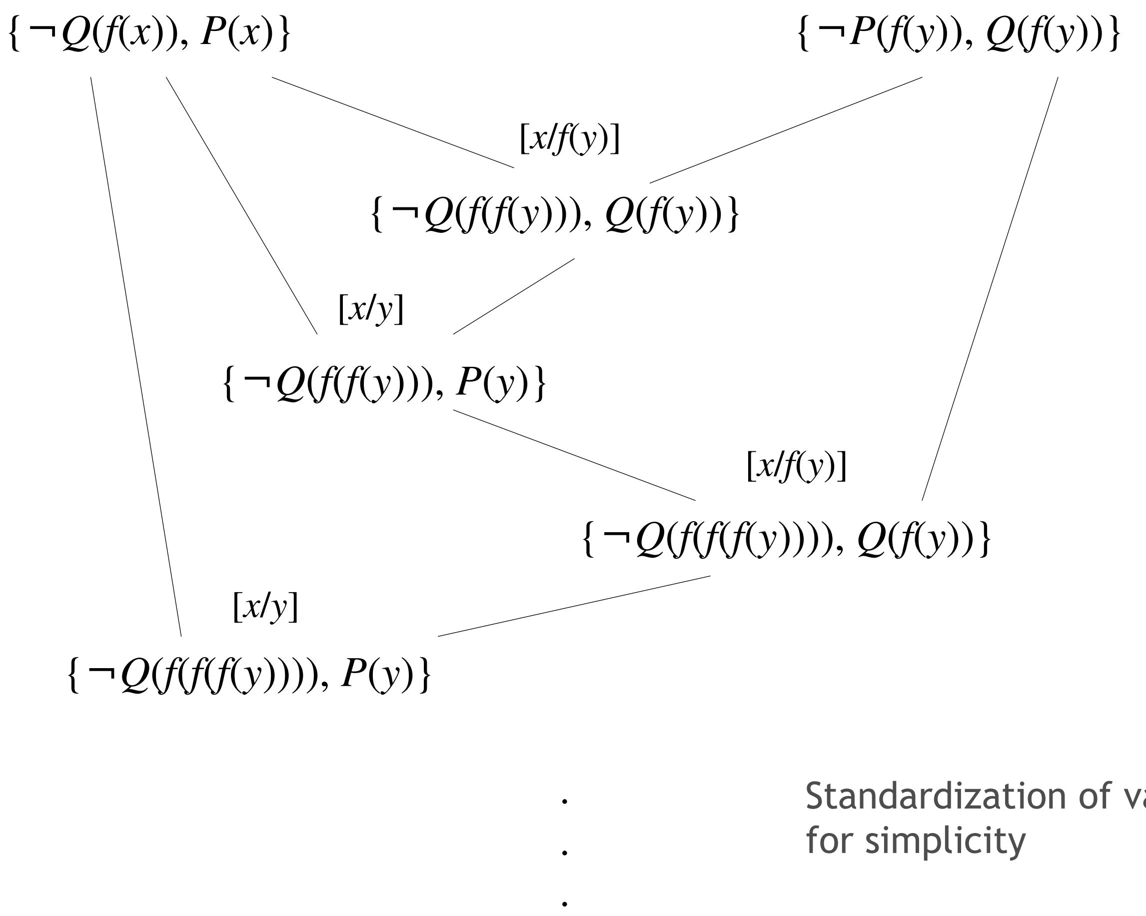
2: $\{\neg Q(f(x_3)), P(x_3)\}, \{\neg Q(f(f(x_4))), Q(f(x_4))\}, [x_3/x_4] \vdash \{\neg Q(f(f(x_4))), P(x_4)\}$

3: $\{\neg Q(f(f(x_5))), P(x_5)\}, \{\neg P(f(x_6)), Q(f(x_6))\}, [x_5/f(x_6)] \vdash \{\neg Q(f(f(f(x_6)))), Q(f(x_6))\}$

4: $\{\neg Q(f(x_7)), P(x_7)\}, , \{\neg Q(f(f(f(x_8)))), Q(f(x_8))\}, [x_7/x_8] \vdash \{\neg Q(f(f(f(x_8)))), P(x_8)\}$

…

The method might diverge…



Standardization of variabiles not shown here, for simplicity

Properties of resolution with unification

- The method is *correct* in L_{FO}

  If the method finds the empty clause for $sko(\Gamma \cup \{\neg\varphi\})$ then $\Gamma \models \varphi$

- Is the method *complete* in L_{FO} ?

  Within the limits of semi-decidability, yes (Robinson, 1963)

  When $\Gamma \models \varphi$, the method will eventually find the empty clause for $sko(\Gamma \cup \{\neg\varphi\})$

  Very often (but not in the worst case) the method is more efficient than the one in the corollary of Herbrand's theorem

  The advantage is due to *lifting*
  (the method can resolve also non-ground clauses)

  When $\Gamma \not\models \varphi$, the method might diverge

  In pratice however (see Prolog) the method might diverge even when $\Gamma \models \varphi$

  Critical element:

  - Selecting the clauses and literals to be resolved

Esempio: il mondo delle liste

- Liste di oggetti $[a, b, c, \ldots]$

  $cons(s, x)$
  funzione, associa ad un oggetto (es. a) ed una lista (es. $[b, c]$)
  la lista ottenuta inserendo l'oggetto all'inizio (es. $[a, b, c]$)_

  $Append(x,y,z)$
  predicato, associa alle liste x e y la concatenazione z

  nil
  costante, indica la lista vuota.

  Notazione abbreviata (Prolog):
  $[] \Leftrightarrow nil$
  $[a] \Leftrightarrow cons(a,nil)$
  $[a,b] \Leftrightarrow cons(a,cons(b,nil))$
  $[a|[b,c]] \Leftrightarrow cons(a,[b,c])$

Assiomi (**AL**)

$\forall x\ Append(nil,x,x)$
$\forall x \forall y \forall z\ (Append(x,y,z) \rightarrow \forall s\ Append([s,x],y,[s,z]))$

Esempi (conseguenze logiche)

$\mathbf{AL} + \exists z\ Append([a],[b,c],z)$	$\models Append([a],[b,c],[a,b,c])$	$= [z/[a,b,c]]$
$\mathbf{AL} + \exists x \exists y\ Append(x,y,[a,b])$	$\models Append([a],[b],[a,b])$	$= [x/[a], x/[b]]$
	$\models Append(nil,[a,b],[a,b])$	$= [x/nil, y/[a,b]]$
	$\models Append([a,b],nil,[a,b])$	$= [x/[a,b], y/nil]$

Esempio: il mondo delle liste

Problema: $\forall x\ Append(nil, x, x) \models \exists y\ \forall x\ Append(nil, cons(y, x), cons(a, x))$

1: $\forall x\ Append(nil, x, x),\ \neg\exists y\ \forall z\ Append(nil, cons(y, z), cons(a, z))$

(refutazione e *ridenominazione* delle variabile x)

2: $\forall x\ Append(nil, x, x),\ \forall y\ \exists z\ \neg Append(nil, cons(y, z), cons(a, z))$ (forma normale prenessa)

3: $\{Append(nil, x, x)\},\ \{\neg Append(nil, cons(y, k(y)), cons(a, k(y)))\}$

($k/1$ funzione di Skolem, forma a clasuole)

(N.B. il Prolog non fa la *skolemizzazione*: deve farla il programmatore)

La coppia di **letterali**

$Append(nil, x, x),\ \neg Append(nil, cons(y, k(y)), cons(a, k(y))))$

… è compatibile (stesso predicato $Append/3$) ma i letterali hanno argomenti **diversi**

Se tuttavia si applica una sostituzione $\sigma = [x/cons(a, k(a)), y/a]$ si ottiene

$\{Append(nil, cons(a,k(a)), cons(a,k(a)))\},\ \{\neg Append(nil, cons(a, k(a)), cons(a, k(a)))\}$

Da cui, per risoluzione, si ottiene la clausola vuota

La sostituzione σ si dice **unificatore** delle due clausole

va applicata integralmente a tutte e due le clausole da risolvere