Resolution in FOL[1]

Lecture 6

[1]The slides have been prepared using the textbook material available on the web, and the slides of the previous editions of the course by Prof. Luigia Carlucci Aiello

Summary

- unification [RN 9.2, algorithmic variant]
- conjunctive Normal Form and Skolemization [RN, 9.5], [different notation]
- resolution [RN 9.5]
- examples

A brief history of reasoning

450B.C.	Stoics	propositional logic, inference (maybe)
322B.C.	Aristotle	“syllogisms” (inf rules), quantifiers
1565	Cardano	probability theory (prop logic + uncertainty)
1847	Boole	propositional logic (again)
1879	Frege	first-order logic
1922	Wittgenstein	proof by truth tables
1930	Gödel	$\exists$ complete algorithm for FOL
1930	Herbrand	complete algorithm for FOL (reduce to prop)
1931	Gödel	$\neg\exists$ complete algorithm for arithmetic
1960	Davis/Putnam	“practical” algorithm for propositional logic
1965	Robinson	“practical” algorithm for FOL—resolution

Problems with function symbols

We need to extend to function symbols and, consequently, to **infinite** domains

Example: A theory for Natural Numbers
Constant symbols: 0;
Function Symbols: $succ(x)$
Predicate symbols: $<$

The set of terms includes:
$0, succ(0), succ(succ(0)), succ(succ(succ(0))), \ldots$

To instantiate variables we need to consider all of them!

Problems with Horn clauses

$$
\begin{array}{l}
\forall x \;\; (P(x) \Rightarrow Q(x)) \\
\forall x \;\; (\neg P(x) \Rightarrow R(x)) \\
\forall x \;\; (Q(x) \Rightarrow S(x)) \\
\forall x \;\; (R(x) \Rightarrow S(x))
\end{array}
\tag{1}
$$

We want to infer $S(A)$, but we can not express the theory in Horn clauses.

Deduction

In order to have a complete deduction method for full FOL:

- introduce **unification** of complex terms to handle variable instantiation
- generalize prop resolution and GMP to non-horn clauses
  - translate the formula in **conjunctive normal form** by eliminating existential quantifiers through **skolemization**;
  - define FOL **resolution**

Notice: there are other deduction methods e.g.

- **tableaux**
- **natural deduction**

- **Hilbert: MP + axioms**

Unification (general version)

Can $f(g(x, w)),\ f(g(z, h(y, w)))$ be unified ?

Let t_1 and t_2 be terms.
Is there a substitution σ such that $t_1\sigma = t_2\sigma$?
(i.e. $\textsc{Subst}(\sigma, t_1) = \textsc{Subst}(\sigma, t_1)$)

In other terms:
determine whether equation $t_1 = t_2$ can be solved.

A substitution σ is a **unifier** for t_1 and t_2 if $t_1\sigma = t_2\sigma$.

Terms t_1 and t_2 are **unifiable** if there exists a unifier.

Most general unifier (mgu)

Intuitively: the unifier that when applied gives the most general instance of the two expressions.

Example:
$\{x/b, y/b, z/a\}$ and $\{x/y, z/a\}$ are both unifiers of $p(a, x)$ and $p(z, y)$, but $\{x/y, z/a\}$ is more general than $\{x/b, y/b, z/a\}$.

A unifier σ is the **most general unifier – mgu (most general unifier)** for t_1 and t_2 if it is a unifier and it is more general of any other unifier for t_1 and t_2.

mgu is unique up to variable renaming.

The unification problem

Finding the MGU of two (or more) expressions (i.e. atoms or terms)

Example1:
The mgu of $P(a, w), P(x, f(b))$ is {x/a,w/f(b)}, obtained by solving the set of equations: {a=x,w=f(b)}

Example2:
$P(a, w), Q(x, f(b))$ are not unifiable because $P \neq Q$

Unification (summary)

1. $t_i = s_i$ success without any constraints for the unifier.
2. t_i is a variable: if t_i occurs in s_i then failure, else success with $t_i = s_i$ included in the unifier and **all** the occurrences of t_i are replaced by s_i.
3. s_i is a variable: symmetric of the previous one.
4. let t_i $f(tt_1, \ldots, tt_n)$ and s_i $g(ss_1, \ldots, ss_m)$ if $\neg(f = g) \vee \neg(n = m)$ then failure, else the pairs $< tt_1, ss_1 >, \ldots < tt_n, ss_n >$ must be unified.

To unify n-ary predicates, we consider a set of pairs $< t_1, s_2 >$ with t_i, s_i terms, that represent the equations obtained from the corresponding parameters (we use $choose$ to pick an element from the set and $rest$ returns the remaining elements).

Unification algorithm

Input: C set of pairs $< t_1, s_2 >$ with t_i, s_i terms
Output: most general unifier θ, if exists, otherwise false
begin
  $\theta := \{\}$;
  success := true;
  while not empty(C) and success **do**
  begin
    choose $< t_i, s_i >$ in C;
    if $t_i = s_i$ **then** C:=C/$\{< t_i, s_i >\}$
      else if var(t_i)
        then if occurs(t_i, s_i)
            then success:=false;
            else begin
              $\theta :=$ subst(θ, t_i, s_i) $\cup$ $\{t_i/s_i\}$;
              C:=subst(rest(C), t_i, s_i)
              end
        else if var(s_i)
          then if occurs(s_i, t_i)
            then success:=false;

else begin

$\theta :=$ subst$(\theta, s_i, t_i) \cup \{s_i/t_i\}$;

C:=subst(rest(C), s_i, t_i)

end

else if $t_i = f(tt_1, \ldots, tt_n)$ **and**

$s_i = g(ss_1, \ldots, ss_m)$ **and**

$f = g \wedge n = m$

then C:= rest(C) $\cup\{< tt_1, ss_1 >, \ldots < tt_n, ss_n >\}$

else success := false

end;

if not success **then** output false **else** output true, θ

end

NOTE

Unification $\neq$ pattern matching.

Pattern matching sometimes called **semi-unification**, since the variables can appear only in one of the two expressions.

Several development environments for "expert systems" use pattern matching and not unification.

Prolog uses unification, often omitting, for efficiency reasons, the **occur check**.

Skolemization

Recall the Existential Elimination rule:
trasform $\exists\, x\ P(x)$ in $P(A)$, where A is a constant symbol that does not appear elsewhere in the KB.

"Everyone has a heart":

$$\forall\, x\ \ (Person(x) \Rightarrow \exists\, y\ \ Heart(y) \wedge Has(x, y))$$

replace y with a constant H:

$$\forall\, x\ \ (Person(x) \Rightarrow Heart(H) \wedge Has(x, H))$$

says that everyone has the same heart H !!!

Skolem Functions

We need a function binding every person to a different heart.

$$\forall\, x \;\; (Person(x) \Rightarrow Heart(F(x)) \wedge has(x, F(x)))$$

where F is a function name that does not appear elsewhere in the KB.

F is called **Skolem Function**.

More generally, the existentially quantified variable is replaced by a term with a Skolem function whose arguments are all the quantified variables in whose scope the existential quantifier is.

Prenex Formulae 1

A formula ψ is *prenex form* if:

1. it does not contain quantifiers, namely:
   (a) no variables occur in ψ,
   (b) or ψ is *open*, and all variables are free;
2. or is of the form

$$\mathcal{Q}_1 x_1 \mathcal{Q}_2 x_2 \ldots \mathcal{Q}_n x_n A$$

where A is a quantifier free formula, $x_1, \ldots, x_n$ are variables, and $\mathcal{Q}_i \in \{\forall, \exists\}$ for $i = 1 \ldots n$.

$\mathcal{Q}_1 x_1 \mathcal{Q}_2 x_2 \ldots \mathcal{Q}_n x_n$ is the *prefix* and A is the *matrix*.

Prenex Formulae 2

Theorem:
For every formula ϕ there exists a prenex formula ψ such that:

$$\phi \equiv \psi \text{ and } var(\phi) = var(\psi)$$

The following equivalences for quantifiers are used in the transformation:

1. $\forall x(\phi \wedge \psi) \equiv \forall x\phi \wedge \forall x\psi$
2. $\exists x(\phi \vee \psi) \equiv \exists x\phi \vee \exists x\psi$
3. $\forall x(\phi(x) \vee \psi) \equiv \forall x\phi(x) \vee \psi \quad x \notin var(\psi)$
4. $\exists x(\phi(x) \wedge \psi) \equiv \exists x\phi(x) \wedge \psi \quad x \notin var(\psi)$
5. $\forall x(\neg\phi) \equiv \neg\exists x\phi$
6. $\exists x(\neg\phi) \equiv \neg\forall x\phi$

Transformation to prenex normal form

Let ϕ be a formula of $\mathcal{L}$, we first apply standard transformation:

1 build a formula ϕ^L where only $\wedge, \vee, \exists, \forall$, occur, where negation is pushed inwards and double negations are eliminated, such that $\vdash \phi \leftrightarrow \phi^L$.

2 Rename bound variables so that each quantifier uses a different variable.

Then:

3 build the prenex formula moving all quantifiers to the left

All the above transformations are validity preserving.

Skolemization

We apply Skolemization ϕ^{sko} to ϕ in prenex normal form:

- $(\exists x \phi)^{sko} = \phi[F(x_1, \ldots, x_n)/x]$, where $var(\phi) = \{x_1, \ldots, x_n\}$, and F is a new n-ary function symbol.

Given ϕ^P in prenex form:

4 Build ϕ^{sko} such that ϕ^{sko} is satisfiable iff ϕ^P is satisfiable. The result ϕ^{sko} is of the form $\overline{\forall x} \psi^-$, where ψ^- does not contain quantifiers.

Properties of Skolemization

Let ϕ be a formula and ϕ^{sko} be its Skolem normal form

$\mathcal{M}$ model of ϕ <u>DOES NOT imply</u> $\mathcal{M}$ model of ϕ^{sko}

$\mathcal{M}$ model of ϕ^{sko} <u>implies</u> $\mathcal{M}$ model of ϕ

Hence ϕ^{sko} valid <u>implies</u> ϕ valid

But ϕ valid <u>DOES NOT imply</u> ϕ^{sko} valid

Example:

$\exists x P(x) \vee \neg(\exists x P(x))$ is valid

$P(c) \vee \neg P(x)$ <u>IS NOT</u> valid

let $D = \{a, b\}$ and $P^I = \{b\}$ and c interpreted as a

Conjunctive Normal Form

Given ϕ^{sko} in prenex form (without existential quantifiers):

5 Get ψ^C, conjunctive normal form such that $\psi^C \equiv \psi^- \equiv \psi_1^- \wedge \ldots \wedge \psi_m^-$, and every ψ_i^- is an open clause.

6 Remove the universal quantifiers and transform the formula ψ^C into a set of clauses $S(\phi) = \{\psi_1^- \ldots \psi_m^-\}$.

The set of clauses $S(\phi)$ is also called **Skolem normal form** of ϕ.

Properties of the transformation

ϕ satisfiable <u>iff</u> ϕ^{sko} satisfiable
ϕ unsatisfiable <u>iff</u> ϕ^{sko} unsatisfiable
$\neg\phi$ valid <u>iff</u> $\neg(\phi^{sko})$ valid

Hence $\Gamma \cup \{\phi\} \vdash \bot$ <u>iff</u> $\Gamma \cup \{\phi^{sko}\} \vdash \bot$

Summarizing

ϕ and ϕ^{sko} are interchangeable for refutations.

Resolution

First-order version of **binary** resolution:

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n}{(\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)\theta}$$

where $\textsc{Unify}(\ell_i, \neg m_j) = \theta$.

For example,

$$\frac{\neg Rich(x) \vee Unhappy(x) \qquad\qquad Rich(Ken)}{Unhappy(Ken)}$$

with $\theta = \{x/Ken\}$

Example

$\neg P(w) \vee Q(w)$
$P(x) \vee R(x)$
$\neg Q(y) \vee S(y)$
$\neg R(z) \vee S(z)$

Can we now derive $S(a)$?

Generalized Resolution

Binary resolution is not complete for first order logic.

Let C_1 and C_2 be two clauses not containing common variables, L_i be literals for $i = 1, \ldots, n + m$ and $n, m \geq 1$ and σ be the *mgu* of $\{L_1, \ldots, L_{n+m}\}$

The clause $(C_1 \cup C_2)\sigma$ is obtained from $\{L_1, \ldots, L_n\} \cup C_1$ and $\{\neg L_{n+1}, \ldots, \neg L_{n+m}\} \cup C_2$ by means of a *Generalized Resolution* step.

Generalized Resolution Rule

$$\frac{\{L_1, \ldots, L_n\} \cup C_1 \;\; \{\neg L_{n+1}, \ldots, \neg L_{n+m}\} \cup C_2}{(C_1 \cup C_2)\sigma} \qquad (GLR)$$

where $\{L_1, \ldots, L_n\} \cup C_1$ and $\{\neg L_{n+1}, \ldots, \neg L_{n+m}\} \cup C_2$ are *parent clauses*. The clause $(C_1 \cup C_2)\sigma$ is called *resolvent* of $\{L_1, \ldots, L_n\} \cup C_1$ and $\{\neg L_{n+1}, \ldots, \neg L_{n+m}\} \cup C_2$ by σ.

GLR stands for $General\ Literal\ Resolution\ Rule$.

If $m = n = 1$ we get binary resolution.

Completeness of Resolution

Resolution is **refutation complete**:

Let S an unsatisfiable set of clauses, then a finite number of Resolution steps in S will lead to a contradiction.

Proof structure

Any set of sentences S is representable in clausal form

↓

Assume S is unsatisfiable, and in clausal form

↓ ← Herbrand's theorem

Some set S' of ground instances is unsatisfiable

↓ ← Ground resolution theorem

Resolution can find a contradiction in S'

↓ ← Lifting lemma

There is a resolution proof for the contradiction in S

Proof sketch

1. If S is unsatisfiable, then there exists a finite set of *ground instances* of S, that is unsatisfiable (Herbrand theorem).
2. Resolution is complete for ground formulae (Easy, since the set of consequences of a propositional theory is always finite)
3. **Lifting Lemma**: for every Resolution proof on a set of ground formulae, there exists a corresponding proof based on the non ground formulae that originated the ground ones.

Search strategies

Breadth first search is inefficient.

Depth-first search + heuristics:

♢ **unit resolution**: chooses clauses with only a literal

♢ **input resolution**: build the proof from α (or by the result of a previous step – **linear resolution**)

♢ **support set resolution**: chooses a **support set** and grows it always choosing a clause from it

Resolution Example: Colonel West

$(1) Amer(x) \land Weapon(y) \land Sells(x, y, z) \land Host(z) \Rightarrow Criminal(x)$

$(1)\{\neg Amer(x), \neg Weapon(y), \neg Sells(x, y, z), \neg Host(z), Criminal(x$

$(2) Owns(Nono, M_1)$ $(2)\{Owns(Nono, M_1)\}$

$(3) missil(M_1)$ $(3)\{missil(M_1)\}$

$(4) \forall x \;\; missil(x) \land Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$

$(4)\{\neg missil(x), \neg Owns(Nono, x), Sells(West, x, Nono)\}$

$(5) missil(x) \;\Rightarrow\; Weapon(x)$

$(5)\{\neg missil(x), Weapon(x)\}$

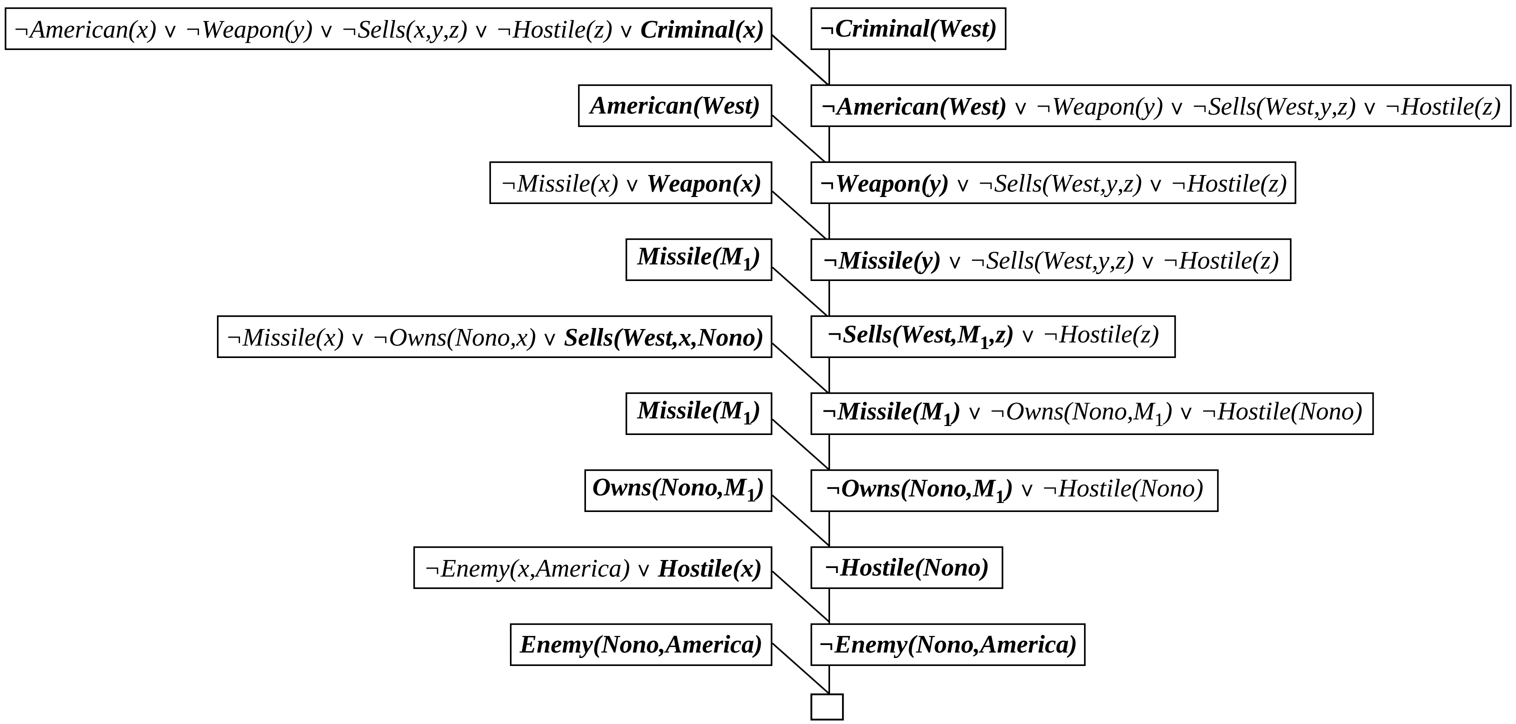
$(6) Enemy(x, America) \;\Rightarrow\; Hostile(x)$

$(6)\{\neg Enemy(x, America), Hostile(x)\}$

$(7) American(West)$ $(7)\{American(West)\}$

$(8) Enemy(Nono, America)$ $(8)\{Enemy(Nono, America)\}$

Resolution


¬American(x) ∨ ¬Weapon(y) ∨ ¬Sells(x,y,z) ∨ ¬Hostile(z) ∨ Criminal(x)
¬Criminal(West)
American(West)
¬American(West) ∨ ¬Weapon(y) ∨ ¬Sells(West,y,z) ∨ ¬Hostile(z)
¬Missile(x) ∨ Weapon(x)
¬Weapon(y) ∨ ¬Sells(West,y,z) ∨ ¬Hostile(z)
Missile(M_1)
¬Missile(y) ∨ ¬Sells(West,y,z) ∨ ¬Hostile(z)
¬Missile(x) ∨ ¬Owns(Nono,x) ∨ Sells(West,x,Nono)
¬Sells(West,M_1,z) ∨ ¬Hostile(z)
Missile(M_1)
¬Missile(M_1) ∨ ¬Owns(Nono,M_1) ∨ ¬Hostile(Nono)
Owns(Nono,M_1)
¬Owns(Nono,M_1) ∨ ¬Hostile(Nono)
¬Enemy(x,America) ∨ Hostile(x)
¬Hostile(Nono)
Enemy(Nono,America)
¬Enemy(Nono,America)

Example

Everyone who loves all animals is loved by someone.
Anyone who kills an animal is loved by no one.
Jack loves all animals.
Jack or Curiosity killed the cat, whose name is Tuna.
Did Curiosity kill the cat?

Formalization

Everyone who loves all animals is loved by someone.

$$\forall x \;\; [\forall y \;\; Animal(y) \Rightarrow Loves(x,y)] \Rightarrow \exists y \;\; Loves(y,x)$$

Anyone who kills an animal is loved by no one.

$$\forall x \;\; [\exists y \;\; Animal(y) \land Kills(x,y)] \Rightarrow \forall z \;\; \neg Loves(z,x)$$

$$\forall x \;\; Animal(x) \Rightarrow Loves(Jack,x)$$
$$kills(Jack,Tuna) \lor Kills(Curiosity,Tuna)$$
$$Cat(Tuna)$$
$$\forall x \;\; Cat(x) \Rightarrow Animal(x)$$

$$\neg Kills(Curiosity,Tuna)$$

Formalization

Everyone who loves all animals is loved by someone.

$$\forall x\ [\forall y\ Animal(y) \Rightarrow Loves(x,y)] \Rightarrow \exists y\ Loves(y,x)$$

$$\forall x\ [\neg(\forall y\ Animal(y) \Rightarrow Loves(x,y))] \vee \exists y\ Loves(y,x)$$

$$\forall x\ [\exists y\ \neg(Animal(y) \Rightarrow Loves(x,y))] \vee \exists y\ Loves(y,x)$$

$$\forall x\ [\exists y\ (Animal(y) \wedge \neg Loves(x,y)] \vee \exists y\ Loves(y,x)$$

$$\forall x\ \exists y\ \exists z\ [Animal(y) \wedge \neg Loves(x,y)] \vee Loves(z,x)$$

$$\forall x\ [Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee Loves(G(x),x)$$

A1. $Animal(F(x)) \vee Loves(G(x),x)$
A2. $\neg Loves(x,F(x)) \vee Loves(G(x),x)$

Formalization

Anyone who kills an animal is loved by no one.

$$\forall x \;\; [\exists y \;\; Animal(y) \wedge Kills(x,y)] \Rightarrow \forall z \;\; \neg Loves(z,x)$$
$$\forall x \;\; \forall y \;\; \forall z \;\; [Animal(y) \wedge Kills(x,y)] \Rightarrow \neg Loves(z,x)$$

B. $\neg Animal(y) \vee \neg Kills(x,y) \vee \neg Loves(z,x)$

C. $\neg Animal(x) \vee Loves(Jack,x)$
D. $kills(Jack,Tuna) \vee Kills(Curiosity,Tuna)$
E. $Cat(Tuna)$
F. $\neg Cat(x) \vee Animal(x)$

$\neg Kills(Curiosity,Tuna)$

Resolution proof