First-Order Logic

Peter Baumgartner

http://users.cecs.anu.edu.au/~baumgart/

Ph: 02 6218 3717

Data61/CSIRO and ANU

July 2017

First-Order Logic (FOL)

Recall: propositional logic: variables are statements ranging over {true/false}

$$\frac{\text{SocratesIsHuman} \qquad \text{SocratesIsHuman} \rightarrow \text{SocratesIsMortal}}{\text{SocratesIsMortal}}$$

FOL: variables range over individual objects

$$\frac{\text{Human}(\text{socrates}) \qquad \forall x.\ (\text{Human}(x) \rightarrow \text{Mortal}(x))}{\text{Mortal}(\mathit{socrates})}$$

In these lectures:

- (Syntax and) semantics of FOL
- Normal forms
- Reasoning: tableau calculus, resolution calculus

First-Order Logic (FOL)

Also called Predicate Logic or Predicate Calculus

FOL Syntax

variables	$x, y, z, \cdots$
constants	$a, b, c, \cdots$
functions	$f, g, h, \cdots$
terms	variables, constants or
	n-ary function applied to n terms as arguments
	$a, x, f(a), g(x, b), f(g(x, g(b)))$
predicates	$p, q, r, \cdots$
atom	$\top$, $\bot$, or an n-ary predicate applied to n terms
literal	atom or its negation
	$p(f(x), g(x, f(x))), \quad \neg p(f(x), g(x, f(x)))$

Note: 0-ary functions: constant

0-ary predicates: $P, Q, R, \ldots$

quantifiers

existential quantifier $\exists x. F[x]$

“there exists an x such that $F[x]$”

universal quantifier $\forall x. F[x]$

“for all x, $F[x]$”

FOL formula literal, application of logical connectives ($\neg$, $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$) to formulae,
or application of a quantifier to a formula

Example

FOL formula

$$\forall x.\ \underbrace{p(f(x), x)\ \rightarrow\ (\exists y.\ \underbrace{p(f(g(x,y)), g(x,y))}_{G})\ \wedge\ q(x, f(x))}_{F}$$

The scope of $\forall x$ is F.
The scope of $\exists y$ is G.
The formula reads:
"for all x,
if $p(f(x), x)$
then there exists a y such that
$p(f(g(x,y)), g(x,y))$ and $q(x, f(x))$"

An occurrence of x within the scope of $\forall x$ or $\exists x$ is bound, otherwise it is free.

Translations of English Sentences into FOL

- The length of one side of a triangle is less than the sum of the lengths of the other two sides

$$\forall x, y, z.\ \mathit{triangle}(x, y, z)\ \rightarrow\ \mathit{length}(x) < \mathit{length}(y) + \mathit{length}(z)$$

- Fermat's Last Theorem.

$$\begin{array}{l}
\forall n.\ \mathit{integer}(n)\ \wedge\ n > 2 \\
\quad \rightarrow\ \forall x, y, z. \\
\quad\quad \mathit{integer}(x)\ \wedge\ \mathit{integer}(y)\ \wedge\ \mathit{integer}(z) \\
\quad\quad \wedge\ x > 0\ \wedge\ y > 0\ \wedge\ z > 0 \\
\quad\quad\quad \rightarrow\ x^n + y^n \neq z^n
\end{array}$$

FOL Semantics

An interpretation $I : (D_I, \alpha_I)$ consists of:

- Domain D_I
  non-empty set of values or objects
  for example $D_I =$ playing cards (finite),
  integers (countably), or
  reals (uncountably infinite)
- Assignment α_I
  - each variable x assigned value $\alpha_I[x] \in D_I$
  - each n-ary function f assigned

    $$\alpha_I[f] : \ D_I^n \to D_I$$

    In particular, each constant a (0-ary function) assigned value $\alpha_I[a] \in D_I$
  - each n-ary predicate p assigned

    $$\alpha_I[p] : \ D_I^n \to \{\text{true}, \ \text{false}\}$$

    In particular, each propositional variable P (0-ary predicate) assigned truth value (true, false)

Example

$$F: \; p(f(x,y),z) \;\rightarrow\; p(y, g(z,x))$$

Interpretation $I : (D_I, \alpha_I)$

$D_I = \mathbb{Z} = \{\cdots, -2, -1, 0, 1, 2, \cdots\}$ integers

$\alpha_I[f]: \quad D_I^2 \mapsto D_I$, $\quad (x,y) \mapsto x+y$

$\alpha_I[g]: \quad D_I^2 \mapsto D_I$, $\quad (x,y) \mapsto x-y$

$\alpha_I[p]: \quad D_I^2 \mapsto \{\text{true}, \text{false}\}$

$$(x,y) \mapsto \begin{cases} \text{true} & \text{if } x < y \\ \text{false} & \text{otherwise} \end{cases}$$

Also $\alpha_I[x] = 13$, $\alpha_I[y] = 42$, $\alpha_I[z] = 1$

Compute the truth value of F under I

1. $I \;\not\models\; p(f(x,y),z)$ since $13 + 42 \geq 1$
2. $I \;\not\models\; p(y, g(z,x))$ since $42 \geq 1 - 13$
3. $I \;\models\; F$ by 1, 2, and $\rightarrow$

F is true under I

Semantics: Quantifiers

Let x be a variable.
An <u>x-variant</u> of interpretation I is an interpretation $J : (D_J, \alpha_J)$ such that

- $D_I = D_J$
- $\alpha_I[y] = \alpha_J[y]$ for all symbols y, except possibly x

That is, I and J agree on everything except possibly the value of x

Denote

$$J : \; I \triangleleft \{x \mapsto \mathrm{v}\}$$

the x-variant of I in which $\alpha_J[x] = \mathrm{v}$ for some $\mathrm{v} \in D_I$. Then

- $I \models \forall x.\ F$ iff for all $\mathrm{v} \in D_I$, $I \triangleleft \{x \mapsto \mathrm{v}\} \models F$
- $I \models \exists x.\ F$ iff there exists $\mathrm{v} \in D_I$ s.t. $I \triangleleft \{x \mapsto \mathrm{v}\} \models F$

Example

Consider

$$F : \forall x.\ \exists y.\ 2 \cdot y = x$$

Here $2 \cdot y$ is the infix notation of the term $\cdot(2, y)$,
and $2 \cdot y = x$ is the infix notation of the atom $=(\cdot(2, y), x)$

- 2 is a 0-ary function symbol (a constant).
- $\cdot$ is a 2-ary function symbol.
- $=$ is a 2-ary predicate symbol.
- x, y are variables.

What is the truth-value of F?

Example ($\mathbb{Z}$)

$$F : \forall x.\ \exists y.\ 2 \cdot y = x$$

Let I be the standard interpretation for integers, $D_I = \mathbb{Z}$.
Compute the value of F under I:

$$
\begin{array}{rcr}
I & \models & \forall x.\ \exists y.\ 2 \cdot y = x \\
\text{iff} & & \\
\text{for all } \mathrm{v} \in D_I,\ I \triangleleft \{x \mapsto \mathrm{v}\} & \models & \exists y.\ 2 \cdot y = x \\
\text{iff} & & \\
\text{for all } \mathrm{v} \in D_I, & & \\
\text{there exists } \mathrm{v}_1 \in D_I,\ I \triangleleft \{x \mapsto \mathrm{v}\} \triangleleft \{y \mapsto \mathrm{v}_1\} & \models & 2 \cdot y = x
\end{array}
$$

The latter is false since for $1 \in D_I$ there is no number v_1 with $2 \cdot \mathrm{v}_1 = 1$.

Example ($\mathbb{Q}$)

$$F : \forall x.\ \exists y.\ 2 \cdot y = x$$

Let I be the standard interpretation for rational numbers, $D_I = \mathbb{Q}$.
Compute the value of F under I:

$$I \models \forall x.\ \exists y.\ 2 \cdot y = x$$

iff

$$\text{for all } \mathsf{v} \in D_I,\ I \triangleleft \{x \mapsto \mathsf{v}\} \models \exists y.\ 2 \cdot y = x$$

iff

for all $\mathsf{v} \in D_I$,
there exists $\mathsf{v}_1 \in D_I$, $I \triangleleft \{x \mapsto \mathsf{v}\} \triangleleft \{y \mapsto \mathsf{v}_1\} \models 2 \cdot y = x$

The latter is true since for arbitrary $v \in D_I$ we can chose v_1 with $\mathsf{v}_1 = \frac{\mathsf{v}}{2}$.

Satisfiability and Validity

F is satisfiable iff there exists an interpretation I such that $I \models F$.

F is valid iff for all interpretations I, $I \models F$.

Note: F is valid iff $\neg F$ is unsatisfiable.

Example

$$F : \; (\forall x.\; p(x,x)) \;\rightarrow\; (\exists x.\; \forall y.\; p(x,y)) \quad \text{is invalid.}$$

How to show this?
Find interpretation I such that

$$I \;\models\; \neg((\forall x.\; p(x,x)) \;\rightarrow\; (\exists x.\; \forall y.\; p(x,y)))$$

i.e.

$$I \;\models\; (\forall x.\; p(x,x)) \;\wedge\; \neg(\exists x.\; \forall y.\; p(x,y))$$

Choose $D_I = \{0, 1\}$
$p_I = \{(0,0),\ (1,1)\}$ i.e. $p_I(0,0)$ and $p_I(1,1)$ are true
$p_I(0,1)$ and $p_I(1,0)$ are false

I falsifying interpretation $\;\Rightarrow\;$ F is invalid.

Example

$$F : \ (\forall x.\ p(x)) \ \leftrightarrow \ (\neg \exists x.\ \neg p(x)) \quad \text{is valid.}$$

How to show this?

1. By expanding definitions. This is easy for *this* example.
2. By constructing a proof with, e.g., a “semantic argument method” adapted to FOL.

Below we will develop such a semantic argument method adapted to FOL. To define it, we first need the concept of “substitutions”.

Substitution

Suppose we want to replace terms with other terms in formulas, e.g.,

$$F : \forall y.\ (p(x, y) \ \rightarrow \ p(y, x))$$

should be transformed to

$$G : \forall y.\ (p(a, y) \ \rightarrow \ p(y, a))$$

We call the mapping from x to a a substitution, denoted as $\sigma : \{x \mapsto a\}$.

We write $F\sigma$ for the Formula G.

Another convenient notation is $F[x]$ for a formula containing the variable x and $F[a]$ for $F\sigma$.

Substitution

A substitution σ is a mapping from variables to terms, written as

$$\sigma : \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$$

such that $n \geq 0$ and $x_i \neq x_j$ for all $i, j = 1..n$ with $i \neq j$.

The set $\mathrm{dom}(\sigma) = \{x_1, \ldots, x_n\}$ is called the domain of σ.

The set $\mathrm{cod}(\sigma) = \{t_1, \ldots, t_n\}$ is called the codomain of σ. The set of all variables occurring in $\mathrm{cod}(\sigma)$ is called the variable codomain of σ, denoted by $\mathrm{varcod}(\sigma)$.

By $F\sigma$ we denote the application of σ to the formula F, i.e., the formula F where all free occurrences of x_i are replaced by t_i.

For a formula named $F[x]$ we write $F[t]$ as a shorthand for $F[x]\{x \mapsto t\}$.

Safe Substitution

Care has to be taken in presence of quantifiers:

$$F[x] : \exists y.\ y = \mathsf{Succ}(x)$$

What is $F[y]$? We cannot just rename x to y with $\{x \mapsto y\}$:

$$F[y] : \exists y.\ y = \mathsf{Succ}(y) \qquad \text{Wrong!}$$

We need to first <u>rename</u> bound variables occuring in the codomain of the substitution:

$$F[y] : \exists y'.\ y' = \mathsf{Succ}(y) \qquad \text{Right!}$$

Renaming does not change the models of a formula:

$$(\exists y.\ y = \mathsf{Succ}(x)) \ \Leftrightarrow\ (\exists y'.\ y' = \mathsf{Succ}(x))$$

Recursive Definition of Substitution

$$t\sigma = \begin{cases} \sigma(x) & \text{if } t = x \text{ and } x \in \mathrm{dom}(\sigma) \\ x & \text{if } t = x \text{ and } x \notin \mathrm{dom}(\sigma) \\ f(t_1\sigma, \ldots, t_n\sigma) & \text{if } t = f(t_1, \ldots, t_n) \end{cases}$$

$$p(t_1, \ldots, t_n) = p(t_1\sigma, \ldots, t_n\sigma)$$

$$(\neg F)\sigma = \neg(F\sigma)$$

$$(F \;\wedge\; G)\sigma = (F\sigma \;\wedge\; G\sigma)$$

$$\ldots$$

$$(\forall x.\ F)\sigma = \begin{cases} \forall x'.\ (F\{x \mapsto x'\})\sigma & \text{if } x \in \mathrm{dom}(\sigma) \cup \mathrm{varcod}(\sigma),\ x' \text{ is fres} \\ \forall x.\ F\sigma & \text{otherwise} \end{cases}$$

$$(\exists x.\ F)\sigma = \begin{cases} \exists x'.\ (F\{x \mapsto x'\})\sigma & \text{if } x \in \mathrm{dom}(\sigma) \cup \mathrm{varcod}(\sigma),\ x' \text{ is fres} \\ \exists x.\ F\sigma & \text{otherwise} \end{cases}$$

Example: Safe Substitution $F\sigma$

$$F : (\forall x.\ \overbrace{p(x, y)}^{\text{scope of } \forall x}\ ) \ \rightarrow\ q(f(y), x)$$

bound by $\forall x$ $\nearrow$ $\nwarrow$ free free $\nearrow$ $\nwarrow$ free

$$\sigma : \{x \mapsto g(x, y),\ y \mapsto f(x)\}$$

$F\sigma$?

1. Rename x to x' in $(\forall x.\ p(x, y))$, as $x \in \text{varcod}(\sigma) = \{x, y\}$:

   $$F' : (\forall x'.\ p(x', y)) \ \rightarrow\ q(f(y), x)$$

   where x' is a fresh variable.
2. Apply σ to F':

   $$F\sigma : (\forall x'.\ p(x', f(x))) \ \rightarrow\ q(f(f(x)), g(x, y))$$

Semantic Argument ("Tableau Calculus")

Recall rules from propositional logic:

$$\frac{I \models \neg F}{I \not\models F} \qquad\qquad \frac{I \not\models \neg F}{I \models F}$$

$$\frac{I \models F \wedge G}{\begin{array}{l} I \models F \\ I \models G \end{array}} \leftarrow\text{and} \qquad\qquad \frac{I \not\models F \wedge G}{I \not\models F \quad \underset{\nwarrow\text{or}}{\mid} \quad I \not\models G}$$

$$\frac{I \models F \vee G}{I \models F \quad \mid \quad I \models G} \qquad\qquad \frac{I \not\models F \vee G}{\begin{array}{l} I \not\models F \\ I \not\models G \end{array}}$$

$$\frac{I \models F \rightarrow G}{I \not\models F \quad \mid \quad I \models G} \qquad\qquad \frac{I \not\models F \rightarrow G}{\begin{array}{l} I \models F \\ I \not\models G \end{array}}$$

$$\frac{I \models F \leftrightarrow G}{I \models F \wedge G \quad \mid \quad I \not\models F \vee G} \qquad\qquad \frac{I \not\models F \leftrightarrow G}{I \models F \wedge \neg G \quad \mid \quad I \models \neg F \wedge G}$$

$$\frac{\begin{array}{l} I \models F \\ I \not\models F \end{array}}{I \models \bot}$$

Semantic Argument for FOL

The following additional rules are used for quantifiers.

(The formula $F[t]$ is obtained from $F[x]$ by application of the substitution $\{x \mapsto t\}$.)

$$\frac{I \models \forall x.\ F[x]}{I \models F[t]} \text{ for any term } t \qquad \frac{I \not\models \forall x.\ F[x]}{I \not\models F[a]} \text{ for a fresh constant } a$$

$$\frac{I \models \exists x.\ F[x]}{I \models F[a]} \text{ for a fresh constant } a \qquad \frac{I \not\models \exists x.\ F[x]}{I \not\models F[t]} \text{ for any term } t$$

(We assume there are infinitely many constant symbols.)

Example

Show that $(\exists x.\ \forall y.\ p(x,y)) \rightarrow (\forall x.\ \exists y.\ p(y,x))$ is valid.

Assume otherwise.
That is, assume I is a falsifying interpretation for this formula.

1.	I	$\not\models$	$(\exists x.\ \forall y.\ p(x,y)) \rightarrow (\forall x.\ \exists y.\ p(y,x))$	assumption
2.	I	$\models$	$\exists x.\ \forall y.\ p(x,y)$	1 and $\rightarrow$
3.	I	$\not\models$	$\forall x.\ \exists y.\ p(y,x)$	1 and $\rightarrow$
4.	I	$\models$	$\forall y.\ p(a,y)$	2 and $\exists$ ($x \mapsto a$ fresh)
5.	I	$\not\models$	$\exists y.\ p(y,b)$	3 and $\forall$ ($x \mapsto b$ fresh)
6.	I	$\models$	$p(a,b)$	4 and $\forall$ ($y \mapsto b$)
7.	I	$\not\models$	$p(a,b)$	5 and $\exists$ ($y \mapsto a$)
8.	I	$\models$	$\perp$	6 and 7

Thus, the formula is valid.

Example

Is $F : (\forall x.\ p(x,x)) \to (\exists x.\ \forall y.\ p(x,y))$ is valid?

Assume I is a falsifying interpretation for F.

1.	I	$\not\models$	$(\forall x.\ p(x,x)) \to (\exists x.\ \forall y.\ p(x,y))$	assumption
2.	I	$\models$	$\forall x.\ p(x,x)$	1 and $\to$
3.	I	$\not\models$	$\exists x.\ \forall y.\ p(x,y)$	1 and $\to$
4.	I	$\models$	$p(a_1,a_1)$	2 and $\forall$ $(x \mapsto a_1)$
5.	I	$\not\models$	$\forall y.\ p(a_1,y)$	3 and $\exists$ $(x \mapsto a_1)$
6.	I	$\not\models$	$p(a_1,a_2)$	5 and $\forall$ $(y \mapsto a_2$ fresh$)$
7.	I	$\models$	$p(a_2,a_2)$	2 and $\forall$ $(x \mapsto a_2)$
8.	I	$\not\models$	$\forall y.\ p(a_2,y)$	3 and $\exists$ $(x \mapsto a_2)$
9.	I	$\not\models$	$p(a_2,a_3)$	8 and $\forall$ $(y \mapsto a_3$ fresh$)$
	$\dots$			

No contradiction. Falsifying interpretation I can be "read" from derivation:

$$D_I = \mathbb{N}, \qquad p_I(x,y) = \begin{cases} \text{true} & \text{if } y = x \\ \text{false} & \text{if } y = x+1 \\ \text{arbitrary} & \text{otherwise} \end{cases}$$

Semantic Argument Proof

To show that FOL formula F is valid, assume $I \not\models F$ and derive a contradiction $I \models \bot$ in all branches.

It holds:

- Soundness
  If every branch of a semantic argument proof reaches $I \models \bot$ then F is valid.
- Completeness
  Every valid formula F has a semantic argument proof in which every branch reaches $I \models \bot$.
- Non-termination
  For an invalid formula F the method is not guaranteed to terminate. In other words, the semantic argument method is not a decision procedure for validity.

Soundness (Proof Sketch)

Given a formula F, the semantic argument method begins with

$$I \not\models F \quad \text{assumption}$$

Suppose that F is not valid, i.e., there is an interpretation I such that the above assumption holds.

By following the semantic argument steps, one can show that each step preserves satisfiability. (For or-nodes, one new branch will be satisfiable.)

This may require updating the current interpretation I. The interpretation I' obtained in the next step may differ in the values $\alpha_I[a_i]$ for fresh constants a_i.

Because the new branch (or one of the new branches, for or-nodes) is satisfiable, it is impossible to reach $\perp$ in *every* branch. This proves the soundness claim (in its contrapositive form).

Completeness (Proof Sketch)

Without loss of generality assume that F has no free variables.
(If so, replace these by fresh constants.)

A ground term is a term without variables.

Consider (finite or infinite) proof trees starting with $I \not\models F$.
We assume fairness:

- All possible proof rules were applied in all non-closed branches.
- The $\forall$ and $\exists$ rules were applied for all ground terms.
  This is possible since the terms are countable.

If every branch is closed, the tree is finite (König's Lemma) and we have a (finite) proof for F.

Completeness (Proof Sketch)

Otherwise the tree has at least one open (possibly infinite) branch P.
We show that F is not valid by extracting from P an interpretation I such that $I \not\models F$, the statement in the root of the proof .

1. The statements on that branch P form a <u>Hintikka set</u>:
   - $I \models F \wedge G \in P$ implies $I \models F \in P$ and $I \models G \in P$.
   - $I \not\models F \wedge G \in P$ implies $I \not\models F \in P$ or $I \not\models G \in P$.
   - $I \models \forall x. F[x] \in P$ implies for all ground terms t, $I \models F[t] \in P$.
   - $I \not\models \forall x. F[x] \in P$ implies for some fresh constant a, $I \not\models F[a] \in P$.
   - Similarly for $\neg$, $\rightarrow$, $\leftrightarrow$ and $\exists$.
2. Choose $D_I := \{t \mid t \text{ is a ground term}\}$
3. Choose $\alpha_I[f](t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$,
   $$\alpha_I[p](t_1, \ldots, t_n) = \begin{cases} \text{true} & \text{if } I \models p(t_1, \ldots, t_n) \in P \\ \text{false} & \text{otherwise} \end{cases}$$
4. I is such that all statements on the branch P hold true.
   In particular $I \not\models F$ in the root, thus F is not valid.

Proof of Item (4)

Item (4) on the previous slide stated more precsisely:

(4.1) if $I \models F \in P$ then $I \models F$, and

(4.2) if $I \not\models F \in P$ then $I \not\models F$, where $I = (D_i, \alpha_i)$ as constructed.

Define an ordering $\succ$ on formulas as follows:

- $F \circ G \ \succ\ F$ and $F \circ G \ \succ\ G$ for $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.
- $\neg F \ \succ\ F$.
- $\forall x.F[x] \ \succ\ F[t]$ and $\exists x.F[x] \ \succ\ F[t]$ for any term t.

Clearly, $\succ$ is a well-founded strict ordering
($\succ$ is irreflexive, transitive and there are no infinite chains).

Prove (4) by induction: let $I \models F \in P$ or $I \not\models F \in P$.

Base case: F is an atom. Directely prove $I \models F$ or $I \not\models F$, respectively.

Induction case: F is of the form $F_1 \circ F_2$, $\neg F_1$, $\forall x.F_1[x]$ or $\exists x.F_1[x]$.
Induction hypotheses: (4) holds for all G with $F \succ G$.
Prove it follows $I \models F$ or $I \not\models F$, respectively.

Proof of Item (4) – Base Case

<u>Case $I \models F \in P$:</u> We show it follows $I \models F$. (*)

<u>Case 1:</u> $F = Q$, for some (ground) atom Q.

That is, $I \models Q \in P$.

By construction of I it follows $I \models Q$.

<u>Case 2:</u> $F = \top$.

That is, $I \models \top \in P$.

Trivial (every interpretation satisfies $\top$ by definition).

<u>Case 3:</u> $F = \bot$.

That is, $I \models \bot \in P$.

This case is impossible as P is open ($I \models \bot \notin P$).

Proof of Item (4) – Induction Case

Case $I \models F \in P$: We show it follows $I \models F$. (*)

Case 1: $F = F_1 \land F_2$, for some F_1 and F_2.

That is, $I \models F_1 \land F_2 \in P$

By Hintikka set, $I \models F_1 \in P$ and $I \models F_2 \in P$.

By induction hypothesis, $I \models F_1$ and $I \models F_2$.

By semantics of $\land$, $I \models F_1 \land F_2$.

Case 2: $F = \neg F_1$, for some F_1.

That is, $I \models \neg F_1 \in P$

By Hintikka set, $I \not\models F_1 \in P$.

By induction hypothesis, $I \not\models F_1$.

By semantics of $\neg$, $I \models \neg F_1$.

Other cases for propositional operators: similar

Proof of Item (4) – Induction Case

<u>Case $I \models F \in P$:</u> We show it follows $I \models F$. (*)

<u>Case 3:</u> $F = \forall x.F_1[x]$, for some F_1.

That is, $I \models \forall x.F_1[x] \in P$.

For every ground term $t \in D_I$ it holds:

  By Hintikka set $I \models F_1[t] \in P$.

  By induction hypothesis $I \models F_1[t]$.

  Because t evaluates to t under I we have $I \lhd \{x \mapsto t\} \models F_1[x]$.

By semantics of $\forall$ it follows $I \models \forall x.F_1[x]$.

Proof of Item (4) – Induction Case

Case $I \models F \in P$: We show it follows $I \models F$. (*)

Case 4: $F = \exists x. F_1[x]$, for some F_1.

That is, $I \models \exists x. F_1[x] \in P$.

By Hintikka set $I \models F_1[a] \in P$ for some (fresh) constant a.

By induction hypothesis $I \models F_1[a]$.

Because a evaluates to a under I it follows $I \lhd \{x \mapsto a\} \models F_1[x]$.

By semantics of $\exists$ it follows $I \models \exists x. F_1[x]$.

Case $I \not\models F \in P$:

The proof of $I \not\models F$ is analogous to the case $I \models F \in P$.

QED

Normal Forms

Also in first-order logic normal forms can be used:

- Devise an algorithm to convert a formula to a normal form.

  Example: CNF (conjunctive normal form)

- Then devise a procedure for satisfiability/validity that only works on the normal form

  Example: both DPLL and the resolution calculus require CNF formulas as input

Negation Normal Form (NNF)

<u>NNF</u>: Negations appear only in literals, and use only $\neg$, $\wedge$, $\vee$,$\forall$, $\exists$.

To transform F to equivalent F' in NNF use recursively
the following template equivalences (left-to-right).

From propositional logic:

$$\neg\neg F_1 \Leftrightarrow F_1 \qquad \neg\top \Leftrightarrow \bot \qquad \neg\bot \Leftrightarrow \top$$

$$\left.\begin{array}{l} \neg(F_1 \wedge F_2) \Leftrightarrow \neg F_1 \vee \neg F_2 \\ \neg(F_1 \vee F_2) \Leftrightarrow \neg F_1 \wedge \neg F_2 \end{array}\right\}\text{De Morgan's Law}$$

$$F_1 \rightarrow F_2 \Leftrightarrow \neg F_1 \vee F_2$$

$$F_1 \leftrightarrow F_2 \Leftrightarrow (F_1 \rightarrow F_2) \wedge (F_2 \rightarrow F_1)$$

Additionally for first-order logic:

$$\neg\forall x.\ F[x] \Leftrightarrow \exists x.\ \neg F[x]$$

$$\neg\exists x.\ F[x] \Leftrightarrow \forall x.\ \neg F[x]$$

Example: Conversion to NNF

$$G : \forall x.\ (\exists y.\ p(x,y)\ \wedge\ p(x,z))\ \rightarrow\ \exists w. p(x,w)\ .$$

1. $\forall x.\ (\exists y.\ p(x,y)\ \wedge\ p(x,z))\ \rightarrow\ \exists w.\ p(x,w)$
2. $\forall x.\ \neg(\exists y.\ p(x,y)\ \wedge\ p(x,z))\ \vee\ \exists w.\ p(x,w)$

   $F_1\ \rightarrow\ F_2\ \Leftrightarrow\ \neg F_1\ \vee\ F_2$
3. $\forall x.\ (\forall y.\ \neg(p(x,y)\ \wedge\ p(x,z)))\ \vee\ \exists w.\ p(x,w)$

   $\neg\exists x.\ F[x]\ \Leftrightarrow\ \forall x.\ \neg F[x]$
4. $\forall x.\ (\forall y.\ \neg p(x,y)\ \vee\ \neg p(x,z))\ \vee\ \exists w.\ p(x,w)$

Prenex Normal Form (PNF)

PNF: All quantifiers appear at the beginning of the formula

$$Q_1 x_1 \cdots Q_n x_n .\ F[x_1, \cdots, x_n]$$

where $Q_i \in \{\forall,\ \exists\}$ and F is quantifier-free.

Every FOL formula F can be transformed to formula F' in PNF such that $F' \ \Leftrightarrow\ F$.

1. Transform F to NNF
2. Rename quantified variables to fresh names
3. Move all quantifiers to the front

$$\begin{array}{ll} (\forall x\ F)\ \vee\ G\ \Leftrightarrow\ \forall x\ (F\ \vee\ G) & (\exists x\ F)\ \vee\ G\ \Leftrightarrow\ \exists x\ (F\ \vee\ G) \\ (\forall x\ F)\ \wedge\ G\ \Leftrightarrow\ \forall x\ (F\ \wedge\ G) & (\exists x\ F)\ \wedge\ G\ \Leftrightarrow\ \exists x\ (F\ \wedge\ G) \end{array}$$

These rules apply modulo symmetry of $\wedge$ and $\vee$

Example: PNF 1

Find equivalent PNF of

$$F : \forall x.\ ((\exists y.\ p(x,y)\ \land\ p(x,z))\ \rightarrow\ \exists y.\ p(x,y))$$

1. Transform F to NNF

$$F_1 : \forall x.\ (\forall y.\ \neg p(x,y)\ \lor\ \neg p(x,z))\ \lor\ \exists y.\ p(x,y)$$

2. Rename quantified variables to fresh names

$$F_2 : \forall x.\ (\forall y.\ \neg p(x,y)\ \lor\ \neg p(x,z))\ \lor\ \exists w.\ p(x,w)$$

$\uparrow$ in the scope of $\forall x$

Example: PNF 2

3. Add the quantifiers before F_2

$$F_3 :\ \forall x.\ \forall y.\ \exists w.\ \neg p(x,y)\ \vee\ \neg p(x,z)\ \vee\ p(x,w)$$

Alternately,

$$F_3' :\ \forall x.\ \exists w.\ \forall y.\ \neg p(x,y)\ \vee\ \neg p(x,z)\ \vee\ p(x,w)$$

<u>Note</u>: In F_3, $\forall y$ is <u>in the scope</u> of $\forall x$, therefore the order of quantifiers must be $\cdots \forall x \cdots \forall y \cdots$

$$\boxed{F_3\ \Leftrightarrow\ F \text{ and } F_3'\ \Leftrightarrow\ F}$$

<u>Note</u>: However $G\ \not\Leftrightarrow\ F$

$$G :\ \forall y.\ \exists w.\ \forall x.\ \neg p(x,y)\ \vee\ \neg p(x,z)\ \vee\ p(x,w)$$

Skolem Normal Form (SNF)

<u>SNF</u>: PNF and additionally all quantifiers are $\forall$

$$\forall x_1 \cdots \forall x_n.\ F[x_1, \cdots, x_n] \quad \text{where } F \text{ is quantifier-free.}$$

Every FOL formula F can be transformed to equi-satisfiable formula F' in SNF.

1. Transform F to NNF
2. Transform to PNF
3. Starting from the left, stepwise remove all $\exists$-quantifiers by <u>Skolemization</u>

Skolemization

Replace

$$\underbrace{\forall x_1 \cdots \forall x_{k-1}}_{\text{no } \exists} .\ \exists x_k .\ \underbrace{Q_{k+1} x_{k+1} \cdots Q_n x_n}_{Q_i \in \{\forall, \exists\}} .\ F[x_1, \cdots, x_k, \cdots, x_n]$$

by

$$\forall x_1 \cdots \forall x_{k-1} .\ Q_{k+1} x_{k+1} \cdots Q_n x_n .\ F[x_1, \cdots, t, \cdots, x_n]$$

where

$t = f(x_1, \ldots, x_{k-1})$ where f is a fresh function symbol

The term t is called a Skolem term for x_k and f is called a Skolem function symbol.

Example: SNF

Convert

$$F_3 : \ \forall x.\ \forall y.\ \exists w.\ \neg p(x, y) \ \lor \ \neg p(x, z) \ \lor \ p(x, w)$$

to SNF.

Let $f(x, y)$ be a Skolem term for w:

$$F_4 : \ \forall x.\ \forall y.\ \neg p(x, y) \ \lor \ \neg p(x, z) \ \lor \ p(x, f(x, y))$$

We have $F_3 \not\Leftrightarrow F_4$ however it holds

$$\boxed{\text{A formula } F \text{ is satisfiable iff the SNF of } F \text{ is satisfiable.}}$$

Conjunctive Normal Form

CNF: Conjunction of disjunctions of literals

$$\bigwedge_i \bigvee_j \ell_{i,j} \quad \text{for literals } \ell_{i,j}$$

Every FOL formula can be transformed into equi-satisfiable CNF.

1. Transform F to NNF
2. Transform to PNF
3. Transform to SNF
4. Leave away $\forall$-quantifiers (This is just a convention)
5. Use the following template equivalences (left-to-right):

$$\begin{array}{rcl} (F_1 \land F_2) \lor F_3 & \Leftrightarrow & (F_1 \lor F_3) \land (F_2 \lor F_3) \\ F_1 \lor (F_2 \land F_3) & \Leftrightarrow & (F_1 \lor F_2) \land (F_1 \lor F_3) \end{array}$$

Example: CNF

Convert

$$F_4 : \ \forall x.\, \forall y.\, \neg p(x, y) \ \vee \ \neg p(x, z) \ \vee \ p(x, f(x, y))$$

to CNF.

Leave away $\forall$-quantifiers

$$F_5 : \ \neg p(x, y) \ \vee \ \neg p(x, z) \ \vee \ p(x, f(x, y))$$

F_5 is already in CNF.

Conversion from SNF to CNF is again an equivalence transformation.

Resolution for FOL

We have seen the resolution calculus for propositional logic.

(Refinements of) the resolution calculus for FOL are the best methods for automated proof search in FOL.

Plan for generalization of propositional resolution to FOL:

1. First-order clause logic
2. Unification
3. FOL resolution inference rules

First-order Clause Logic: Syntax

CNF as clause sets

$$\underbrace{\bigwedge_i \underbrace{\bigvee_j \ell_{i,j} \quad \text{for literals } \ell_{i,j}}_{\text{Clause}}}_{\text{Taken as a clause set } N}$$

<u>Example</u>

$$N = \{P(a),\ \neg P(x) \vee P(f(x)),\ Q(y,z),\ \neg P(f(f(x)))\}$$

By convention, $\forall$-quantifiers are not written. An explicitly quantified formula can be restored by first connecting the clauses by $\wedge$ and then $\forall$-quantifying over all variables, or the other way round.

$$\begin{aligned} &\forall x.\ \forall y.\ \forall z.\ (P(a) \wedge (\neg P(x) \vee P(f(x))) \wedge Q(y,z) \wedge \neg P(f(f(x)))) \\ \Leftrightarrow &P(a) \wedge (\forall x.\ (\neg P(x) \vee P(f(x)))) \wedge (\forall y.\ \forall z.\ Q(y,z)) \wedge (\forall x.\ \neg P(f(f(x \end{aligned}$$

Semantic Argument Method applied to Clause Logic

Let $N = \{C_1[\vec{x}], \ldots, C_n[\vec{x}]\}$ be a set of clauses.
Either N is unsatisfiable or else semantic argument gives open branch:

$$
\begin{array}{lll}
I & \not\models & \neg(C_1 \wedge \cdots \wedge C_n) \\
I & \models & C_1 \wedge \cdots \wedge C_n \\
I & \models & C_1 \\
\ldots & & \\
I & \models & C_n \\
\ldots & & \\
I & \models & C_i[\vec{t}] \qquad \text{for all } i = 1..n \text{ and all ground terms } \vec{t} \\
\ldots & &
\end{array}
$$

Conclusion (a bit sloppy): checking satisfiability of N can be done "syntactically", by fixing the domain D_I, interpretation $\alpha_I[f]$ and treating $\forall$-quantification by exhaustive replacement by ground terms.

First-order Clause Logic: Herbrand Semantics

Let F be a formula. An input term (wrt. F) is a term that contains function symbols occurring in F only.

Proposition ("Herband models existence".) Let N be a clause set. If N is satisfiable then there is a model $I \models N$ such that

- $D_I := \{t \mid t \text{ is a input ground term over } \}$
- $\alpha_I[f](t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$.

Proof. Assume N is satisfiable. By soundness, the semantic argument method gives us an (at least one) open branch. The completeness proof allows us to extract from this branch the model I such that

- $D_I := \{t \mid t \text{ is a ground term}\}$
- $\alpha_I[f](t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$
- $\alpha_I[p](t_1, \ldots, t_n) =$ "extracted from open branch"

Because N is a clause set, no inference rule that introdcues a fresh constant is ever applicable. Thus, D_I consists of input (ground) terms only. □

First-order Clause Logic: Herbrand Semantics

Reformulate the previous in commonly used terminology

Herbrand interpretation

- $HU_I := D_I$ from above is the Herbrand universe, however use ground terms only (terms without variables).
- $HB_I = \{p(t_1, \ldots, t_n) \mid t_1, \ldots, t_n \in HU_I\}$ is the Herbrand base.
- Any subset of HB_I is a Herbrand interpretation (misnomer!), exactly those atoms that are true.
- For a clause $C[x]$ and $t \in HU_I$ the clause $C[t]$ is a ground instance.
- For a clause set N the set $\{C[t] \mid C[x] \in N\}$ is its Herbrand expansion.

Example: Herbrand Interpretation

Function symbols: 0, s (for the "+1" function), $+$

Predicate symbols: $<$, $\leq$

$HU_I = \{0, s(0), s(s(0)), \ldots, 0+0, 0+s(0), s(0)+0, \ldots\}$

$\mathbb{N}$ as a Herbrand interpretation, a subset of HB_I:

$$\begin{aligned} I = \{ \quad & 0 \leq 0,\ 0 \leq s(0),\ 0 \leq s(s(0)),\ \ldots, \\ & 0+0 \leq 0,\ 0+0 \leq s(0),\ \ldots, \\ & \ldots,\ (s(0)+0)+s(0) \leq s(0)+(s(0)+s(0)) \\ & \ldots \\ & s(0)+0 < s(0)+0+0+s(0) \\ & \ldots\} \end{aligned}$$

Herbrand Theorem

The soundness and completeness proof of the semantic argument method applied to clause logic provides the following results.

- If a clause set N is unsatisfiable then it has no Herbrand model (trivial).
- If a clause set N is satisfiable then it has a Herbrand model.

  This is the "Herbrand models existence" proposition above.
- Herbrand theorem: if a clause set N is unsatisfiable then some *finite* subset of its Herbrand expansion is unsatisfiable.

  Proof: Suppose N is unsatisfiable. By completeness, there is a proof by semantic argument using the Herbrand expansion of N. Tye proof is a finite tree and hence can use only finitely many elements of the Herbrand expansion.

Herbrand Theorem Illustration

Clause set

$$N = \{P(a),\ \neg P(x) \vee P(f(x)),\ Q(y,z),\ \neg P(f(f(a)))\}$$

Herbrand universe

$$HU_I = \{a,\ f(a),\ f(f(a)), f(f(f(a))), \ldots$$

Herbrand expansion

$$\begin{aligned} N^{\text{gr}} &= \{P(a)\} \\ &\quad \cup \{\neg P(a) \vee P(f(a)),\ \neg P(f(a)) \vee P(f(f(a))), \\ &\qquad\qquad\qquad \neg P(f(f(a))) \vee P(f(f(f(a)))), \ldots\} \\ &\quad \cup \{Q(a,a),\ Q(a,f(a)),\ Q(f(a),a),\ Q(f(a),f(a)), \ldots\} \\ &\quad \cup \{\neg P(f(f(a)))\} \end{aligned}$$

Herbrand Theorem Illustration

$$HB_I = \{\underbrace{P(a)}_{A_0},\ \underbrace{P(f(a))}_{A_1},\ \underbrace{P(f(f(a)))}_{A_2},\ \underbrace{P(f(f(f(a))))}_{A_3}, \ldots\}$$

$$\cup\, \{\underbrace{Q(a,a)}_{B_0},\ \underbrace{Q(a,f(a))}_{B_1},\ \underbrace{Q(f(a),a)}_{B_2},\ \underbrace{Q(f(a),f(a))}_{B_3}, \ldots\}$$

By construction, every atom in N^{gr} occurs in HB_I

Replace in N^{gr} every (ground) atom by its propositional counterpart:

$$\begin{aligned} N^{\text{gr}}_{\text{prop}} = {} & \{A_0\} \\ & \cup \{\neg A_0 \vee A_1,\ \neg A_1 \vee A_2, \neg A_2 \vee A_3, \ldots\} \\ & \cup \{B_0,\ B_1,\ B_2,\ B_3, \ldots\} \\ & \cup \{\neg A_2\} \end{aligned}$$

The subset $\{A_0,\ \neg A_0 \vee A_1,\ \neg A_1 \vee A_2,\ \neg A_2\}$ is unsatisfiable, hence so is N.

Resolution for FOL

Where we are at:

1. Introduced CNF for first-order logic ("Clause logic").
2. Need to define inference rules for first-order logic resolution.
3. "Derivation" has been introduced for propositional logic resolution. No change required.

Alternatives for 2:

- Instantiation (bad!).
- Using unification.

First-Order Resolution through Instantiation

Idea: instantiate clauses appropriately:

$$\begin{tikzcd}
P(z', z') \lor \neg Q(z) \arrow[d, "{[a/z', f(a,b)/z]}"'] & & & \neg P(a, y) \arrow[dl, "{[a/y]}"'] \arrow[dr, "{[b/y]}"] & & & P(x', b) \lor Q(f(x', x)) \arrow[d, "{[a/x', b/x]}"] \\
P(a, a) \lor \neg Q(f(a, b)) \arrow[dr] & & \neg P(a, a) \arrow[dl] & & \neg P(a, b) \arrow[dr] & & P(a, b) \lor Q(f(a, b)) \arrow[dl] \\
& \neg Q(f(a, b)) \arrow[d] & & & & Q(f(a, b)) \arrow[d] & \\
& {} \arrow[drr] & & & & {} \arrow[dll] & \\
& & & \bot & & &
\end{tikzcd}$$

Notation: $[t_1/x_1, \ldots, t_n/x_n]$ is the substitution $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$.

First-Order Resolution through Instantiation

Problems:

- More than one instance of a clause can participate in a proof.
- Even worse: There are infinitely many possible instances.

Observation:

- Instantiation must produce complementary literals (so that inferences become possible).

Idea:

- Do not instantiate more than necessary to get complementary literals.

First-Order Resolution through Instantiation

Idea: do not instantiate more than necessary:

$$\begin{tikzcd}
P(z', z') \vee \neg Q(z) \arrow[d, "{[a/z']}"'] & & & \neg P(a, y) \arrow[dl, "{[a/y]}"'] \arrow[dr, "{[b/y]}"] & & & P(x', b) \vee Q(f(x', x)) \arrow[d, "{[a/x']}"] \\
P(a, a) \vee \neg Q(z) \arrow[dr] & & \neg P(a, a) \arrow[dl] & & \neg P(a, b) \arrow[dr] & & P(a, b) \vee Q(f(a, x)) \arrow[dl] \\
& \neg Q(z) \arrow[d, "{[f(a, x)/z]}"'] & & & & Q(f(a, x)) \arrow[d] & \\
& \neg Q(f(a, x)) \arrow[drr] & & & & Q(f(a, x)) \arrow[dll] & \\
& & & \bot & & &
\end{tikzcd}$$

Lifting Principle

Problem: Make resolution derivations of infinite sets of clauses as they arise from taking the (ground) instances of finitely many first-order clauses (with variables) effective and efficient.

Idea (Robinson 1965):

- Resolution for first-order clauses:
- *Equality* of ground atoms is generalized to *unifiability* of first-order atoms;
- Only compute *most general* (minimal) unifiers.

Lifting Propositional Resolution to First-Order Resolution

Propositional resolution

Clauses	Ground instances
$P(f(x), y)$	$\{P(f(a), a), \ldots, P(f(f(a)), f(f(a))), \ldots\}$
$\neg P(z, z)$	$\{\neg P(a), \ldots, \neg P(f(f(a)), f(f(a))), \ldots\}$

Only common instances of $P(f(x), y)$ and $P(z, z)$ give rise to inference:

$$\frac{P(f(f(a)), f(f(a))) \qquad \neg P(f(f(a)), f(f(a)))}{\perp}$$

Unification

All common instances of $P(f(x), y)$ and $P(z, z)$ are instances of $P(f(x), f(x))$
$P(f(x), f(x))$ is computed deterministically by *unification*

First-order resolution

$$\frac{P(f(x), y) \qquad \neg P(z, z)}{\perp}$$

Justified by existence of $P(f(x), f(x))$

$\boxed{\text{Can represent infinitely many propositional resolution inferences}}$

Unification

A substitution γ is a unifier of terms s and t iff $s\gamma = t\gamma$.

A unifier σ is most general iff for every unifier γ of the same terms there is a substitution δ such that $\gamma = \delta \circ \sigma$ (we write $\sigma\delta$).

Notation: $\sigma = \mathrm{mgu}(s, t)$

Example

$s = car(red, y, z)$
$t = car(u, v, ferrari)$
Then

$$\gamma = \{u \mapsto red,\ y \mapsto fast,\ v \mapsto fast,\ z \mapsto ferrari\}$$

is a unifier, and

$$\sigma = \{u \mapsto red,\ y \mapsto v,\ z \mapsto ferrari\}$$

is a mgu for s and t.
With $\delta = \{v \mapsto fast\}$ obtain $\sigma\delta = \gamma$.

Unification of Many Terms

Let $E = \{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$ be a multiset of equations, where s_i and t_i are terms or atoms. The set E is called a unification problem.

A substitution σ is called a unifier of E if $s_i\sigma = t_i\sigma$ for all $1 \leq i \leq n$.

If a unifier of E exists, then E is called unifiable.

The rule system on the next slide computes a most general unifer of a unification problems or “fail” ($\perp$) if none exists.

Rule Based Naive Standard Unification

Starting with a given unification problem E, apply the following rules as long as possible. The notation "$s \doteq t,\ E$" means "$\{s \doteq t\} \cup E$".

$$t \doteq t, E \Rightarrow E \tag{Trivial}$$

$$f(s_1, \ldots, s_n) \doteq f(t_1, \ldots, t_n), E \Rightarrow s_1 \doteq t_1, \ldots, s_n \doteq t_n, E \tag{Decompose}$$

$$f(\ldots) \doteq g(\ldots), E \Rightarrow \bot \tag{Clash}$$

$$x \doteq t, E \Rightarrow x \doteq t, E\{x \mapsto t\} \tag{Apply}$$

$$\text{if } x \in var(E),\ x \notin var(t)$$

$$x \doteq t, E \Rightarrow \bot \tag{Occur Check}$$

$$\text{if } x \neq t,\ x \in var(t)$$

$$t \doteq x, E \Rightarrow x \doteq t, E \tag{Orient}$$

$$\text{if } t \text{ is not a variable}$$

Example 1

Let $E_1 = \{f(x, g(x), z) \doteq f(x, y, y)\}$ the unification problem to be solved. In each step, the selected equation is underlined.

$$
\begin{array}{lll}
E_1 : & \underline{f(x, g(x), z) \doteq f(x, y, y)} & \text{(given)} \\
E_2 : & \underline{x \doteq x},\ g(x) \doteq y,\ z \doteq y & \text{(by Decompose)} \\
E_3 : & \underline{g(x) \doteq y},\ z \doteq y & \text{(by Trivial)} \\
E_4 : & \underline{y \doteq g(x)},\ z \doteq y & \text{(by Orient)} \\
E_5 : & y \doteq g(x),\ z \doteq g(x) & \text{(by Apply } \{y \mapsto g(x)\}\text{)}
\end{array}
$$

Result is mgu $\sigma = \{y \mapsto g(x),\ z \mapsto g(x)\}$.

Example 2

Let $E_1 = \{f(x, g(x)) \doteq f(x, x)\}$ the unification problem to be solved.
In each step, the selected equation is underlined.

$$
\begin{array}{lll}
E_1: & \underline{f(x, g(x)) \doteq f(x, x)} & \text{(given)} \\
E_2: & \underline{x \doteq x},\ g(x) \doteq x & \text{(by Decompose)} \\
E_3: & \underline{g(x) \doteq x} & \text{(by Trivial)} \\
E_4: & \underline{x \doteq g(x)} & \text{(by Orient)} \\
E_5: & \perp & \text{(by Occur Check)}
\end{array}
$$

There is no unifier of E_1.

Main Properties

The above unification algorithm is sound and complete:
Given $E = \{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$, exhaustive application of the above rules always terminates, and one of the following holds:

- The result is a set equations in solved form, that is, is of the form

  $$x_1 \doteq u_1, \ldots, x_k \doteq u_k$$

  with x_i pairwise distinct variables, and $x_i \notin var(u_j)$.
  In this case, the solved form represents the substitution $\sigma_E = \{x_1 \mapsto u_1, \ldots, x_k \mapsto u_k\}$ and it is a mgu for E.
- The result is $\perp$. In this case no unifier for E exists.

First-Order Resolution Inference Rules

$$\frac{C \vee A \qquad D \vee \neg B}{(C \vee D)\sigma} \quad \text{if } \sigma = \mathrm{mgu}(A, B) \quad \text{[resolution]}$$

$$\frac{C \vee A \vee B}{(C \vee A)\sigma} \quad \text{if } \sigma = \mathrm{mgu}(A, B) \quad \text{[factoring]}$$

For the resolution inference rule, the premise clauses have to be renamed apart (made variable disjoint) so that they don't share variables.

Example

$$\frac{Q(z) \vee P(z, z) \quad \neg P(x, y)}{Q(x)} \quad \text{where } \sigma = [z \mapsto x, y \mapsto x] \quad \text{[resolution]}$$

$$\frac{Q(z) \vee P(z, a) \vee P(a, y)}{Q(a) \vee P(a, a)} \quad \text{where } \sigma = [z \mapsto a, y \mapsto a] \quad \text{[factoring]}$$

Example

(1) $\forall x \,.\, \mathsf{allergies}(x) \rightarrow \mathsf{sneeze}(x)$

(2) $\forall x \,.\, \forall y \,.\, \mathsf{cat}(y) \wedge \mathsf{livesWith}(x, y) \wedge \mathsf{allergicToCats}(x) \rightarrow \mathsf{allergies}(x)$

(3) $\forall x \,.\, \mathsf{cat}(\mathsf{catOf}(x))$

(4) $\mathsf{livesWith}(\mathsf{jerry}, \mathsf{catOf}(\mathsf{jerry}))$

Next

- Resolution applied to the CNF of $(1) \wedge \cdots \wedge (4)$.
- Proof that $(1) \wedge \cdots \wedge (4)$ entails $\mathsf{allergicToCats}(\mathsf{jerry}) \rightarrow \mathsf{sneeze}(\mathsf{jerry})$

Sample Derivation From (1) - (4)

(1) $\neg \mathsf{allergies}(x) \vee \mathsf{sneeze}(x)$ (Given)

(2) $\neg \mathsf{cat}(y) \vee \neg \mathsf{livesWith}(x, y) \vee \neg \mathsf{allergicToCats}(x) \vee \mathsf{allergies}(x)$ (Given)

(3) $\mathsf{cat}(\mathsf{catOf}(x))$ (Given)

(4) $\mathsf{livesWith}(\mathsf{jerry}, \mathsf{catOf}(\mathsf{jerry}))$ (Given)

(5) $\neg \mathsf{livesWith}(x, \mathsf{catOf}(x)) \vee \neg \mathsf{allergicToCats}(x) \vee \mathsf{allergies}(x)$
(Res 2+3, $\sigma = [y \mapsto \mathsf{catOf}(x)]$)

(6) $\neg \mathsf{livesWith}(x, \mathsf{catOf}(x)) \vee \neg \mathsf{allergicToCats}(x) \vee \mathsf{sneeze}(x)$
(Res 1+5, $\sigma = []$)

(7) $\neg \mathsf{allergicToCats}(\mathsf{jerry}) \vee \mathsf{sneeze}(\mathsf{jerry})$ (Res 4+6, $\sigma = [x \mapsto \mathsf{jerry}]$)

Some more (few) clauses are derivable, but not infinitely many.
Not derivable are, e.g.,:
$\mathsf{cat}(\mathsf{catOf}(\mathsf{jerry})), \mathsf{cat}(\mathsf{catOf}(\mathsf{catOf}(\mathsf{jerry}))), \ldots$
But the tableau method would derive then all!

Refutation Example

We want to show

$$(1) \wedge \cdots \wedge (4) \Rightarrow \text{allergicToCats(jerry)} \rightarrow \text{sneeze(jerry)}$$

Equivalently, the CNF of

$$\neg((1) \wedge \cdots \wedge (4) \rightarrow (\text{allergicToCats(jerry)} \rightarrow \text{sneeze(jerry)}))$$

is unsatisfiable. Equivalently

(1) – (4) (Given)

(A) allergicToCats(jerry) (Conclusion)

(B) $\neg$sneeze(jerry) (Conclusion)

is unsatisfiable.

But with the derivable clause

(7) $\neg$allergicToCats(jerry) $\vee$ sneeze(jerry)

the empty clause $\square$ is derivable in two more steps.

Sample Refutation – The Barber Problem

```
set(binary_res). %% This is an "otter" input file
formula_list(sos).
%% Every barber shaves all persons who do not shave themselves:
all x (B(x) -> (all y (-S(y,y) -> S(x,y)))).
%% No barber shaves a person who shaves himself:
all x (B(x) -> (all y (S(y,y) -> -S(x,y)))).
%% Negation of "there are no barbers"
exists x B(x).
end_of_list.
```

otter finds the following refutation (clauses 1 – 3 are the CNF):

```
1 [] -B(x)|S(y,y)|S(x,y).
2 [] -B(x)| -S(y,y)| -S(x,y).
3 [] B($c1).
4 [binary,1.1,3.1] S(x,x)|S($c1,x).
5 [factor,4.1.2] S($c1,$c1).
6 [binary,2.1,3.1] -S(x,x)| -S($c1,x).
10 [factor,6.1.2] -S($c1,$c1).
11 [binary,10.1,5.1] $F.
```

Completeness of First-Order Resolution

Theorem: Resolution is refutationally complete.

- That is, if a clause set is unsatisfiable, then resolution will derive the empty clause $\square$ eventually.
- More precisely: If a clause set is unsatisfiable and closed under the application of the resolution and factoring inference rules, then it contains the empty clause $\square$.
- Proof: Herbrand theorem + completeness of propositional resolution + *Lifting Lemma*

Moreover, in order to implement a resolution-based theorem prover, we need an effective procedure to close a clause set under the application of the resolution and factoring inference rules. See the "given clause loop" below.

Lifting Lemma

Let C and D be variable-disjoint clauses. If

$$\begin{tikzcd} D \arrow[d, "\sigma"] & C \arrow[d, "\rho"] \\ D\sigma & C\rho \end{tikzcd}$$

$$\frac{D\sigma \qquad C\rho}{C'} \qquad \text{[propositional resolution]}$$

then there exists a substitution τ such that

$$\frac{D \qquad C}{C''} \qquad \text{[first-order resolution]}$$

$$\begin{tikzcd} C'' \arrow[d, "\tau"] \\ C' = C''\tau \end{tikzcd}$$

An analogous lifting lemma holds for factoring.

The “Given Clause Loop”

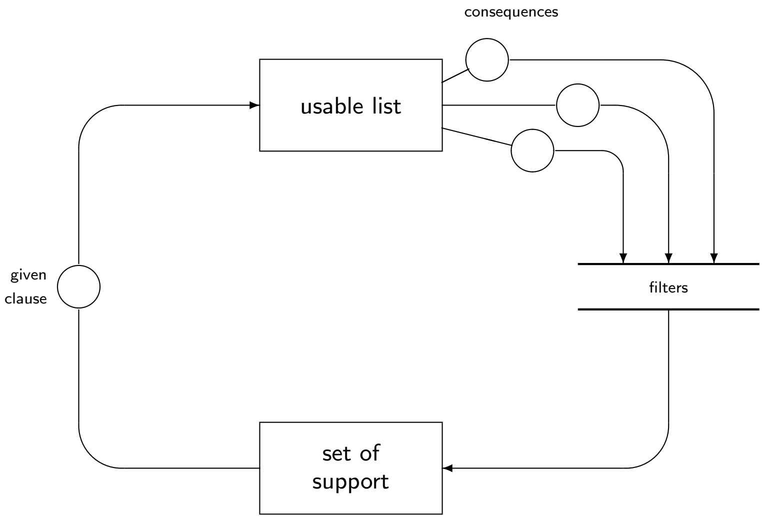
As used in the Otter theorem prover:
Lists of clauses maintained by the algorithm: `usable` and `sos`.
Initialize `sos` with the input clauses, `usable` empty.

Algorithm (straight from the Otter manual):

```
While (sos is not empty and no refutation has been found)
  1. Let given_clause be the ‘lightest’ clause in sos;
  2. Move given_clause from sos to usable;
  3. Infer and process new clauses using the inference rules in
     effect; each new clause must have the given_clause as
     one of its parents and members of usable as its other
     parents;  new clauses that pass the retention tests
     are appended to sos;
End of while loop.
```

Fairness: define clause weight e.g. as “depth + length” of clause.

The “Given Clause Loop” - Graphically



Decidability of FOL

- FOL is undecidable (Turing & Church)
  There does not exist an algorithm for deciding if a FOL formula F is valid, i.e. always halt and says "yes" if F is valid or say "no" if F is invalid.
- FOL is semi-decidable
  There is a procedure that always halts and says "yes" if F is valid, but may not halt if F is invalid.

On the other hand,

- PL is decidable
  There does exist an algorithm for deciding if a PL formula F is valid, e.g. the truth-table procedure.