

# Decision Procedures

Jochen Hoenicke



Software Engineering  
Albert-Ludwigs-University Freiburg

Winter Term 2016/17

- Syntax and Semantics of First Order Logic (FOL)
- Semantic Tableaux for FOL
- FOL is only **semi**-decidable

⇒ Restrictions to decidable fragments of FOL

- Quantifier Free Fragment (QFF)
- QFF of Equality
- Presburger arithmetic
- (QFF of) Linear integer arithmetic
- Real arithmetic
- (QFF of) Linear real/rational arithmetic
- QFF of Recursive Data Structures
- QFF of Arrays
- Putting it all together (Nelson-Oppen).

# First-Order Logic

Also called Predicate Logic or Predicate Calculus

## FOL Syntax

<u>variables</u>	$x, y, z, \dots$
<u>constants</u>	$a, b, c, \dots$
<u>functions</u>	$f, g, h, \dots$ with arity $n > 0$
<u>terms</u>	variables, constants or n-ary function applied to n terms as arguments $a, x, f(a), g(x, b), f(g(x, f(b)))$
<u>predicates</u>	$p, q, r, \dots$ with arity $n \geq 0$
<u>atom</u>	$\top, \perp$ , or an n-ary predicate applied to n terms
<u>literal</u>	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:  $P, Q, R, \dots$

## 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

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  $q(x, f(x))$ ”

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

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

- Fermat's Last Theorem.

$$\forall n. \text{integer}(n) \wedge n > 2$$

$$\rightarrow \forall x, y, z.$$

$$\text{integer}(x) \wedge \text{integer}(y) \wedge \text{integer}(z)$$

$$\wedge x > 0 \wedge y > 0 \wedge z > 0$$

$$\rightarrow x^n + y^n \neq z^n$$

For every regular Language  $L$  there is some  $n \geq 0$ , such that for all words  $z \in L$  with  $|z| \geq n$  there is a decomposition  $z = uvw$  with  $|v| \geq 1$  and  $|uv| \leq n$ , such that for all  $i \geq 0$ :  $uv^i w \in L$ .

$$\begin{aligned} \forall L. \text{regularlanguage}(L) \rightarrow \\ \exists n. \text{integer}(n) \wedge n \geq 0 \wedge \\ \forall z. z \in L \wedge |z| \geq n \rightarrow \\ \exists u, v, w. \text{word}(u) \wedge \text{word}(v) \wedge \text{word}(w) \wedge \\ z = uvw \wedge |v| \geq 1 \wedge |uv| \leq n \wedge \\ \forall i. \text{integer}(i) \wedge i \geq 0 \rightarrow uv^i w \in L \end{aligned}$$

Predicates: *regularlanguage*, *integer*, *word*,  $\cdot \in \cdot$ ,  $\cdot \leq \cdot$ ,  $\cdot \geq \cdot$ ,  $\cdot = \cdot$

Constants: 0, 1

Functions:  $|\cdot|$  (word length), concatenation, iteration



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 (countable infinite), or  
reals (uncountable infinite)
- Assignment  $\alpha_I$ 
  - each variable  $x$  assigned value  $\alpha_I[x] \in D_I$
  - each n-ary function  $f$  assigned

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

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

- each n-ary predicate  $p$  assigned

$$\alpha_I[p] : D_I^n \rightarrow \{\top, \perp\}$$

In particular, each propositional variable  $P$  (0-ary predicate) assigned  
truth value ( $\top, \perp$ )

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

Interpretation  $I : (D_I, \alpha_I)$

$$D_I = \mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\} \quad \text{integers}$$

$$\alpha_I[f] : D_I^2 \rightarrow D_I \quad \alpha_I[g] : D_I^2 \rightarrow D_I$$

$$(x, y) \mapsto x + y \quad (x, y) \mapsto x - y$$

$$\alpha_I[p] : D_I^2 \rightarrow \{\top, \perp\}$$

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

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  $I$

For a variable  $x$ :

## Definition ( $x$ -variant)

An  $x$ -variant 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  $J : I \triangleleft \{x \mapsto v\}$  the  $x$ -variant of  $I$  in which  $\alpha_J[x] = v$  for some  $v \in D_I$ . Then

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

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  $F$ ?

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

Let  $I$  be the standard interpretation for integers,  $D_I = \mathbb{Z}$ .

Compute the value of  $F$  under  $I$ :

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

iff

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

iff

$$\text{for all } v \in D_I, \text{ there exists } v_1 \in D_I, I \triangleleft \{x \mapsto v\} \triangleleft \{y \mapsto v_1\} \models 2 \cdot y = x$$

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

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

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

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

iff

$$\text{for all } v \in D_I, \text{ there exists } v_1 \in D_I, I \triangleleft \{x \mapsto v\} \triangleleft \{y \mapsto v_1\} \models 2 \cdot y = x$$

The latter is true since for  $v \in D_I$  we can choose  $v_1 = \frac{v}{2}$ .

## Definition (Satisfiability)

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

## Definition (Validity)

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

## Note

$F$  is valid iff  $\neg F$  is unsatisfiable

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  $F\sigma$ .



## Definition (Substitution)

A substitution is a mapping from terms to terms, e.g.

$$\sigma : \{t_1 \mapsto s_1, \dots, t_n \mapsto s_n\}$$

By  $F\sigma$  we denote the application of  $\sigma$  to formula  $F$ , i.e., the formula  $F$  where all occurrences of  $t_1, \dots, t_n$  are replaced by  $s_1, \dots, s_n$ .

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

Care has to be taken in the presence of quantifiers:

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

What is  $F[y]$ ?

We need to **rename** bounded variables occurring in the substitution:

$$F[y] : \exists y'. y' = Succ(y)$$

Bounded renaming does not change the models of a formula:

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

$$t\sigma = \begin{cases} \sigma(t) & t \in \text{dom}(\sigma) \\ f(t_1\sigma, \dots, t_n\sigma) & t \notin \text{dom}(\sigma) \wedge t = f(t_1, \dots, t_n) \\ x & t \notin \text{dom}(\sigma) \wedge t = x \end{cases}$$

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

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

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

...

$$(\forall x. F)\sigma = \begin{cases} \forall x. F\sigma & x \notin \text{Vars}(\sigma) \\ \forall x'. ((F\{x \mapsto x'\})\sigma) & \text{otherwise and } x' \text{ is fresh} \end{cases}$$

$$(\exists x. F)\sigma = \begin{cases} \exists x. F\sigma & x \notin \text{Vars}(\sigma) \\ \exists x'. ((F\{x \mapsto x'\})\sigma) & \text{otherwise and } x' \text{ is fresh} \end{cases}$$

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

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

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

$F\sigma$ ?

- 1 Rename

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

$\uparrow$      $\uparrow$

where  $x'$  is a fresh variable

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

Recall rules from propositional logic:

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

$$\frac{I \models F \wedge G}{\begin{array}{l} I \models F \\ I \models G \end{array}} \leftarrow \text{and}$$

$$\frac{I \models F \vee G}{I \models F \mid I \models G}$$

$$\frac{I \models F \rightarrow G}{I \not\models F \mid I \models G}$$

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

$$\begin{array}{l} I \models F \\ I \not\models F \\ \hline I \models \perp \end{array}$$

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

$$\frac{I \not\models F \wedge G}{\begin{array}{l} I \not\models F \\ I \not\models G \end{array}} \leftarrow \text{or}$$

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

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

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

The following additional rules are used for quantifiers:

$$\frac{I \models \forall x.F[x] \text{ for any term } t}{I \models F[t]}$$

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

$$\frac{I \models \exists x.F[x] \text{ for a fresh constant } a}{I \models F[a]}$$

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

(We assume that there are infinitely many constant symbols.)

The formula  $F[t]$  is created from the formula  $F[x]$  by the substitution  $\{x \mapsto t\}$  (roughly, replace every  $x$  by  $t$ ).

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

Assume otherwise.

- |    |   |  |
|----|---|--|
| 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, $\exists (x \mapsto a \text{ fresh})$ |
| 5. | $I \not\models \exists y. p(y, b)$  | 3, $\forall (x \mapsto b \text{ fresh})$ |
| 6. | $I \models p(a, b)$   | 4, $\forall (y \mapsto b)$               |
| 7. | $I \not\models p(a, b)$   | 5, $\exists (y \mapsto a)$               |
| 8. | $I \models \perp$   | 6,7 contradictory                        |

Thus, the formula is valid.

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

Assume  $I$  is a falsifying interpretation for  $F$  and apply semantic argument:

- |    |  |                     |
|----|--|---------------------|
| 1. | $I \not\models (\forall x. p(x, x)) \rightarrow (\exists x. \forall y. p(x, y))$ |                     |
| 2. | $I \models \forall x. p(x, x)$   | 1 and $\rightarrow$ |
| 3. | $I \not\models \exists x. \forall y. p(x, y)$                                    | 1 and $\rightarrow$ |
| 4. | $I \models p(a_1, a_1)$  | 2, $\forall$        |
| 5. | $I \not\models \forall y. p(a_1, y)$   | 3, $\exists$        |
| 6. | $I \not\models p(a_1, a_2)$  | 5, $\forall$        |
| 7. | $I \models p(a_2, a_2)$  | 2, $\forall$        |
| 8. | $I \not\models \forall y. p(a_2, y)$   | 3, $\exists$        |
| 9. | $I \not\models p(a_2, a_3)$  | 8, $\forall$        |
|    | $\vdots$   |                     |

No contradiction. Falsifying interpretation  $I$  can be “read” from proof:

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



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

- **Soundness**

If every branch of a semantic argument proof reach  $I \models \perp$ , then  $F$  is valid

- **Completeness**

Each valid formula  $F$  has a semantic argument proof in which every branch reaches  $I \models \perp$

- **Non-termination**

For an invalid formula  $F$  the method is not guaranteed to terminate. Thus, the semantic argument is **not** a decision procedure for validity.

If for interpretation  $I$  the assumption of the proof holds then there is an interpretation  $I'$  and a branch such that all statements on that branch hold.

$I'$  differs from  $I$  in the values  $\alpha_I[a_i]$  of fresh constants  $a_i$ .

If all branches of the proof end with  $I \models \perp$ , then the assumption was wrong. Thus, if the assumption was  $I \not\models F$ , then  $F$  must be valid.

Consider (finite or infinite) proof trees starting with  $I \not\vdash F$ . We assume that

- all possible proof rules were applied in all non-closed branches.
- the  $\forall$  and  $\exists$  rules were applied for all 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  $F$ .

Otherwise, the proof tree has at least one open branch  $P$ . We show that  $F$  is not valid.

- 1 The statements on that branch  $P$  form a **Hintikka set**:
  - $I \models F \wedge G \in P$  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 terms  $t$ ,  $I \models F[t] \in P$ .
  - $I \not\models \forall x. F[x] \in P$  implies for some term  $a$ ,  $I \not\models F[a] \in P$ .
  - Similarly for  $\exists, \rightarrow, \leftrightarrow, \exists$ .
- 2 Choose  $D_I := \{t \mid t \text{ is term}\}$ ,  $\alpha_I[f](t_1, \dots, t_n) = f(t_1, \dots, t_n)$ ,  
 $\alpha_I[x] = x$  (every term is interpreted as itself)

$$\alpha_I[p](t_1, \dots, t_n) = \begin{cases} \text{true} & I \models p(t_1, \dots, t_n) \in P \\ \text{false} & \text{otherwise} \end{cases}$$

- 3  $I$  satisfies all statements on the branch.  
In particular,  $I$  is a falsifying interpretation of  $F$ , thus  $F$  is not valid.

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

- Devise an algorithm to convert a formula to a normal form.
- Then devise an algorithm for satisfiability/validity that only works on the normal form.

Negations appear only in literals. (only  $\neg, \wedge, \vee, \exists, \forall$ )

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

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

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

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

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

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

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

$$\textcircled{1} \quad \forall x. (\exists y. p(x, y) \wedge p(x, z)) \rightarrow \exists w. p(x, w)$$

$$\textcircled{2} \quad \forall x. \neg(\exists y. p(x, y) \wedge p(x, z)) \vee \exists w. p(x, w)$$

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

$$\textcircled{3} \quad \forall x. (\forall y. \neg(p(x, y) \wedge p(x, z))) \vee \exists w. p(x, w)$$

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

$$\textcircled{4} \quad \forall x. (\forall y. \neg p(x, y) \vee \neg p(x, z)) \vee \exists w. p(x, w)$$

All quantifiers appear at the beginning of the formula

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

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

Every FOL formula  $F$  can be transformed to formula  $F'$  in PNF s.t.  
 $F' \Leftrightarrow F$ :

- 1 Write  $F$  in NNF
- 2 Rename quantified variables to fresh names
- 3 Move all quantifiers to the front



Find equivalent PNF of

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

- Write  $F$  in NNF

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

- Rename quantified variables to fresh names

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

↑  
in the scope of  $\forall x$

- Move all quantifiers to the front

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

Alternately,

$$F'_3 : \forall x. \exists w. \forall y. \neg p(x, y) \vee \neg p(x, z) \vee p(x, w)$$

**Note:** In  $F_2$ ,  $\forall y$  is **in the scope** of  $\forall x$ , therefore the order of quantifiers must be  $\dots \forall x \dots \forall y \dots$

$$F_4 \Leftrightarrow F \text{ and } F'_4 \Leftrightarrow F$$

**Note:** However  $G \not\Leftrightarrow F$

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

- 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 exists an algorithm for deciding if a PL formula  $F$  is valid, e.g., the truth-table procedure.

Similarly for satisfiability