

VNU-HUS MAT1206E/3508: Introduction to AI

First-order Predicate Logic

Hoàng Anh Đức

Bộ môn Tin học, Khoa Toán-Cơ-Tin học
Đại học KHTN, ĐHQG Hà Nội
hoanganhduc@hus.edu.vn



Textbook Basis



First-order Predicate Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

These lecture slides were constructed based primarily on the contents of:

- **Wolfgang Ertel (2025).** *Introduction to Artificial Intelligence*. 3rd. Springer. DOI: 10.1007/978-3-658-43102-0

Contents



First-order Predicate Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and Normal Forms

Proof Calculi

Resolution

Automated Theorem Provers

Mathematical Examples

Applications

Summary

Appendix: Using E

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

Motivation



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Many practical, relevant problems *cannot be* or *can only very inconveniently be formulated* in the language of *propositional logic*

Example 1

- Statement: “Robot 7 is situated at the xy position $(35, 79)$ ”
- Propositional variable:
Robot_7_is_situated_at_ xy _position_ $(35, 79)$ (which is true if the statement holds and false otherwise)
- Assume that 100 robots can stop anywhere on a grid of 100×100 points. We need 10^6 propositional variables to describe all possible positions of the robots [Why?]

3 Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

Motivation



First-order Predicate
Logic

Hoàng Anh Đức

Example 1 (cont.)

- Relationship between objects (robots): “Robot 7 is to the right of robot 12”
- Propositional variable:
Robot_7_is_to_the_right_of_robot_12
- The relation can be represented by an ordered pair of x -coordinates (x_7, x_{12}) where $x_7 > x_{12}$
 - There are $99 + 98 + \dots + 1 = (100 \cdot 99)/2 = 4950$ such ordered pairs
- There are 10^4 propositional variables to describe all possible relations and therefore 10^4 formulas of the type:
Robot_7_is_to_the_right_of_robot_12 $\Leftrightarrow$
Robot_7_is_situated_at_xy_position_(35, 79) $\wedge$
Robot_12_is_situated_at_xy_position_(10, 93) $\vee \dots$

with total $4950 \cdot 10^4$ alternatives on the right-hand side of the formulas [Why?]

4 Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

Motivation



First-order Predicate
Logic

Hoàng Anh Đức

5 Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

- Given 100 robots, describing the relation “Robot A is to the right of robot B ” for all pairs of robots requires a huge number of propositional variables
- In *first-order predicate logic (PL1)*, we can define a predicate $position(number, xPosition, yPosition)$
- Now the relation can be described abstractly with

$$\forall u \forall v is_further_right(u, v) \Leftrightarrow$$

$$\exists x_u \exists y_u \exists x_v \exists y_v position(u, x_u, y_u) \wedge position(v, x_v, y_v) \wedge x_u > x_v$$

where, in general, $\forall z$ is read as “for every z ” and $\exists w$ as “there exists w ”

Syntax



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

6 Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

- Set of *variables* V
- Set of *constants* K (which *stand for objects*)
- Set of *function symbols* F (which *stand for functions*)
- The sets V, K, F are *pairwise disjoint*

Terms

- A *term* is a logical expression that refers to an object
- All *variables* and *constants* are *(atomic) terms*
- If $t_1, \dots, t_n$ are terms and f is an n -place function symbol, then $f(t_1, \dots, t_n)$ is a *(complex) term*

Example 2

- $f(\sin(\ln(3)), \exp(x))$ is a term
 - $V = \{x\}$, $K = \{3\}$, and $F = \{\sin, \ln, \exp, f\}$
- $g(g(g(x)))$ is a term
 - $V = \{x\}$, $K = \emptyset$, and $F = \{g\}$
- $LeftLeg(John)$ is a term
 - $V = \emptyset$, $K = \{John\}$, and $F = \{LeftLeg\}$
- It is important to remember that *a complex term is just a complicated kind of name*. It is not a “subroutine call” that “returns a value.”
 - Constant symbols are names for objects
 - It is not always convenient to have a distinct symbol for each object. Therefore, we also use function symbols to indicate names for objects
 - For example, instead of giving a constant symbol for “John’s left leg”, we use $LeftLeg(John)$

Syntax



To be able to establish *logical relationships between terms*, we build formulas from terms

Predicate Logic Formulas

Let P be a set of *predicate symbols* (which *stand for relations*)

- If $t_1, \dots, t_n$ are terms and p is an n -place predicate symbol, then $p(t_1, \dots, t_n)$ is an *(atomic) formula*
- If A and B are formulas, then $\neg A$, (A) , $A \wedge B$, $A \vee B$, $A \Rightarrow B$, $A \Leftrightarrow B$ are also *formulas*
- If x is a variable and A a formula, then $\forall x A$ and $\exists x A$ are also *formulas*. $\forall$ is the *universal quantifier* and $\exists$ the *existential quantifier*
- $p(t_1, \dots, t_n)$ and $\neg p(t_1, \dots, t_n)$ are called *literals*
- Formulas in which every variable is in the scope of a quantifier are called *first-order sentences* or *closed formulas*. Variables which are not in the scope of a quantifier are called *free variables*
- Definitions of *CNF* and *Horn clauses* hold for formulas of predicate logic analogously

First-order Predicate Logic

Hoàng Anh Đức

Motivation

8 Syntax

Semantics

Quantifiers and Normal Forms

Proof Calculi

Resolution

Automated Theorem Provers

Mathematical Examples

Applications

Summary

References

Appendix: Using E

Example 3

Formula	Description
$\forall x \text{ frog}(x) \Rightarrow \text{green}(x)$	All frogs are green
$\forall x \text{ frog}(x) \wedge \text{brown}(x) \Rightarrow \text{big}(x)$	All brown frogs are big
$\forall x \text{ likes}(x, \text{cake})$	Everyone likes cake
$\neg \forall x \text{ likes}(x, \text{cake})$	Not everyone likes cake
$\neg \exists x \text{ likes}(x, \text{cake})$	No one likes cake
$\exists x \forall y \text{ likes}(y, x)$	There is something that everyone likes
$\exists x \forall y \text{ likes}(x, y)$	There is someone who likes everything
$\forall x \exists y \text{ likes}(y, x)$	Everything is loved by someone
$\forall x \exists y \text{ likes}(x, y)$	Everyone likes something

Example 3 (cont.)

Formula	Description
$\forall x \text{ customer}(x) \Rightarrow \text{likes}(\text{bob}, x)$	Bob likes every customer
$\exists x \text{ customer}(x) \wedge \text{likes}(\text{bob}, x)$	There is a customer whom Bob likes
$\exists x \text{ baker}(x) \wedge \forall y \text{ customer}(y) \Rightarrow \text{likes}(x, y)$	There is a baker who likes all of his customers
$\forall x \text{ older}(\text{mother}(x), x)$	Every mother is older than her child
$\forall x \text{ older}(\text{mother}(\text{mother}(x)), x)$	Every grandmother is older than her daughter's child
$\forall x \forall y \forall z \text{ rel}(x, y) \wedge \text{rel}(y, z) \Rightarrow \text{rel}(x, z)$	<i>rel</i> is a transitive relation

Semantics



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

11 Semantics

Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

In *propositional logic*, an *interpretation* (*assignment*) is a mapping that assigns a truth value (either t or f) to each variable. Analogously, in *first-order predicate logic*, we have

Interpretation

An *interpretation* (or *assignment*) $\mathbb{I}$ is defined as

- A mapping from the set of *constants and variables* $K \cup V$ to a set W of names of *objects* in the world
- A mapping from the set of *function symbols* to the set of *functions* in the world. Every n -place function symbol is assigned an n -place function
- A mapping from the set of *predicate symbols* to the set of *relations* in the world. Every n -place predicate symbol is assigned an n -place relation.

- The *truth of a formula* in PL1 depends on the *interpretation*

Example 4

- Constants $K = \{c_1, c_2, c_3\}$
- A two-place function symbols *plus*
- A two-place predicate symbol *gr*

Consider the formula $\varphi \equiv \text{gr}(\text{plus}(c_1, c_3), c_2)$

Interpretation $\mathbb{I}_1$

- Mapping constants: $c_1 \mapsto 1, c_2 \mapsto 2, \text{ and } c_3 \mapsto 3$
 - Mapping function symbol: *plus* $\mapsto +$
 - Mapping predicate symbol: *gr* $\mapsto >$
- $\varphi \mapsto 1 + 3 > 2$ or equivalently $\varphi \mapsto 4 > 2$
 - The greater-than relation $>$ on the set $\{1, 2, 3, 4\}$ is the set of pairs (x, y) of numbers in $\{1, 2, 3, 4\}$ with $x > y$, and $(4, 2)$ is *in* that set
 - φ is *true* under the interpretation $\mathbb{I}_1$

The truth of a formula

- An *atomic formula* $p(t_1, \dots, t_n)$ is *true under the interpretation* $\mathbb{I}$ if, after interpretation and evaluation of all terms $t_1, \dots, t_n$ and interpretation of the predicate p through the n -place relation r , it holds that $(\mathbb{I}(t_1), \dots, \mathbb{I}(t_n)) \in r$
- *The truth of quantifierless formulas* follows from the *truth of atomic formulas through the semantics of the logical operators* $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$, like in propositional logic
- A *formula* $\forall x F$ is *true under the interpretation* $\mathbb{I}$ exactly when it is *true given an arbitrary change of the interpretation for the variable* x (and only for x)
- A *formula* $\exists x F$ is *true under the interpretation* $\mathbb{I}$ exactly when *there is an interpretation for* x *which makes the formula true*

The definitions of *semantic equivalence of formulas*, for the concepts *satisfiable*, *true*, *unsatisfiable*, and *model*, along with *semantic entailment* carry over *unchanged from propositional calculus to predicate logic*

Semantics



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

14 Semantics

Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

The semantics of logical operators holds analogously for PL1

A	B	(A)	$\neg A$	$A \wedge B$	$A \vee B$	$A \Rightarrow B$	$A \Leftrightarrow B$
t	t	t	f	t	t	t	t
t	f	t	f	f	t	f	f
f	t	f	t	f	t	t	f
f	f	f	t	f	f	t	t

The following theorems hold analogously for PL1

Theorem 1 (Deduction theorem)

$A \models B$ if and only if $\models A \Rightarrow B$

Theorem 2 (Proof by contradiction)

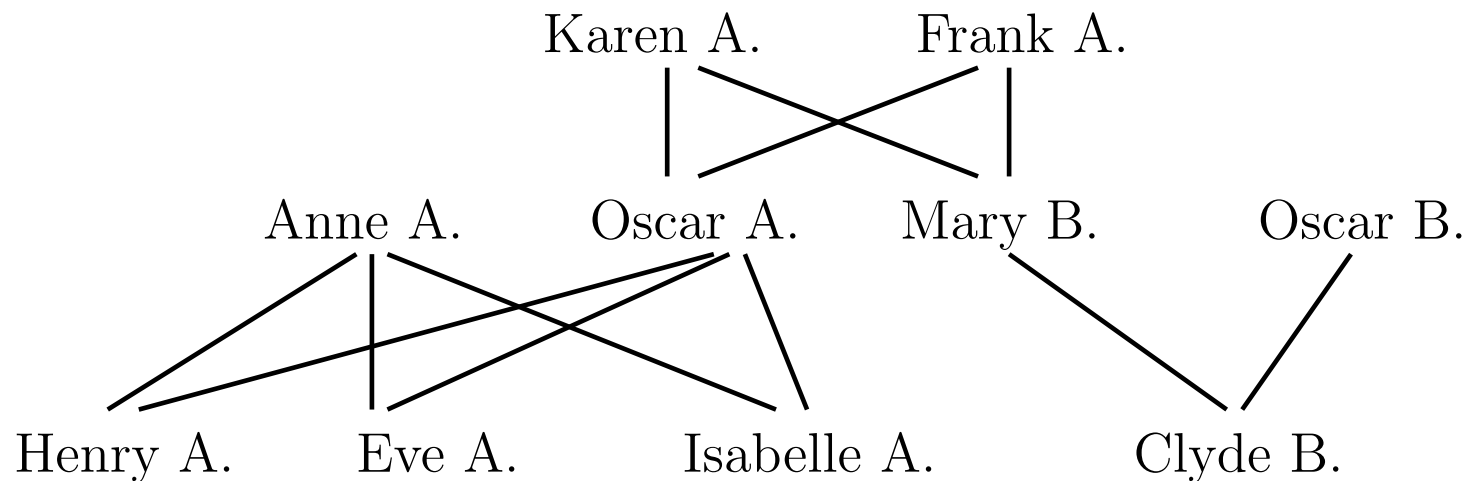
$KB \models Q$ if and only if $KB \wedge \neg Q$ is unsatisfiable

Semantics



Example 5

A family tree



- **Child** – A three-place relation (predicate)

$Child = \{(Oscar\ A., Karen\ A., Frank\ A.), (Mary\ B., Karen\ A., Frank\ A.), (Henry\ A., Anne\ A., Oscar\ A.), (Eve\ A., Anne\ A., Oscar\ A.), (Isabelle\ A., Anne\ A., Oscar\ A.), (Clyde\ B., Mary\ B., Oscar\ B.)\}$

- The triple $(Oscar\ A., Karen\ A., Frank\ A.)$ stands for “Oscar A. is a child of Karen A. and Frank A.”

First-order Predicate Logic

Hoàng Anh Đức

Motivation

Syntax

15 Semantics

Quantifiers and Normal Forms

Proof Calculi

Resolution

Automated Theorem Provers

Mathematical Examples

Applications

Summary

References

Appendix: Using E

Example 5 (cont.)

- From the names, we read off the one-place relation (predicate) *Female* of the women
 $Female = \{Karen A., Anne A., Mary B., Eve A., Isabelle A.\}$
- We now want to *establish formulas for family relationships*
 - We define a *three-place predicate* $child(x, y, z)$ with the semantics $\mathbb{I}(child(x, y, z)) = w \equiv (\mathbb{I}(x), \mathbb{I}(y), \mathbb{I}(z)) \in Child$
 - If $\mathbb{I}$ is the interpretation $oscar \mapsto Oscar A., eve \mapsto Eve A.,$ and $anne \mapsto Anne A.$, then $\mathbb{I}(child(eve, anne, oscar)) \equiv (Eve A., Anne A., Oscar A.) \in Child$. Thus, $child(eve, anne, oscar)$ is *true under the interpretation $\mathbb{I}$*
 - Naturally, based on what we knew from the real world, we would *want* $child(eve, oscar, anne)$ *also to be true*. To have this, we require $\forall x \forall y \forall z child(x, y, z) \Leftrightarrow child(x, z, y)$
 - We can also define a *two-place predicate* $descendant(x, y)$ recursively as

$$\forall x \forall y descendant(x, y) \Leftrightarrow \exists z child(x, y, z)$$

$$\vee (\exists u \exists v child(x, u, v) \wedge descendant(u, y))$$

Semantics



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

17 Semantics

Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

The predicates we defined in Example 5 are based on our knowledge from the real world. Analogously to the propositional logic, we can *build a small knowledge base with rules and facts*, for example, like

$$\begin{aligned} KB \equiv & \text{female}(\text{karen}) \wedge \text{female}(\text{anne}) \wedge \text{female}(\text{mary}) \wedge \\ & \text{female}(\text{eve}) \wedge \text{female}(\text{isabelle}) \wedge \text{child}(\text{oscar}, \text{karen}, \text{frank}) \wedge \\ & \text{child}(\text{mary}, \text{karen}, \text{frank}) \wedge \text{child}(\text{eve}, \text{anne}, \text{oscar}) \wedge \\ & \text{child}(\text{henry}, \text{anne}, \text{oscar}) \wedge \text{child}(\text{isabelle}, \text{anne}, \text{oscar}) \wedge \\ & \text{child}(\text{clyde}, \text{mary}, \text{oscar}) \wedge (\forall x \forall y \forall z \text{child}(x, y, z) \Rightarrow \text{child}(x, z, y)) \\ & \wedge (\forall x \forall y \text{descendant}(x, y) \Leftrightarrow \exists z \text{child}(x, y, z) \\ & \vee (\exists u \exists v \text{child}(x, u, v) \wedge \text{descendant}(u, y))) \end{aligned}$$

and ask, for example, *whether child(eve, oscar, anne) or descendant(eve, frank) are (syntactically) derivable?* To that end, we require a *calculus*

Exercise 1 ([Ertel 2025], Exercise 3.1, p. 65)

Let the three-place predicate “*child*” and the one-place predicate “*female*” from Example 5 be given. Define:

- (a) A one-place predicate “*male*”
- (b) A two-place predicate “*father*” and “*mother*”
- (c) A two-place predicate “*siblings*”
- (d) A predicate “*parents*(x, y, z)”, which is true if and only if x is the father and y is the mother of z
- (e) A predicate “*uncle*(x, y)”, which is true if and only if x is the uncle of y (use the predicates that have already been defined)
- (f) A two-place predicate “*ancestor*” with the meaning:
ancestors are parents, grandparents, etc. of arbitrarily many generations



Exercise 2 ([Ertel 2025], Exercise 3.2, p. 65)

Formalize the following statements in predicate logic:

- (a) Every person has a father and a mother
- (b) Some people have children
- (c) All birds fly
- (d) There is an animal that eats (some) grain-eating animals
- (e) Every animal eats plants or plant-eating animals which are much smaller than itself

Semantics



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

20 Semantics

Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

- To be able to compare terms, *equality* is a very important relation in predicate logic
- The *equality of terms in mathematics* is an *equivalence relation*, meaning it is *reflexive*, *symmetric* and *transitive*
- We define a *predicate* “=” using infix notation as is customary in mathematics (that is, instead of writing “ $eq(x, y)$ ”, we write “ $x = y$ ”)

Equality Axioms

$$\forall x \quad x = x \quad (\text{reflexivity})$$

$$\forall x \forall y \quad x = y \Rightarrow y = x \quad (\text{symmetry})$$

$$\forall x \forall y \forall z \quad x = y \wedge y = z \Rightarrow x = z \quad (\text{transitivity})$$

To guarantee the uniqueness of functions, we additionally require that for any function symbol f and any predicate symbol p

$$\forall x \forall y \quad x = y \Rightarrow f(x) = f(y) \quad (\text{substitution axiom})$$

$$\forall x \forall y \quad x = y \Rightarrow p(x) \Leftrightarrow p(y) \quad (\text{substitution axiom})$$



Exercise 3 ([Ertel 2025], Exercise 3.3, p. 65)

Adapt Exercise 1 by using one-place function symbols and equality instead of “*father*” and “*mother*”

Exercise 4 ([Ertel 2025], Exercise 3.4, p. 66)

Give predicate logic axioms for the two-place relation “ $<$ ” as a total order. For a total order we must have (1) Any two elements are comparable. (2) It is asymmetric. (3) It is transitive

Semantics



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

22 Semantics

Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

Often a variable must be replaced by a term

Example 6

Consider the formula $\forall x x = 5 \Rightarrow x = y$

Replace y by the term $\sin(x)$?

$$\forall x x = 5 \Rightarrow x = \sin(x) \quad \text{WRONG!}$$

The “ x ” in $\sin(x)$ is *captured* by $\forall x$. Since y is independent of x , two correct ways avoid this:

- (1) rename the *bound* variable: $\forall x' x' = 5 \Rightarrow x' = \sin(x)$
- (2) rename the variable in the *term*: $\forall x x = 5 \Rightarrow x = \sin(z)$

Both **CORRECT** — the same formula up to the free variable’s name (x vs z).

Replacing a variable by a term

We write $\varphi[x/t]$ for the formula that results when we *replace every free occurrence of the variable x in φ with the term t* . Thereby we *do not allow any variables in the term t that are quantified in φ* . In those cases *variables must be renamed* (the bound variable, or the clashing variable in the term)

Substitution

A *substitution* σ is *a map from variables to terms*.

- ϵ : the empty substitution
- $\sigma : x_1/t_1, x_2/t_2, \dots, x_n/t_n$: a substitution that maps each variable x_i to the corresponding term t_i
- Also write $\sigma = \{x_1/t_1, x_2/t_2, \dots, x_n/t_n\}$
- *Apply σ to a term t* , denoted by $\sigma(t)$ or $t[x_1/t_1, \dots, x_n/t_n]$ to indicate $\sigma(t)$, means *simultaneously replacing every occurrence of each x_i in t by t_i*



Quantifiers and Normal Forms

First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

24 Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

- By definition, $\forall x p(x)$ is true if and only if $p(x)$ is true for all interpretations of the variable x . Instead, we can write

$$\forall x p(x) \equiv p(a_1) \wedge \cdots \wedge p(a_n),$$

where $a_1, \dots, a_n$ are all constants in K

- Similarly, for $\exists x p(x)$

$$\exists x p(x) \equiv p(a_1) \vee \cdots \vee p(a_n),$$

where $a_1, \dots, a_n$ are all constants in K

- From this, it follows with the De Morgan's law that

$$\forall x \varphi \equiv \neg \exists x \neg \varphi$$

Through this equivalence, *universal and existential quantifiers are mutually replaceable*

Example 7

“Everyone wants to be loved” $\equiv$ “Nobody does not want to be loved”



Quantifiers and Normal Forms

Prenex Normal Form

A predicate logic formula φ is in *prenex normal form* if it holds that

- $\varphi = Q_1x_1 \dots Q_nx_n\psi$
- ψ is a quantifierless formula
- $Q_i \in \{\forall, \exists\}$ for $i = 1, \dots, n$

Theorem 3

Every predicate logic formula can be transformed into an equivalent formula in prenex normal form

Example 8

- Be careful in case *a quantified variable appears outside the scope of its quantifier*
 - For example, consider $\forall x (p(x) \Rightarrow \exists x q(x))$
 - In this case, one of the two variables “ x ” (in $\forall x$ and $\exists x$) must be renamed, for example, like $\forall x (p(x) \Rightarrow \exists y q(y))$

First-order Predicate Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

25 Quantifiers and Normal Forms

Proof Calculi

Resolution

Automated Theorem Provers

Mathematical Examples

Applications

Summary

References

Appendix: Using E

Quantifiers and Normal Forms



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

26 Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

Transformation into prenex normal form

- Transformation into conjunctive normal form
 - Elimination of equivalences
 - Elimination of implications
 - Repeated application of De Morgan's law and distributive law
 - Renaming of variables if necessary
 - Factoring out universal quantifiers
-
- The formula $\forall x (p(x) \Rightarrow \exists y q(y))$ can be written in prenex normal form by bringing the quantifier $\exists$ to the front, as in $\forall x \exists y (p(x) \Rightarrow q(y))$

Exercise 5

How about $(\forall x p(x)) \Rightarrow (\exists y q(y))$? Can you bring this formula to prenex normal form? (**Hint:** Notice the scope of $\forall x$.)

Quantifiers and Normal Forms

Example 9 (Transforming into prenex normal form)

- The convergence of a series $(a_n)_{n \in \mathbb{N}}$ to a limit a is defined by $\forall \epsilon > 0 \exists n_0 \in \mathbb{N} \forall n > n_0 |a_n - a| < \epsilon$
- We define the **functions** $abs(x)$ for x , $a(n)$ for a_n , $minus(x, y)$ for $x - y$, and the **predicates** $el(x, y)$ for $x \in y$, $gr(x, y)$ for $x > y$. The formula becomes

$$\begin{aligned} & \forall \epsilon (gr(\epsilon, 0) \Rightarrow \exists n_0 (el(n_0, \mathbb{N}) \Rightarrow \forall n (gr(n, n_0) \\ & \Rightarrow gr(\epsilon, abs(minus(a(n), a)))))) \end{aligned}$$

- To bring the formula to **prenex normal form**:
 - No variables need to be renamed
 - Eliminating the implications

$$\begin{aligned} & \forall \epsilon (\neg gr(\epsilon, 0) \vee \exists n_0 (\neg el(n_0, \mathbb{N}) \vee \forall n (\neg gr(n, n_0) \\ & \vee gr(\epsilon, abs(minus(a(n), a)))))) \end{aligned}$$

- Move quantifiers to the front

$$\begin{aligned} & \forall \epsilon \exists n_0 \forall n (\neg gr(\epsilon, 0) \vee \neg el(n_0, \mathbb{N}) \vee \neg gr(n, n_0) \\ & \vee gr(\epsilon, abs(minus(a(n), a)))) \end{aligned}$$

Quantifiers and Normal Forms



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

28 Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

Skolemization

■ Goal:

- Eliminate all existential quantifiers from a formula in prenex normal form

■ General Method:

- Replacement of existentially quantified variables by new Skolem functions
- Deletion of resulting universal quantifiers

Note

- After Skolemization, *the resulting formula is no longer equivalent to the original one*
- However, its *satisfiability remains unchanged*: the original is satisfiable *iff* the Skolemized formula is
 - This is particularly useful when we want to show that a formula, say $KB \wedge \neg Q$, is *unsatisfiable*

Quantifiers and Normal Forms



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

29 Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

Example 10 (Skolemization)

- We *skolemize* the following formula

$$\forall x_1 \forall x_2 \exists y_1 \forall x_3 \exists y_2 p(f(x_1), x_2, y_1) \vee q(y_1, x_3, y_2)$$

- The variable y_1 apparently *depends on x_1 and x_2* . We *replace every occurrence of y_1 by a Skolem function $g(x_1, x_2)$* , where g is a new function symbol that has not yet appeared in the formula

$$\forall x_1 \forall x_2 \forall x_3 \exists y_2 p(f(x_1), x_2, g(x_1, x_2)) \vee q(g(x_1, x_2), x_3, y_2)$$

- Analogously, the variable y_2 apparently *depends on x_1, x_2 , and x_3* . We *replace every occurrence of y_2 by a Skolem function $h(x_1, x_2, x_3)$*

$$\forall x_1 \forall x_2 \forall x_3 p(f(x_1), x_2, g(x_1, x_2)) \vee q(g(x_1, x_2), x_3, h(x_1, x_2, x_3))$$

- Finally, as all variables are universally quantified, the universal quantifiers can be left out, giving us the resulting skolemized formula

$$p(f(x_1), x_2, g(x_1, x_2)) \vee q(g(x_1, x_2), x_3, h(x_1, x_2, x_3))$$

Quantifiers and Normal Forms



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

30 Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

Example 11 (Skolemization)

- We *skolemize* the following formula

$$\forall \epsilon \exists n_0 \forall n (\neg gr(\epsilon, 0) \vee \neg el(n_0, \mathbb{N}) \vee \neg gr(n, n_0) \\ \vee gr(\epsilon, abs(minus(a(n), a))))$$

- The variable n_0 apparently *depends on* ϵ . We *replace every occurrence of n_0 by a Skolem function $n_0(\epsilon)$* .

$$\forall \epsilon \forall n (\neg gr(\epsilon, 0) \vee \neg el(n_0(\epsilon), \mathbb{N}) \vee \neg gr(n, n_0(\epsilon)) \\ \vee gr(\epsilon, abs(minus(a(n), a))))$$

- Finally, we have the skolemized formula

$$(\neg gr(\epsilon, 0) \vee \neg el(n_0(\epsilon), \mathbb{N}) \vee \neg gr(n, n_0(\epsilon)) \\ \vee gr(\epsilon, abs(minus(a(n), a))))$$



Quantifiers and Normal Forms

When skolemizing a formula in prenex normal form

- *All existential quantifiers are eliminated from the outside inward.* That is, a formula of the form $\forall x_1 \forall x_2 \dots \forall x_n \exists y \varphi$ is replaced by $\forall x_1 \forall x_2 \dots \forall x_n \varphi[y/f(x_1, x_2, \dots, x_n)]$ where f is a new function that has not yet appeared in φ
- *If an existential quantifier is on the far outside, such as in $\exists y p(y)$, then y must be replaced by a constant*

Running time (in the number of literals)

- Transformation into prenex normal form
 - Exponential (naive)
 - Polynomial [Eder 1992]
- Skolemization
 - Polynomial

First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

31 Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

Proof Calculi



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

32 Proof Calculi

Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

- For *reasoning in predicate logic*, various calculi of *natural reasoning* have been developed
- For example, *Gentzen calculus* or *sequent calculus*
 - *These calculi are meant to be applied by humans*: the inference rules are more or less intuitive, the calculi work on arbitrary PL1 formulas
- Based on Example 5, we give an example of a *small “natural” proof* using the following *inference rules*
 - *Modus Ponens (MP)*
 - *$\forall$ -Elimination ($\forall$ -E)*

$$\frac{A, A \Rightarrow B}{B}$$

$$\frac{\forall x A}{A[x/t]}$$

Note: t is a ground term that
contains no variables

Proof Calculi



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

33 Proof Calculi

Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

Example 12 (A natural deduction)

- In Example 5, we built a small knowledge base KB with rules and facts

$$\begin{aligned} KB \equiv & \text{female}(\text{karen}) \wedge \text{female}(\text{anne}) \wedge \text{female}(\text{mary}) \wedge \\ & \text{female}(\text{eve}) \wedge \text{female}(\text{isabelle}) \wedge \text{child}(\text{oscar}, \text{karen}, \text{frank}) \wedge \\ & \text{child}(\text{mary}, \text{karen}, \text{frank}) \wedge \text{child}(\text{eve}, \text{anne}, \text{oscar}) \wedge \\ & \text{child}(\text{henry}, \text{anne}, \text{oscar}) \wedge \text{child}(\text{isabelle}, \text{anne}, \text{oscar}) \wedge \\ & \text{child}(\text{clyde}, \text{mary}, \text{oscar}) \wedge (\forall x \forall y \forall z \text{child}(x, y, z) \Rightarrow \text{child}(x, z, y)) \\ & \wedge (\forall x \forall y \text{descendant}(x, y) \Leftrightarrow \exists z \text{child}(x, y, z) \\ & \vee (\exists u \exists v \text{child}(x, u, v) \wedge \text{descendant}(u, y))) \end{aligned}$$

- Can $\text{child}(\text{eve}, \text{oscar}, \text{anne})$ be derived using the MP and $\forall$ -E inference rules?

Step	Proved by
1. $\text{child}(\text{eve}, \text{anne}, \text{oscar})$	KB
2. $\forall x \forall y \forall z \text{child}(x, y, z) \Rightarrow \text{child}(x, z, y)$	KB
3. $\text{child}(\text{eve}, \text{anne}, \text{oscar}) \Rightarrow \text{child}(\text{eve}, \text{oscar}, \text{anne})$	$\forall$ -E for 2: $x/\text{eve},$ $y/\text{anne}, z/\text{oscar}$
4. $\text{child}(\text{eve}, \text{oscar}, \text{anne})$	MP for 1 and 3

Proof Calculi



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

34 Proof Calculi

Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

- The calculus with *just MP and $\forall$ -E* is *not complete*
- It *can be extended to a complete calculus* by adding further inference rules

Theorem 4 (Gödel's completeness theorem [Gödel 1930])

First-order predicate logic is complete. That is, there is a calculus with which every proposition that is a consequence of a knowledge base KB can be proved. If $KB \models \varphi$, then it holds that $KB \vdash \varphi$

- Every true proposition in first-order predicate logic is provable (= syntactically derivable)
- Is the reverse true? In other words, is everything we can derive syntactically actually true?

Proof Calculi



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

35 Proof Calculi

Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

Indeed, the answer is “yes”

Theorem 5 (Correctness)

There are calculi with which only true propositions can be proved. That is, if $KB \vdash \varphi$, then $KB \models \varphi$

- *Provability and semantic consequence are therefore equivalent concepts, as long as correct and complete calculus is being used.* Thereby first-order predicate logic becomes a powerful tool for mathematics and AI
- The aforementioned calculi of natural deduction are rather *unsuited for automatization*
- Only *resolution calculus*, which was introduced in 1965 [Robinson 1965] and essentially works with only one simple inference rule, *enabled the construction of powerful automated theorem provers*, which later were employed as inference machines for expert systems

Resolution



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

36 Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

Recall: Resolution Rule

$$\frac{A \vee B, \neg B \vee C}{A \vee C} \quad \text{or} \quad \frac{A \vee B, B \Rightarrow C}{A \vee C}$$

Recall: Resolution Proof

- **Input:** Knowledge base KB
- **Goal:** Decide whether $KB \models Q$
- **Method:** Add $\neg Q$ to the knowledge base. If the empty clause can be derived, conclude $KB \models Q$. If there is no more resolvable pair of clauses (and the empty clause is not derived), conclude $KB \not\models Q$
- **In PL1:** resolution only *semi-decides* entailment — if $KB \not\models Q$, the search may not terminate, so the second case need not occur (see Lecture 4).

Resolution



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

37 Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

Example 13 (Resolution Proof of Example 12)

- $KB \equiv child(eve, anne, oscar) \wedge (\forall x \forall y \forall z child(x, y, z) \Rightarrow child(x, z, y))$
- $Q \equiv child(eve, oscar, anne)$
- Proof:

Step	Proved by
1. $child(eve, anne, oscar)$	KB
2. $\forall x \forall y \forall z child(x, y, z) \Rightarrow child(x, z, y)$	KB
3. $\neg child(eve, oscar, anne)$	$\neg Q$
4. $child(eve, anne, oscar) \Rightarrow child(eve, oscar, anne)$	Unification for 2: $x/eve, y/anne, z/oscar$
5. $\neg child(eve, anne, oscar) \vee child(eve, oscar, anne)$	Equivalent form of 4
6. $\neg child(eve, anne, oscar)$	$Res(3, 5)$
7. $()$	$Res(1, 6)$

Resolution



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

38 Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

Example 14

- Everyone knows his own mother. Does Henry know anyone?
- We use a function symbol *mother* and a predicate symbol *knows*
- $KB \equiv \forall x \text{ knows}(x, \text{mother}(x))$
- $Q \equiv \exists y \text{ knows}(\text{henry}, y)$
- Proof:

Step	Proved by
1. $\forall x \text{ knows}(x, \text{mother}(x))$	KB
2. $\forall y \neg \text{knows}(\text{henry}, y)$	$\neg Q$
3. $\text{knows}(\text{henry}, \text{mother}(\text{henry}))$	Uni. (σ) for 1: x/henry
4. $\neg \text{knows}(\text{henry}, \text{mother}(\text{henry}))$	Uni. (σ) for 2: $y/\text{mother}(\text{henry})$
5. $()$	Res(3, 4)

- The replacement step σ defined by x/henry and $y/\text{mother}(\text{henry})$ is called *unification*

Resolution



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

39 Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

Unifier

- Two literals are called *unifiable* if *there is a substitution σ for all variables which makes the literals identical*. Such a σ is called a *unifier*.
- A unifier is called the *most general unifier (MGU)* if *all other unifiers can be obtained from it by substitution of variables*.

Resolution



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

40 Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

Example 15

- Two literals $knows(x, mother(x))$ and $knows(henry, y)$ in Example 14 are unifiable
With the substitution $\sigma : x/henry, y/mother(henry)$ both literals become $knows(henry, mother(henry))$.
- Two literals $knows(henry, x)$ and $knows(x, marry)$ are not unifiable.
 - The variable x cannot take on the values $henry$ and $marry$ at the same time.
 - However, $knows(henry, x)$ means that “Henry knows everyone”. So we should be able to infer that Henry knows Marry
 - The problem arises because both literals use the same variable x , which can be avoided by renaming x in $knows(x, marry)$ to y (a new variable name) without changing its meaning (bound-variable renaming, as in Example 6 — *standardizing apart*). Now, $knows(henry, x)$ and $knows(y, marry)$ are unifiable by the substitution $\sigma : x/marry, y/henry$

Example 16

- There are *different unifiers* for unifying the literals $p(f(g(x)), y, z)$ and $p(u, u, f(u))$

$\sigma_1 :$	$y/f(g(x)),$	$z/f(f(g(x))),$	$u/f(g(x))$	
$\sigma_2 :$	$x/h(v),$	$y/f(g(h(v))),$	$z/f(f(g(h(v)))),$	$u/f(g(h(v)))$
$\sigma_3 :$	$x/h(h(v)),$	$y/f(g(h(h(v))))),$	$z/f(f(g(h(h(v))))),$	$u/f(g(h(h(v))))$
$\sigma_4 :$	$x/h(a),$	$y/f(g(h(a))),$	$z/f(f(g(h(a)))),$	$u/f(g(h(a)))$
$\sigma_5 :$	$x/a,$	$y/f(g(a)),$	$z/f(f(g(a))),$	$u/f(g(a))$

- Among these unifiers, σ_1 *is the most general unifier (MGU)*: The other unifiers can be respectively obtained from σ_1 through the substitutions $x/h(v)$, $x/h(h(v))$, $x/h(a)$, and x/a

Resolution



First-order Predicate Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and Normal Forms

Proof Calculi

42 Resolution

Automated Theorem Provers

Mathematical Examples

Applications

Summary

References

Appendix: Using E

- Need a procedure to find a MGU given a set of expressions
- Requirements:
 - stop after a finite number of steps
 - return an MGU if the set is unifiable
 - state that the set is not unifiable otherwise
- There are many possibilities
- We go for a recursive procedure

Resolution



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

43 Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

Basic Idea

- Given a set of expressions $\{E_1, \dots, E_k\}$.
- Find a *disagreement set* (which we will define later).
- Build a substitution that can eliminate the disagreement.

Example 17 (Disagreement Elimination)

- Consider the set $\{p(a), p(x)\}$ of expressions.
- They disagree because of the arguments a and x .
- The disagreement set here is $\{a, x\}$. Since x is a variable, we can eliminate this disagreement by using the substitution $\sigma : x/a$.
- $\sigma(p(a)) = \sigma(p(x)) = p(a)$.

Resolution



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

44 Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

Disagreement Set

The *disagreement set* of a nonempty set of expressions W is obtained by *finding the first position (starting from the left) at which not all the expressions in the W have the same symbol*. Here, *parentheses and commas* are also counted as symbols. We *then extract*, from each expression, *the sub-expression that begins with the symbol occupying that position*. *The set of these sub-expressions* is the *disagreement set*.

Example 18

- Consider the set $W = \{p(x), p(a)\}$.
- The strings $p(a)$ and $p(x)$ first differ at the 3rd symbol.
- The sub-expressions beginning at the 3rd symbol are a and x , respectively.
- Disagreement set: $\{a, x\}$.



Example 19

- Consider the set
 $W = \{p(x, f(y, z)), p(x, a), p(x, g(h(k(x))))\}.$
- The strings $p(x, f(y, z))$, $p(x, a)$, and $p(x, g(h(k(x))))$ first differ at the 5th symbol.
- The sub-expressions beginning at the 5th symbol are $f(y, z)$, a , and $g(h(k(x)))$, respectively.
- Disagreement set: $\{f(y, z), a, g(h(k(x)))\}$

Resolution



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

46 Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

Unification Algorithm:

- (1) Set $k = 0$, $W_0 = W$, and $\sigma_0 = \epsilon$ (here ϵ denotes the empty substitution).
- (2) If W_k is a singleton, STOP, σ_k is a MGU. Otherwise, find the disagreement set D_k for W_k .
- (3) If there is a pair (v_k, t_k) such that $v_k, t_k \in D_k$, v_k is a variable that does not occur in the term t_k , go to step (4); Otherwise STOP, W is not unifiable.
- (4) Apply the substitution v_k/t_k to σ_k and then add v_k/t_k to the resulting set to obtain σ_{k+1} . Apply the substitution v_k/t_k to W_k to obtain W_{k+1} .
- (5) Set $k = k + 1$ and go to step (2).

Resolution



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

47 Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

Example 20

- $W_0 = W = \{p(f(g(x)), y, z), p(u, u, f(u))\}$, $\sigma_0 = \epsilon$.
- $D_0 = \{f(g(x)), u\} \Rightarrow \sigma_1 = \{u/f(g(x))\}$,
 $W_1 = \{p(f(g(x)), y, z), p(f(g(x)), f(g(x)), f(f(g(x))))\}$.
- $D_1 = \{y, f(g(x))\} \Rightarrow \sigma_2 = \{u/f(g(x)), y/f(g(x))\}$, $W_2 = \{p(f(g(x)), f(g(x)), z), p(f(g(x)), f(g(x)), f(f(g(x))))\}$.
- $D_2 = \{z, f(f(g(x)))\} \Rightarrow$
 $\sigma_3 = \{u/f(g(x)), y/f(g(x)), z/f(f(g(x)))\}$,
 $W_3 = \{p(f(g(x)), f(g(x)), f(f(g(x))))\} \Rightarrow \text{STOP and output } \sigma_3$.
- MGU: $u/f(g(x)), y/f(g(x)), z/f(f(g(x)))$. Term:
 $p(f(g(x)), f(g(x)), f(f(g(x))))$.

Example 21

- $W_0 = W = \{p(f(x), g(y)), p(z, z)\}$, $\sigma_0 = \epsilon$.
- $D_0 = \{f(x), z\} \Rightarrow \sigma_1 = \{z/f(x)\}$,
 $W_1 = \{p(f(x), g(y)), p(f(x), f(x))\}$.
- $D_1 = \{g(y), f(x)\} \Rightarrow \text{STOP, } W \text{ is not unifiable.}$

Resolution



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

48

Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

■ Complexity of unification

- The simplest unification algorithms are very fast in most cases
- Worst case: the computation time grows exponentially with the size of the terms.
- In practice, nearly all unification attempts fail, in most cases the worst case complexity has no dramatic effect
- The fastest unification algorithms have nearly linear complexity [Bibel 1982]

Exercise 6 ([Ertel 2025], Exercise 3.5, p. 66)

Unify (if possible) the following terms and give the MGU and the resulting terms

- (a) $p(x, f(y))$ and $p(f(z), u)$
- (b) $p(x, f(x))$ and $p(y, y)$
- (c) $x = 4 - 7 \cdot x$ and $\cos y = z$
- (d) $x < 2 \cdot x$ and $3 < 6$
- (e) $q(f(x, y, z), f(g(w, w), g(x, x), g(y, y)))$ and $q(u, u)$

Resolution



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

49 Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

Generalized Resolution Rule for PL1

The resolution rule for two clauses in conjunctive normal form reads

$$\frac{A_1 \vee \dots \vee A_m \vee B, \neg B' \vee C_1 \vee \dots \vee C_n \quad \sigma(B) = \sigma(B')}{\sigma(A_1) \vee \dots \vee \sigma(A_m) \vee \sigma(C_1) \vee \dots \vee \sigma(C_n)},$$

where σ is the MGU of B and B'

What does this definition mean?

- Premise 1: $A_1 \vee \dots \vee A_m \vee B$
- Premise 2: $\neg B' \vee C_1 \vee \dots \vee C_n$
- $\sigma(B) = \sigma(B')$ means that B and B' are matched by applying the MGU σ
- Apply σ for every literal in each premise
- Now, Premise 1 becomes $\sigma(A_1) \vee \dots \vee \sigma(A_m) \vee \sigma(B)$ and Premise 2 becomes $\neg \sigma(B') \vee \sigma(C_1) \vee \dots \vee \sigma(C_n)$
- The usual resolution rule can now be applied

Theorem 6 (Correctness)

The resolution rule is correct. That is, the resolvent is a semantic consequence of the two parent clauses

Resolution



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

50 Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

For **Completeness**, we need an additional inference rule

Factorization Rule for PL1

Factorization of a clause is accomplished by

$$\frac{A_1 \vee A_2 \vee \cdots \vee A_n, \quad \sigma(A_1) = \sigma(A_2)}{\sigma(A_2) \vee \cdots \vee \sigma(A_n)},$$

where σ is the MGU of A_1 and A_2

What does this definition mean?

- Premise: $A_1 \vee A_2 \vee \cdots \vee A_n$
- $\sigma(A_1) = \sigma(A_2)$ means that A_1 and A_2 are matched after their MGU σ is applied
- Apply σ for every literal in the premise
- Now, the Premise becomes $\sigma(A_1) \vee \sigma(A_2) \vee \cdots \vee \sigma(A_n)$
- As $p \vee p \equiv p$, the Conclusion $\sigma(A_2) \vee \cdots \vee \sigma(A_n)$ is derived
- Intuitively, after σ is applied, as $\sigma(A_1) = \sigma(A_2)$, one of these literals becomes “redundant” and can be “removed”

Resolution



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

51 Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

Example 22

- Russell's paradox: *There is a barber who shaves everyone who does not shave himself*
- $Q \equiv \forall x [shaves(barber, x) \Leftrightarrow \neg shaves(x, x)] \equiv$
 $\forall x [(\neg shaves(barber, x) \vee \neg shaves(x, x)) \wedge$
 $(shaves(x, x) \vee shaves(barber, x))]$
- Proof (that Q is contradictory):

Step	Proved by
1. $\neg shaves(barber, x) \vee \neg shaves(x, x)$	Q
2. $shaves(x, x) \vee shaves(barber, x)$	Q
3. $\neg shaves(barber, barber)$	$Fak(1, \sigma : x / barber)$
4. $shaves(barber, barber)$	$Fak(2, \sigma : x / barber)$
5. $()$	$Res(3, 4)$

Note: Without the Factorization Rule, from the two clauses 1 and 2, we can derive several tautologies, but no contradiction

Resolution



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

52 Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

Exercise 7 ([Ertel 2025], Exercise 3.6, p. 66)

- (a) Transform Russell's Paradox from Example 22 into CNF
- (b) Show that the empty clause cannot be derived using resolution without factorization. Try to understand this intuitively

Resolution



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

53 Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

Theorem 7

The resolution rule together with the factorization rule is refutation complete. That is, by application of factorization and resolution steps, the empty clause can be derived from any unsatisfiable formula in conjunctive normal form

Note

- The *search for a proof* can be *very frustrating in practice*
 - Even when $KB \wedge \neg Q$ has only a few clauses, *every resolution step generates a new clause* which increases the number of possible resolution steps in the next iteration

Resolution Strategies

- Unit Resolution
- Set of Support (SOS) Strategy
- Input Resolution
- Pure Literal Rule
- Subsumption

Resolution



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

54

Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

Unit Resolution

- Prioritizes resolution steps in which one of the two clauses *consists of only one literal*, called a *unit clause*
- Complete
- Heuristic (search space reduction not guaranteed)

Set of Support (SOS) $\subset KB \wedge \neg Q$

- *Every resolution step* must *involve a clause from the SOS*
- The *resolvent* is *added to the SOS*
- Search space reduction guaranteed
- Not complete
- Complete, if $(KB \wedge \neg Q) \setminus SOS$ is satisfiable
- Often, the negated query $\neg Q$ is used as the initial SOS

Resolution



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

55 Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

Input Resolution

- *A clause from the input set $KB \wedge \neg Q$ must be involved in every resolution step*
- Search space reduction guaranteed
- Not complete

Pure Literal Rule

- All clauses that contain *literals for which there are no complementary literals in other clauses* are *deleted*
- Search space reduction guaranteed
- Complete
- Is used by practically all resolution provers

Resolution



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

56 Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

Subsumption

- If, *after applying some substitution*, the literals of a clause K_1 form a subset of the literals of clause K_2 , then K_2 *can be deleted*
 - Example: $K_1 = \{P(x)\}$ and $K_2 = \{P(a), Q(b)\}$. With $x \mapsto a$, K_1 becomes $\{P(a)\} \subseteq K_2$, so K_2 is redundant.
- Search space reduction guaranteed
- Complete
- Is used by practically all resolution provers

Resolution



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

57 Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

Exercise 8 ([Ertel 2025], Exercise 3.7, p. 66)

- (a) Why is resolution with the set of support strategy incomplete?
- (b) Justify (without proving) why the set of support strategy becomes complete if $(KB \wedge \neg Q) \setminus \text{SOS}$ is satisfiable
- (c) Why is resolution with the pure literal rule complete?

Resolution



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

58 Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

Equality is an especially inconvenient cause of *explosive growth of the search space*

Example 23

■ Knowledge Base KB includes

■ Equality Axioms

■ $\forall x \ x = x$

■ $\forall x \ \forall y \ x = y \Rightarrow y = x$

■ $\forall x \ \forall y \ \forall z \ x = y \wedge y = z \Rightarrow x = z$

■ Substitution Axioms

■ $\forall x \ \forall y \ x = y \Rightarrow f(x) = f(y)$, for every function symbol f

■ $\forall x \ \forall y \ x = y \Rightarrow p(x) \Leftrightarrow p(y)$, for every predicate symbol p

■ For an n -place symbol, one such axiom is needed per argument, e.g. $x = y \Rightarrow f(x, z) = f(y, z)$ for a two-place f

- The symmetry clause $\neg x = y \vee y = x$ *can be unified with every positive or negated equation*. This leads to the *derivation of new clauses and equations upon which equality axioms can again be applied, and so on*. (Most of these new clauses are not helpful to a proof)

Resolution



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

59 Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

- *Special inference rules for equality have been developed* which get by without explicit equality axioms and, in particular, reduce the search space
 - **Demodulation:** Take a clause $t_1 = t_2$ and some clause α that contains the term t_1 , and yields a new clause formed by substituting t_2 for t_1 within α . Note that *demodulation is directional*; *given $t_1 = t_2$, the t_1 always gets replaced with t_2 , never vice versa*

$$\frac{t_1 = t_2, (\dots t \dots), \quad \sigma(t_1) = \sigma(t)}{(\dots \sigma(t_2) \dots)}$$

- Premise 1: $t_1 = t_2$
- Premise 2: A formula $\alpha = (\dots t \dots)$
- $\sigma(t_1) = \sigma(t)$ means that the terms t_1 and t are matched after σ is applied
- The substitution σ is applied for α , t_1 , and t_2
- Now, α becomes $(\dots \sigma(t_1) \dots)$
- Replace $\sigma(t_1)$ by $\sigma(t_2)$ everywhere in α , we get the Conclusion $(\dots \sigma(t_2) \dots)$
- **Paramodulation:** More general rule, works with conditional equations

Resolution



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

60 Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

Example 24 (Demodulation)

- KB contains
 - $father(father(x)) = grandfather(x)$
 - $birthdate(father(father(John)), 1945)$
- By demodulation, we get
 - $birthdate(grandfather(John), 1945)$

Automated Theorem Provers



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

Resolution

61 Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

Theorem Provers: Implementations of proof calculi on computers.

■ Otter, 1984

- One of the oldest resolution provers (with equality handling).
- L. Wos, W. McCune: Argonne National Laboratory, Chicago.
- Otter was successfully applied in specialized areas of mathematics: “Currently, the main application of Otter is research in abstract algebra and formal logic. Otter and its predecessors have been used to answer many open questions in the areas of finite semigroups, ternary Boolean algebra, logic calculi, combinatory logic, group theory, lattice theory, and algebraic geometry.”

■ SETHEO, 1987

- Early automated theorem prover for first-order logic.
- W. Bibel, J. Schumann, R. Letz: Munich Technical University.
- PARTHEO: implementation on parallel computers.

Automated Theorem Provers



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

Resolution

62

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Appendix: Using E

87

■ E, 2000

- Modern equation prover.
- S. Schulz: Munich Technical University.
- Homepage of E: “E is a purely equational theorem prover for clausal logic. That means it is a program that you can stuff a mathematical specification (in clausal logic with equality) and a hypothesis into, and which will then run forever, using up all of your machines resources. Very occasionally it will find a proof for the hypothesis and tell you so ;-).”

■ Vampire

- Resolution with equality handling.
- A. Voronkov: University of Manchester, England.
- Winner of CADE (CADE = Conference on Automated Deduction) ATP (Automated Theorem Prover) System Competition from 2001 to 2006.

■ Isabelle, 2002

- Interactive prover for higher-order predicate logic
- T. Nipkov, L. Paulson, M. Wenzel: Univ. Cambridge, Munic Techn. Univ.

Mathematical Examples



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

63 Mathematical
Examples

Applications

Summary

References

Appendix: Using E

We now wish to demonstrate the application of an automated prover with the aforementioned prover E. E is a specialized equality prover which greatly shrinks the search space through an optimized treatment of equality.

Definition

A structure $(M, \cdot)$ consisting of a set M with a two-place inner operation “ $\cdot$ ” is called a semigroup if the law of associativity

$$\forall x \forall y \forall z (x \cdot y) \cdot z = x \cdot (y \cdot z)$$

holds. An element $e \in M$ is called *left-neutral* (*right-neutral*) if $\forall x e \cdot x = x$ ($\forall x x \cdot e = x$).

Theorem 8

If a semigroup has a left-neutral element e_l and a right-neutral element e_r , then $e_l = e_r$.

Mathematical Examples



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

64 Mathematical
Examples

Applications

Summary

References

Appendix: Using E

First, we prove by intuitive mathematical reasoning.

Proof of Theorem 8.

For every $x \in M$, it holds that

$$e_l \cdot x = x \quad (1)$$

$$x \cdot e_r = x \quad (2)$$

Replacing $x = e_r$ in Eq. (1) and $x = e_l$ in Eq. (2), we have

$$e_l \cdot e_r = e_r \quad (3)$$

$$e_l \cdot e_r = e_l \quad (4)$$

Combining Eq. (3) and Eq. (4), we have $e_l = e_l \cdot e_r = e_r$. $\square$

Mathematical Examples



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

65 Mathematical
Examples

Applications

Summary

References

Appendix: Using E

Next, we carry out the resolution proof manually. The function $m(x, y) = x \cdot y$.

- Negated query $(\neg e_l = e_r)_1$
- Knowledge Base KB
 - Definitions of semi-groups and left-/right- neutrals.

$$(m(m(x, y), z) = m(x, m(y, z)))_2 \quad \text{associativity}$$

$$(m(e_l, x) = x)_3 \quad \text{left-neutral}$$

$$(m(x, e_r) = x)_4 \quad \text{right-neutral}$$

- Equality axioms (for comparing terms)

$$(x = x)_5 \quad \text{reflexive}$$

$$(\neg x = y \vee y = x)_6 \quad \text{symmetry}$$

$$(\neg x = y \vee \neg y = z \vee x = z)_7 \quad \text{transitivity}$$

$$(\neg x = y \vee m(x, z) = m(y, z))_8 \quad \text{substitution for } m$$

$$(\neg x = y \vee m(z, x) = m(z, y))_9 \quad \text{substitution for } m$$

Mathematical Examples



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

66 Mathematical
Examples

Applications

Summary

References

Appendix: Using E

A resolution proof may be as follows.

Step	Proved by
10. $x = m(e_l, x)$	$\text{Res}(3, 6, x_6/m(e_l, x_3), y_6/x_3)$
11. $\neg m(e_l, x) = z \vee x = z$	$\text{Res}(7, 10, x_7/x_{10}, y_7/m(e_l, x_{10}))$
12. $e_r = e_l$	$\text{Res}(4, 11, x_4/e_l, x_{11}/e_r, z_{11}/e_l)$
13. $e_l = e_r$	$\text{Res}(6, 12, x_6/e_r, y_6/e_l)$
14. $()$	$\text{Res}(1, 13, \emptyset)$

For example, $\text{Res}(3, 6, x_6/m(e_l, x_3), y_6/x_3)$ means that in the resolution of clause 3 with clause 6, the x in clause 6 is replaced by $m(e_l, x)$ where the variable x is from clause 3 and y from clause 6 is replaced by x from clause 3.

Note: Unlike Ertel — who writes $\text{Res}(1, 12, \emptyset)$ directly — we make the symmetry step (step 13) explicit, since clause 1 ($\neg e_l = e_r$) and step 12 ($e_r = e_l$) are not directly complementary. Ertel does note that the proof relies on equality being symmetric and transitive.

Mathematical Examples



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

67 Mathematical
Examples

Applications

Summary

References

Appendix: Using E

Now we want to apply the prover E (<https://www.lehre.dhbw-stuttgart.de/~sschulz/E/E.html>) to the problem.

Transformation in clause normal form language LOP (a clause input language accepted by E).

$$(\neg A_1 \vee \dots \vee \neg A_m \vee B_1 \vee \dots \vee B_n) \mapsto B_1; \dots; B_n \leftarrow A_1, \dots, A_m$$

An input file for E

halbgr1.lop

```
1  <- eq(e1,er). % query
2  eq( m(m(X,Y),Z), m(X,m(Y,Z)) ). % associativity of m
3  eq( m(e1,X), X ). % left-neutral element of m
4  eq( m(X,er), X ). % right-neutral element of m
5  eq(X,X). % equality: reflexivity
6  eq(Y,X) <- eq(X,Y). % equality: symmetry
7  eq(X,Z) <- eq(X,Y), eq(Y,Z). % equality: transitivity
8  eq( m(X,Z), m(Y,Z) ) <- eq(X,Y). % equality: substitution in m
9  eq( m(Z,X), m(Z,Y) ) <- eq(X,Y). % equality: substitution in m
```

Mathematical Examples



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

68 Mathematical
Examples

Applications

Summary

References

Appendix: Using E

Calling the prover with

`eprover --proof-object halbgr1.lop | epclextract.` The
output from E 3.3.5:

```
0 : : (eq(X1,X2)|~(eq(X2,X1))) :  
    initial("halbgr1.lop", at_line_6_column_1)  
1 : : (eq(X1,X2)|(~(eq(X1,X3))|~(eq(X3,X2)))) :  
    initial("halbgr1.lop", at_line_7_column_1)  
2 : : eq(m(e1,X1),X1) : initial("halbgr1.lop", at_line_3_column_1)  
3 : : ~(eq(e1,er)) : initial("halbgr1.lop", at_line_1_column_1)  
4 : : eq(m(X1,er),X1) : initial("halbgr1.lop", at_line_4_column_1)  
5 : : (eq(X1,X2)|~(eq(X2,X1))) : fof_simplification(0)  
6 : : (eq(X1,X2)|(~(eq(X1,X3))|~(eq(X3,X2)))) : fof_simplification(1)  
7 : : [++eq(X1,X2),--eq(X2,X1)] : 5  
8 : : [++eq(m(e1,X1),X1)] : 2  
9 : : ~(eq(e1,er)) : fof_simplification(3)  
10 : : [++eq(X1,X2),--eq(X1,X3),--eq(X3,X2)] : 6  
11 : : [++eq(X1,m(e1,X1))] : pm(7,8)  
12 : : [--eq(e1,er)] : 9  
13 : : [++eq(X1,X2),--eq(m(e1,X1),X2)] : pm(10,11)  
14 : : [++eq(m(X1,er),X1)] : 4  
15 : : [--eq(er,e1)] : pm(12,7)  
16 : : [] : sr(pm(13,14),15) : 'proof'
```

Mathematical Examples



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

69 Mathematical
Examples

Applications

Summary

References

Appendix: Using E

- Rows marked with `initial` are clauses from the input data. E proof objects list dependencies before the steps that use them [Schulz 2019], Sec. 6.4.
- `++` marks a positive literal; `--` marks a negative literal.
- `pm(a, b)` means that E combines rows `a` and `b` using equality: one equal expression is replaced by the other. This rule is called paramodulation [Schulz 2019], Sec. 3.1.
- `sr(A, B)` means that E uses row `B` to remove an opposite equality literal from row `A`. This rule is called simplify-reflect.
- In the compact proof on the next slide, read `rw(6,7)` as a rewrite of row 6 using row 7; `cn(...)` records E's normalization of that final simplification.
- The proof found by E is somewhat similar to the manually created proof.

Mathematical Examples



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

70 Mathematical
Examples

Applications

Summary

References

Appendix: Using E

Because we explicitly model the equality by the predicate `eq`, the particular strengths of `E` do not come into play. Now we omit the equality axioms and obtain

halbgr2.lop

```
1  <- el = er. % query
2  m(m(X,Y),Z) = m(X,m(Y,Z)) . % associativity of m
3  m(el,X) = X . % left-neutral element of m
4  m(X,er) = X . % right-neutral element of m
```

as the input.

The proof also becomes more compact.

```
0 : : ~(equal(el,er)) : initial("halbgr2.lop", at_line_1_column_1)
1 : : equal(m(X1,er),X1) : initial("halbgr2.lop", at_line_4_column_1)
2 : : equal(m(el,X1),X1) : initial("halbgr2.lop", at_line_3_column_1)
3 : : ~(equal(el,er)) : fof_simplification(0)
4 : : [++equal(m(X1,er), X1)] : 1
5 : : [++equal(m(el,X1), X1)] : 2
6 : : [--equal(el, er)] : 3
7 : : [++equal(er, el)] : pm(4,5)
8 : : [] : cn(rw(6,7)) : 'proof'
```

From the above output, the proof consists essentially of a single inference step on the two relevant clauses 4 and 5.

Mathematical Examples



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

71 Mathematical
Examples

Applications

Summary

References

Appendix: Using E

Exercise 9 ([Ertel 2025], Exercise 3.8, p. 66)

Formalize and prove with resolution that in a semigroup with at least two different elements a, b , a left-neutral element e (i.e., for every element x , we have $e \cdot x = x$), and a left null element n (i.e., for every element x , we have $n \cdot x = n$), these two elements have to be different, that is, that $n \neq e$. Use demodulation.

Exercise 10 ([Ertel 2025], Exercise 3.9, p. 66)

Obtain the theorem prover E (<https://www.lehre.dhbw-stuttgart.de/~sschulz/E/E.html>) or another prover and prove the following statements. Compare these proofs with those in the text.

- (a) The claim from Example 7 in Lecture “Propositional Logic”.
- (b) Russell’s paradox from Example 22.
- (c) The claim from Exercise 9.



Some applications of Automated Theorem Provers.

- Four color theorem was first proved in 1976 with the help of a special prover.
- Inference in expert systems.
 - Little use nowadays due to the problems of predicate logic in modeling uncertainty
- Automated program verification.
 - For example, in software engineering, a proof of certain security characteristics of a program is required.
 - Such a proof cannot be brought about through testing of a finished program, for in general it is impossible to apply a program to all possible inputs. This is therefore an ideal domain for general or even specialized inference systems.
- Software reuse.

Example 25 (Software reuse)

■ Specification of the query

PRE_Q : the preconditions, which must hold before the desired program is applied.

$POST_Q$: the postconditions, which must hold after the desired program is applied.

- **Task:** Search a software database for modules, which fulfil these requirements. For each module M , the software database contains a description of the preconditions PRE_M and the postconditions $POST_M$.

- Thus it must hold: $PRE_Q \Rightarrow PRE_M$.

- If, for example, a module in the database only accepts lists of integers, then lists of integers as input must also appear as preconditions in the query.
- An additional requirement in the query that, for example, only even numbers appear, doesn't cause a problem.

- It must also hold: $POST_M \Rightarrow POST_Q$.

- After the application of the module, all properties required by the query must be fulfilled.

Summary



First-order Predicate Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and Normal Forms

Proof Calculi

Resolution

Automated Theorem Provers

Mathematical Examples

Applications

74 Summary

References

Appendix: Using E

- The proof of mathematical theorems can be automated.
- Automated provers can be used for verification tasks.
- PL1 is not suitable for reasoning in everyday life.

References



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

75 References

Appendix: Using E



Ertel, Wolfgang (2025). *Introduction to Artificial Intelligence*. 3rd. Springer. DOI:

10.1007/978-3-658-43102-0.



Schulz, Stephan (Oct. 27, 2019). *E 2.4 User Manual*.

URL: https://www.lehre.dhbw-stuttgart.de/~sschulz/WORK/E_DOWNLOAD/V_2.4/eprover.pdf (visited on 07/03/2026).



Schulz, Stephan (2002). “E—A Brainiac Theorem Prover.” In: *AI Communications* 15.2–3, pp. 111–126. URL: <https://www.lehre.dhbw-stuttgart.de/~sschulz/E/E.html>.



Eder, Elmar (1992). *Relative Complexities of First Order Calculi*. Springer-Verlag. DOI:

10.1007/978-3-322-84222-0.



Bibel, Wolfgang (1982). *Automated Theorem Proving*. Springer Science & Business Media. DOI:

10.1007/978-3-322-90100-2.

References (cont.)



First-order Predicate Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and Normal Forms

Proof Calculi

Resolution

Automated Theorem Provers

Mathematical Examples

Applications

Summary

76 References

Appendix: Using E



Robinson, John Alan (1965). “A Machine-Oriented Logic Based on the Resolution Principle.” In: *Journal of the ACM* 12.1, pp. 23–41. DOI: 10.1145/321250.321253.



Gödel, Kurt (1930). “Die Vollständigkeit der Axiome des logischen Funktionenkalküls.” In: *Monatshefte für Mathematik und Physik* 37, pp. 349–360. DOI: 10.1007/BF01696781.

Appendix: Using E

What this appendix covers



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

77

Appendix: Using E

87

- The goal is only to read the E examples used in this course.
- E is an automated theorem prover for first-order logic with equality [Schulz 2002].
- In this lecture, we use E as a *refutational prover*: add the negated query and look for a contradiction.
- We will not discuss E's search strategies, advanced options, or proof optimization.

Appendix: Using E

From consequence to refutation



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

- To prove that a query Q follows from a knowledge base KB , E works with

$$KB \wedge \neg Q.$$

- If E derives the *empty clause* $[]$, then $KB \wedge \neg Q$ is contradictory.
- Therefore, Q follows from KB .
- If E reports no proof, read this carefully: it means *no contradiction was derived in that run*; it is not a proof of $\neg Q$.

Appendix: Using E

Minimal LOP syntax



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

LOP is a compact clause input language for E theorem-proving examples [Schulz 2019].

```
% Goal: prove animal(fido).  
<- animal(fido).           % negated query  
  
% Knowledge base.  
dog(fido).                  % fact  
mammal(X) <- dog(X).        % rule  
animal(X) <- mammal(X).    % rule
```

- In LOP input, % starts a comment that runs to the end of the line.
- Comment lines and blank lines are ignored as formulas, but they still count when E reports input line numbers.
- E also accepts block comments of the form /* ... */; we use % for short examples.

79

Appendix: Using E

Appendix: Using E

Running E



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

For the examples in this course, use the proof-object output and then extract the compact proof:

```
eprover --proof-object file.lop | epclextract
```

- The current course output was checked with *E 3.3.5 Countess Grey*.
- The important result lines are:

```
% Proof found!
% SZS status Unsatisfiable
...
12 :   : [] : cn(rw(pm(9, 10), 11)) : 'proof'
```

The % lines are E's status/comment output. The numbered rows are the proof object; here, [] is the contradiction and 'proof' marks the proof row.

80

Appendix: Using E

87

Appendix: Using E

Reading output: conventions



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

The next slide explains a compact E proof output part by part. Keep these conventions in mind [Schulz 2019], Sec. 3.1 and Sec. 6.4.

- Rows tagged `initial` come from the input file.
- Rows tagged `fof_simplification` are E's simplified forms.
- In bracketed clauses, `++` marks a positive literal, `--` marks a negative literal, and commas mean OR.
- Example: `[++mammal(X1),-dog(X1)]` means

$$\text{mammal}(X1) \vee \neg \text{dog}(X1).$$

- The final row `[]` means the empty clause, so the negated query led to a contradiction.

81

Appendix: Using E

87

Appendix: Using E

Reading the proof output



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Output from E 3.3.5 for the dog/mammal/animal example:

```
0 : : (mammal(X1)|~dog(X1)) : initial(line 6)
1 : : ~animal(fido) : initial(line 2)
2 : : (animal(X1)|~mammal(X1)) : initial(line 7)
3 : : dog(fido) : initial(line 5)
4 : : (mammal(X1)|~dog(X1)) : fof_simplification(0)
5 : : ~animal(fido) : fof_simplification(1)
6 : : (animal(X1)|~mammal(X1)) : fof_simplification(2)
7 : : [++mammal(X1),--dog(X1)] : 4
8 : : [++dog(fido)] : 3
9 : : [--animal(fido)] : 5
10 : : [++animal(X1),--mammal(X1)] : 6
11 : : [++mammal(fido)] : pm(7,8)
12 : : [] : cn(rw(pm(9,10),11)) : 'proof'
```

Rows 0–3: input. These rows come from the LOP file: two rules, the negated query, and the fact `dog(fido)`. The % comment lines and blank lines only affect the reported line numbers. **Rows 4–10: preparation.** E simplifies the input and prints selected rows as bracketed clauses. For example, row 7 says $mammal(X1) \vee \neg dog(X1)$. **Row 11.** E combines row 7 with row 8. In ordinary rule language: from “dogs are mammals” and `dog(fido)`, it derives `mammal(fido)`. **Row 12**

82

Appendix: Using E

87

Appendix: Using E

Inference tags



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

- pm(i, j)** In equality proofs, this is a paramodulation step: use an equality from one parent row to replace a matching expression in the other [Schulz 2019], Sec. 3.1. In simple Horn outputs, first read it as: rows *i* and *j* are the parents of this derived row.
- spm(i, j)** Superposition/paramodulation-style step. If this appears in another E output, read it as the same family of equality-based replacement steps.
- rw(i, j)** Rewrite: simplify row *i* using row *j*.
- sr(A, B)** Simplify-reflect: simplify a clause by removing a literal contradicted by another row.
- cn(...)** Clause normalization: final cleanup of the derived clause.

These tags describe *how E obtained a row*; they are not new logical connectives.

Appendix: Using E

Tiny equality example



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Equality is where E's special inference rules become most visible.

```
% Goal: prove registered(fido).  
<- registered(fido).  
  
% Equality fact.  
fido = rex.  
  
% Predicate fact.  
registered(rex).
```

```
4 : [++registered(rex)] : 1  
5 : [++equal(fido, rex)] : 2  
6 : [--registered(fido)] : 3  
7 : [++registered(fido)] : rw(4,5)  
8 : [] : cn(rw(6,7)) : 'proof'
```

Read row 7 as: because `fido = rex`, E rewrites `registered(rex)` into `registered(fido)`.

Appendix: Using E

Practice exercises



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Exercise 11

For each item, write an E input file with extension `.lop`. Add % comments that label the negated query, facts, and rules; then run

```
eprover --proof-object <file> | epclextract
```

and identify the row where E derives []. Do not reuse the semigroup/equality, Russell, high-jump, or fido/rex examples from the lecture.

- (a) Facts: `powered(lab1)` and `switch_on(lab1)`. Rule: if a room is powered and its switch is on, then it is lit. Query: `lit(lab1)`.
- (b) Facts: `enrolled(mai, ai)`, `passed(mai, logic)`, and `prereq(logic, ai)`. Rules: a student is ready for a course if enrolled in it and has passed a prerequisite; a ready student can take the exam. Query: `can_take_exam(mai, ai)`.

Appendix: Using E

Practice exercises



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

Exercise 12

Write an E input file with extension `.lop` for the following equality case. Add % comments that label the negated query, the equality fact, and the predicate fact:

- Equality fact: `badge(mai) = id17.`
- Predicate fact: `cleared(id17).`
- Query: `cleared(badge(mai)).`

Run

```
eprover --proof-object <file> | epclextract
```

and compare the final row with the tiny equality example above.

Appendix: Using E

What E is not



First-order Predicate
Logic

Hoàng Anh Đức

Motivation

Syntax

Semantics

Quantifiers and
Normal Forms

Proof Calculi

Resolution

Automated Theorem
Provers

Mathematical
Examples

Applications

Summary

References

87

Appendix: Using E

87

- E is not an interactive calculator for formulas.
- E searches for a contradiction from clauses.
- Therefore, E output should be read as *proof evidence*: it records how the empty clause was derived.
- If E does not find a proof in a run, that does not by itself prove that the query is false.