

COPYRIGHT NOTICE / THÔNG BÁO BẢN QUYỀN

© 2025 Duc A. Hoang (Hoàng Anh Đức)

COPYRIGHT (English):

This document is licensed under Creative Commons Attribution-ShareAlike 4.0 International (CC-BY-SA 4.0). You are free to share and adapt this material with appropriate attribution and under the same license.

This document is not up to date and may contain several errors or outdated information.

Last revision date: 2025-09-22

BẢN QUYỀN (Tiếng Việt):

Tài liệu này được cấp phép theo Giấy phép Quốc tế Creative Commons Attribution-ShareAlike 4.0 (CC-BY-SA 4.0). Bạn được tự do chia sẻ và chỉnh sửa tài liệu này với điều kiện ghi nguồn phù hợp và sử dụng cùng loại giấy phép.

Tài liệu này không được cập nhật và có thể chứa nhiều lỗi hoặc thông tin cũ.

Ngày sửa đổi cuối cùng: 2025-09-22



Creative Commons Attribution-ShareAlike 4.0 International

VNU-HUS MAT1206E/3508: Introduction to AI

First-order Predicate Logic

In-class Discussion

Hoàng Anh Đức

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



Contents



First-order Predicate Logic

Hoàng Anh Đức

Introduction

Basics of First-order Predicate Logic

Syntax and Semantics

Variable Replacement and Substitution

Quantifiers and Normal Forms

Proof Calculi for Predicate Logic

Automated Theorem Provers

References

Introduction

Basics of First-order Predicate Logic

Syntax and Semantics

Variable Replacement and Substitution

Quantifiers and Normal Forms

Proof Calculi for Predicate Logic

Automated Theorem Provers



- Logic is one of the oldest areas in AI and was the dominant approach from the 1950s through the 1980s (recall symbolic AI).
- Today, machine learning—especially statistical and deep learning methods—drives most practical applications, and logic-based methods are less prominent in mainstream AI.
- Still, logic is crucial for understanding AI's foundations and for applications that require explicit, interpretable, and verifiable reasoning.
 - Task planning for service robots.
 - Verification and decision-making in autonomous driving.
 - Combining symbolic knowledge in predicate logic with sub-symbolic sensor data (e.g., via feature extraction from deep learning) is a promising research direction.
- Building on our earlier discussion of propositional logic, we now introduce predicate logic as a more expressive framework for knowledge representation and reasoning.

Basics of First-order Predicate Logic

Syntax and Semantics



First-order Predicate Logic

Hoàng Anh Đức

Introduction

Basics of First-order Predicate Logic

3 Syntax and Semantics

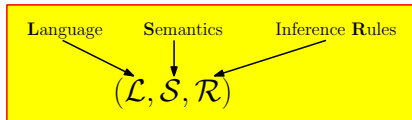
Variable Replacement and Substitution

Quantifiers and Normal Forms

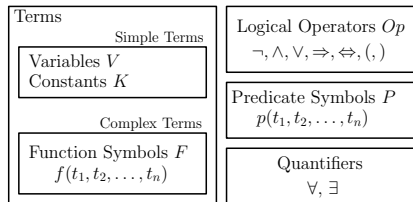
Proof Calculi for Predicate Logic

Automated Theorem Provers

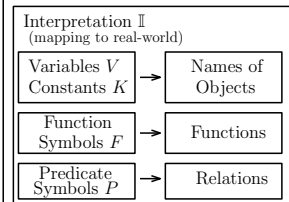
References



Predicate Logic Formulas (Syntax)



Predicate Logic Formulas (Semantics)



Predicate Logic Formulas
(Premise)

Derivation
(Applying inference rules)

Predicate Logic Formula
(Conclusion)

Basics of First-order Predicate Logic

Syntax and Semantics



First-order Predicate Logic

Hoàng Anh Đức

Introduction

Basics of First-order Predicate Logic

3 Syntax and Semantics

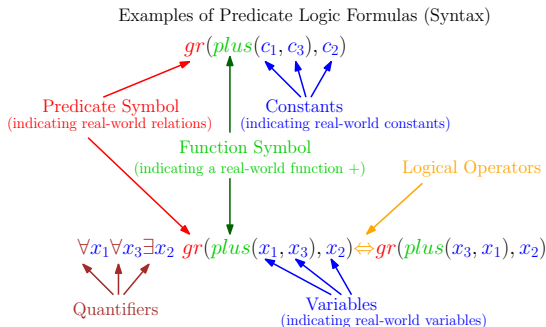
Variable Replacement and Substitution

Quantifiers and Normal Forms

Proof Calculi for Predicate Logic

Automated Theorem Provers

References



Basics of First-order Predicate Logic

Syntax and Semantics



First-order Predicate Logic

Hoàng Anh Đức

Introduction

Basics of First-order Predicate Logic

3 Syntax and Semantics

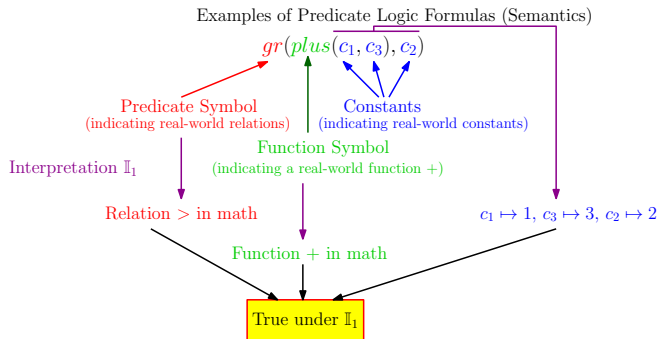
Variable Replacement and Substitution

Quantifiers and Normal Forms

Proof Calculi for Predicate Logic

Automated Theorem Provers

References



Basics of First-order Predicate Logic

Syntax and Semantics



First-order Predicate Logic

Hoàng Anh Đức

Introduction

Basics of First-order Predicate Logic

3 Syntax and Semantics

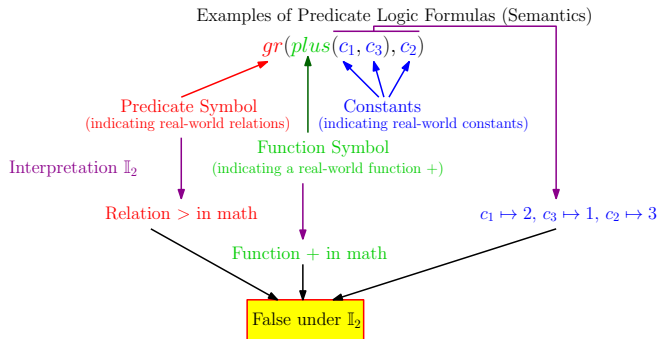
Variable Replacement and Substitution

Quantifiers and Normal Forms

Proof Calculi for Predicate Logic

Automated Theorem Provers

References



Basics of First-order Predicate Logic

Syntax and Semantics



First-order Predicate Logic

Hoàng Anh Đức

Introduction

Basics of First-order Predicate Logic

4

Syntax and Semantics

Variable Replacement and Substitution

Quantifiers and Normal Forms

Proof Calculi for Predicate Logic

Automated Theorem Provers

References

Exercise 1

Review the following concepts:

- (a) Variables, Constants, Function Symbols, Predicate Symbols
- (b) (Simple/Complex) Terms
- (c) Predicate-Logic Formulas, Literals, Free Variables
- (d) CNF, Horn Formulas
- (e) Interpretation (Assignment), Truth Value of a Formula
- (f) Semantics Equivalence, Satisfiable/Unsatisfiable/Valid Formulas, Models, Semantics Entailment

Basics of First-order Predicate Logic

Syntax and Semantics



First-order Predicate
Logic

Hoàng Anh Đức

Introduction

Basics of First-order
Predicate Logic

5

Syntax and Semantics

Variable Replacement and
Substitution

Quantifiers and Normal
Forms

Proof Calculi for Predicate
Logic

Automated Theorem
Provers

References

Exercise 2

- (a) What are the input and output of a function symbol? (**Hint:** Think about what a function symbol like *plus*(c_1, c_3) would take as input and what it would return as output)
- (b) What are the input and output of a predicate symbol? (**Hint:** Think about what a predicate symbol like *greater*(c_1, c_3) would take as input and what it would return as output)
- (c) Are there any differences between a function symbol and a real function in mathematics? Similarly, are there any differences between a predicate symbol and a real relation? (**Hint:** Think about what a function symbol or a predicate symbol **represents** in real life)

Basics of First-order Predicate Logic

Syntax and Semantics



First-order Predicate Logic

Hoàng Anh Đức

Introduction

Basics of First-order Predicate Logic

6

Syntax and Semantics

Variable Replacement and Substitution

Quantifiers and Normal Forms

Proof Calculi for Predicate Logic

Automated Theorem Provers

References

Exercise 3

Using the concepts mentioned in the previous exercises, explain the family tree example in [Ertel 2025], Example 3.2, p. 45. You may start by answering the following questions:

- (a) Which concepts from the above list are used in the example?
- (b) What are the “meanings” of the predicate symbols in the example? Give examples of interpretations (assignments) which make the predicates true/false.
- (c) What is the knowledge base KB formed in the example? Can you give an example of a query Q which we hope to derive from KB (other than the given query in the example)?

Basics of First-order Predicate Logic

Syntax and Semantics



First-order Predicate Logic

Hoàng Anh Đức

Introduction

Basics of First-order Predicate Logic

Syntax and Semantics

Variable Replacement and Substitution

Quantifiers and Normal Forms

Proof Calculi for Predicate Logic

Automated Theorem Provers

References

7

42

- 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

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

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

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

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

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

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

Basics of First-order Predicate Logic

Syntax and Semantics



First-order Predicate Logic

Hoàng Anh Đức

Introduction

Basics of First-order Predicate Logic

8

Syntax and Semantics

Variable Replacement and Substitution

Quantifiers and Normal Forms

Proof Calculi for Predicate Logic

Automated Theorem Provers

References

Exercise 4

Can you define the predicate “ $<$ ” in a similar way as the equality “ $=$ ”?

Basics of First-order Predicate Logic

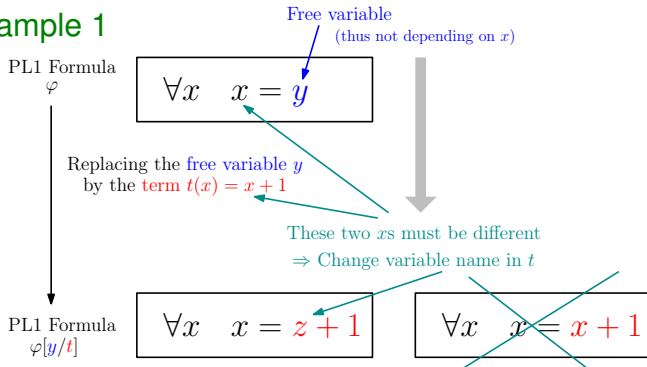
Variable Replacement and Substitution



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* to ensure this

Example 1



First-order Predicate Logic

Hoàng Anh Đức

Introduction

Basics of First-order Predicate Logic

Syntax and Semantics

Variable Replacement and Substitution

Quantifiers and Normal Forms

Proof Calculi for Predicate Logic

Automated Theorem Provers

References

Basics of First-order Predicate Logic

Variable Replacement and Substitution



First-order Predicate
Logic

Hoàng Anh Đức

Introduction

Basics of First-order
Predicate Logic

Syntax and Semantics

10 Variable Replacement and
Substitution

Quantifiers and Normal
Forms

Proof Calculi for Predicate
Logic

Automated Theorem
Provers

References

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*
- As before, *we do not allow any variables in the term t_i that are quantified in t* . In those cases *variables must be renamed* to ensure this

Basics of First-order Predicate Logic

Quantifiers and Normal Forms



First-order Predicate Logic

Hoàng Anh Đức

Introduction

Basics of First-order Predicate Logic

Syntax and Semantics

Variable Replacement and Substitution

11

Quantifiers and Normal Forms

Proof Calculi for Predicate Logic

Automated Theorem Provers

References

Negating Quantified Formulas

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

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

Basics of First-order Predicate Logic

Quantifiers and Normal Forms



Prenex Normal Form (PNF)

A formula is in *prenex normal form* if it is of the form

$$Q_1x_1Q_2x_2\ldots Q_nx_n\psi$$

where each Q_i is a quantifier (\forall or \exists), x_i are variables, and ψ is a quantifier-free formula

Theorem 1

For every formula φ there is an equivalent formula φ' in prenex normal form

Example 2

- Let $\varphi = \forall x (P(x) \rightarrow \exists y Q(x, y))$.
- φ is not in prenex normal form **[Why?]**
- An equivalent prenex form: $\varphi' = \forall x \exists y (P(x) \rightarrow Q(x, y))$.
[Verify!]

First-order Predicate Logic

Hoàng Anh Đức

Introduction

Basics of First-order Predicate Logic

Syntax and Semantics

Variable Replacement and Substitution

12

Quantifiers and Normal Forms

Proof Calculi for Predicate Logic

Automated Theorem Provers

References

Basics of First-order Predicate Logic

Quantifiers and Normal Forms



First-order Predicate Logic

Hoàng Anh Đức

Introduction

Basics of First-order Predicate Logic

Syntax and Semantics

Variable Replacement and Substitution

13

Quantifiers and Normal Forms

Proof Calculi for Predicate Logic

Automated Theorem Provers

References

Exercise 5

Review the following algorithms:

- (a) Transforming a formula into an equivalent formula in Prenex Normal Form (PNF)
- (b) Skolemization:
 - Eliminate existential quantifiers by introducing Skolem functions
 - If the original formula is true, then the resulting formula is also true. The converse is not always true

To have a better understanding, try to apply these algorithms to some predicate-logic formulas.

Basics of First-order Predicate Logic

Proof Calculi for Predicate Logic



First-order Predicate
Logic

Hoàng Anh Đức

Introduction

Basics of First-order
Predicate Logic

Syntax and Semantics

Variable Replacement and
Substitution

Quantifiers and Normal
Forms

14

Proof Calculi for Predicate
Logic

Automated Theorem
Provers

References

- Our main goal is to introduce the *resolution calculus* for predicate logic
- We shall first begin with a simpler calculus formed by two inference rules: Modus Ponens and \forall -Elimination

■ *Modus Ponens (MP)*

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

■ *\forall -Elimination (\forall -E)*

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

Note: t is a ground term that
contains no variables

Basics of First-order Predicate Logic

Proof Calculi for Predicate Logic



First-order Predicate Logic

Hoàng Anh Đức

Introduction

Basics of First-order Predicate Logic

Syntax and Semantics

Variable Replacement and Substitution

Quantifiers and Normal Forms

15

Proof Calculi for Predicate Logic

Automated Theorem Provers

References

KB

?

Q

$female(karen) \quad female(anne) \quad female(mary)$

$female(eve) \quad female(isabelle)$

$child(oscar, karen, frank) \quad child(mary, karen, frank)$

$child(eve, anne, oscar) \quad child(henry, anne, oscar)$

$child(isabelle, anne, oscar) \quad child(clyde, mary, oscarb)$

$\forall x \forall y \forall z \ child(x, y, z) \Rightarrow child(x, z, y)$

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

$child(eve, oscar, anne)$

Basics of First-order Predicate Logic

Proof Calculi for Predicate Logic



First-order Predicate
Logic

Hoàng Anh Đức

Introduction

Basics of First-order
Predicate Logic

Syntax and Semantics

Variable Replacement and
Substitution

Quantifiers and Normal
Forms

15

Proof Calculi for Predicate
Logic

Automated Theorem
Provers

References

KB

?

Q

$female(karen) \quad female(anne) \quad female(mary)$

$female(eve) \quad female(isabelle)$

$child(oscar, karen, frank) \quad child(mary, karen, frank)$

$child(eve, anne, oscar) \quad child(henry, anne, oscar)$

$child(isabelle, anne, oscar) \quad child(clyde, mary, oscarb)$

$\forall x \forall y \forall z \ child(x, y, z) \Rightarrow child(x, z, y)$

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

$child(eve, oscar, anne)$

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. $child(eve, anne, oscar) \Rightarrow child(eve, oscar, anne)$

\forall -E for 2: $x/eve, y/anne, z/oscar$

4. $child(eve, oscar, anne)$

MP for 1 and 3

Basics of First-order Predicate Logic

Proof Calculi for Predicate Logic



First-order Predicate Logic

Hoàng Anh Đức

Introduction

Basics of First-order Predicate Logic

Syntax and Semantics

Variable Replacement and Substitution

Quantifiers and Normal Forms

16

Proof Calculi for Predicate Logic

Automated Theorem Provers

References

Exercise 6

In the textbook, the author claimed in page 52 that “The calculus consisting of the two given inference rules is not complete.” where here “the two given inference rules” are Modus Ponens and \forall -Elimination.

To prove that the claim is indeed correct, give an example of a query Q that cannot be derived from the above example KB using only these two inference rules. (That is, $KB \models Q$ but $KB \not\vdash Q$ using only Modus Ponens and \forall -Elimination) (**Hint:** Think about what appears in the knowledge base but not in the inference rules)

Basics of First-order Predicate Logic

Proof Calculi for Predicate Logic



First-order Predicate Logic

Hoàng Anh Đức

Introduction

Basics of First-order Predicate Logic

Syntax and Semantics

Variable Replacement and Substitution

Quantifiers and Normal Forms

17 Proof Calculi for Predicate Logic

Automated Theorem Provers

References

Theorem 2 (Gödel's completeness theorem [Godel 1931])

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 $KB \vdash \varphi$.

Note

Indeed, as we will see later, there is a calculus (e.g., resolution calculus) with which only true propositions can be proved. That is, if $KB \vdash \varphi$ holds, then $KB \models \varphi$.

Basics of First-order Predicate Logic

Proof Calculi for Predicate Logic



Triggered by the resolution calculus (introduced in 1965), around 1970s, many scientists believed that one could formulate almost every task of knowledge representation and reasoning in PL1 and then solve it with an automated prover.

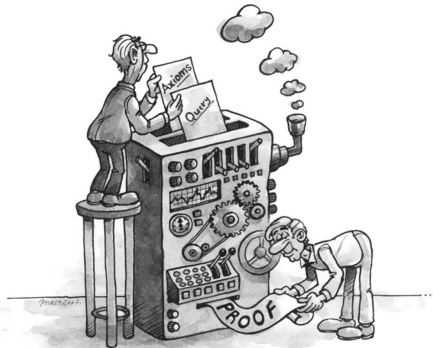


Figure: The Universal Logic Machine (from [Ertel 2025], Figure 3.3, p. 53)

First-order Predicate Logic

Hoàng Anh Đức

Introduction

Basics of First-order Predicate Logic

Syntax and Semantics

Variable Replacement and Substitution

Quantifiers and Normal Forms

18

Proof Calculi for Predicate Logic

Automated Theorem Provers

References

Basics of First-order Predicate Logic

Proof Calculi for Predicate Logic



First-order Predicate
Logic

Hoàng Anh Đức

Introduction

Basics of First-order
Predicate Logic

Syntax and Semantics

Variable Replacement and
Substitution

Quantifiers and Normal
Forms

19

Proof Calculi for Predicate
Logic

Automated Theorem
Provers

References

Recall: Resolution Rule in Propositional Logic

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

Recall: Resolution Proof in Propositional Logic

- **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 $KB \not\models Q$

Basics of First-order Predicate Logic

Proof Calculi for Predicate Logic



First-order Predicate
Logic

Hoàng Anh Đức

Introduction

Basics of First-order
Predicate Logic

Syntax and Semantics

Variable Replacement and
Substitution

Quantifiers and Normal
Forms

Proof Calculi for Predicate
Logic

Automated Theorem
Provers

References

Generalized Resolution Rule for PL1

The resolution rule for two clauses in CNF 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 **[What is σ ? How to find it?]**
- 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

20

42

Basics of First-order Predicate Logic

Proof Calculi for Predicate Logic



First-order Predicate
Logic

Hoàng Anh Đức

Introduction

Basics of First-order
Predicate Logic

Syntax and Semantics

Variable Replacement and
Substitution

Quantifiers and Normal
Forms

21

Proof Calculi for Predicate
Logic

Automated Theorem
Provers

References

Exercise 7

Explain how to obtain the resolution rule in propositional logic from the generalized resolution rule for PL1

Basics of First-order Predicate Logic

Proof Calculi for Predicate Logic



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

Factorization Rule for PL1

Factorization of a clause is accomplished by

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

where σ is the MGU of A_1 and A_2

What does this definition mean?

- Premise: $A_1 \vee A_2 \vee \dots \vee A_n$
- $\sigma(A_1) = \sigma(A_2)$ 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 $\sigma(A_1) \vee \sigma(A_2) \vee \dots \vee \sigma(A_n)$
- As $p \vee p \equiv p$, the Conclusion $\sigma(A_2) \vee \dots \vee \sigma(A_n)$ is derived
- Intuitively, after σ is applied, as $\sigma(A_1) = \sigma(A_2)$, one of these literals becomes “redundant” and can be “removed”

First-order Predicate Logic

Hoàng Anh Đức

Introduction

Basics of First-order Predicate Logic

Syntax and Semantics

Variable Replacement and Substitution

Quantifiers and Normal Forms

22

Proof Calculi for Predicate Logic

Automated Theorem Provers

References

Basics of First-order Predicate Logic

Proof Calculi for Predicate Logic



First-order Predicate
Logic

Hoàng Anh Đức

Introduction

Basics of First-order
Predicate Logic

Syntax and Semantics

Variable Replacement and
Substitution

Quantifiers and Normal
Forms

23

Proof Calculi for Predicate
Logic

Automated Theorem
Provers

References

KB

?

Q

$female(karen) \quad female(anne) \quad female(mary)$

$female(eve) \quad female(isabelle)$

$child(oscar, karen, frank) \quad child(mary, karen, frank)$

$child(eve, anne, oscar) \quad child(henry, anne, oscar)$

$child(isabelle, anne, oscar) \quad child(clyde, mary, oscarb)$

$\forall x \forall y \forall z \ child(x, y, z) \Rightarrow child(x, z, y)$

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

$child(eve, oscar, anne)$

$\sigma : x/eve, y/anne, z/oscar$

$\neg B' \quad \neg child(x, y, z) \vee child(x, z, y)$

\equiv

$B \quad child(eve, anne, oscar), child(x, y, z) \Rightarrow child(x, z, y)$

$\sigma(B) = \sigma(B')$

Gen. Res.

$child(eve, oscar, anne)$

$\sigma(child(x, z, y))$

Basics of First-order Predicate Logic

Proof Calculi for Predicate Logic



First-order Predicate Logic

Hoàng Anh Đức

Introduction

Basics of First-order Predicate Logic

Syntax and Semantics

Variable Replacement and Substitution

Quantifiers and Normal Forms

24

Proof Calculi for Predicate Logic

Automated Theorem Provers

References

With the above generalized rules, a resolution proof for the family example would be:

Step

Proved by

- | | |
|---|--------------|
| 1. <i>child(eve, anne, oscar)</i> | <i>KB</i> |
| 2. $\neg \textit{child}(x, y, z) \vee \textit{child}(x, z, y)$ | <i>KB</i> |
| 3. $\neg \textit{child}(\textit{eve}, \textit{oscar}, \textit{anne})$ | $\neg Q$ |
| 4. <i>child(eve, oscar, anne)</i> | GenRes(1, 2) |
| 5. () | GenRes(3, 4) |

Basics of First-order Predicate Logic

Proof Calculi for Predicate Logic



First-order Predicate
Logic

Hoàng Anh Đức

Introduction

Basics of First-order
Predicate Logic

Syntax and Semantics

Variable Replacement and
Substitution

Quantifiers and Normal
Forms

25

Proof Calculi for Predicate
Logic

Automated Theorem
Provers

References

Exercise 8

Write a resolution proof for the query $Q = \exists y \text{ knows}(\text{henry}, y)$ with respect to the knowledge base

$KB = \forall x \text{ knows}(x, \text{mother}(x))$, where $\text{mother}(x)$ is a function that returns the mother of x , $\text{knows}(x, y)$ means “ x knows y ”, and henry is a constant representing a person named Henry.

Exercise 9

Use the resolution calculus to prove that the formula

$R = \forall x (\neg \text{shaves}(\text{barber}, x) \vee \neg \text{shaves}(x, x)) \wedge (\text{shaves}(x, x) \vee \text{shaves}(\text{barber}, x))$ (a.k.a the Russell's paradox) is unsatisfiable, where $\text{shaves}(x, y)$ means “ x shaves y ” and barber is a constant representing a barber. (**Hint:** Remember that you can also use the Factorization rule along with the Generalized Resolution rule.)

Basics of First-order Predicate Logic

Proof Calculi for Predicate Logic



First-order Predicate
Logic

Hoàng Anh Đức

Introduction

Basics of First-order
Predicate Logic

Syntax and Semantics

Variable Replacement and
Substitution

Quantifiers and Normal
Forms

26

Proof Calculi for Predicate
Logic

Automated Theorem
Provers

References

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

Exercise 10

Do your own research on some *strategies* that can be used to carry out a resolution proof more efficiently.

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

Basics of First-order Predicate Logic

Proof Calculi for Predicate Logic



First-order Predicate Logic

Hoàng Anh Đức

Introduction

Basics of First-order Predicate Logic

Syntax and Semantics

Variable Replacement and Substitution

Quantifiers and Normal Forms

27

Proof Calculi for Predicate Logic

Automated Theorem Provers

References

Exercise 11

Equality is a particularly problematic source of search-space explosion. When the equality axioms are included in the knowledge base, they can generate new clauses and equations that in turn trigger further applications of the equality axioms, potentially causing unbounded growth.

Because of this, special inference rules for equality like *demodulation* and more generally *paramodulation* have been developed which get by without explicit equality axioms and, in particular, reduce the search space. Do your own research on these two inference rules and explain how they work.

Automated Theorem Provers



First-order Predicate
Logic

Hoàng Anh Đức

Introduction

Basics of First-order
Predicate Logic

Syntax and Semantics

Variable Replacement and
Substitution

Quantifiers and Normal
Forms

Proof Calculi for Predicate
Logic

28

Automated Theorem
Provers

References

- An *automated theorem prover* is a software tool that uses logical reasoning to automatically prove or disprove mathematical theorems or logical statements
- It takes a set of axioms and a conjecture as input and attempts to derive the conjecture from the axioms using formal logic rules
- Automated theorem provers are used in various fields, including mathematics, computer science, and artificial intelligence, to assist in formal verification, program analysis, and knowledge representation
- We demonstrate how to use an automated theorem prover called *E* to solve some problems in first-order predicate logic

Automated Theorem Provers



First-order Predicate
Logic

Hoàng Anh Đức

Introduction

Basics of First-order
Predicate Logic

Syntax and Semantics

Variable Replacement and
Substitution

Quantifiers and Normal
Forms

Proof Calculi for Predicate
Logic

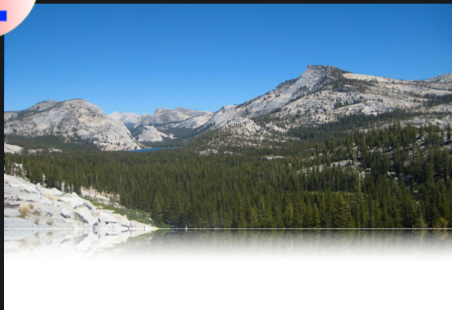
29 Automated Theorem
Provers

References



The E Theorem Prover

E
Technology
Download
Archive
Usage
FAQ
Awards
References
Me
Impressum



Overview

E is a theorem prover for full first-order logic (and now monomorphic higher-order logic) with equality. It accepts a problem specification, typically consisting of a number of clauses or formulas, and a conjecture, again either in clausal or full first-order form. The system will then try to find a formal proof for the conjecture, assuming the axioms.

Figure: E Theorem Prover

<https://www.lehre.dhbw-stuttgart.de/~sschulz/E/E.html>



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 3

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

Goal

Prove Theorem 3

Automated Theorem Provers

Proof by Intuitive Mathematical Reasoning



First-order Predicate
Logic

Hoàng Anh Đức

Introduction

Basics of First-order
Predicate Logic

Syntax and Semantics

Variable Replacement and
Substitution

Quantifiers and Normal
Forms

Proof Calculi for Predicate
Logic

31 Automated Theorem
Provers

References

Proof of Theorem 3.

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 $e_l = e_l \cdot e_r = e_r$. □

Automated Theorem Provers

Proof by Resolution Calculus



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)

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

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

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

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

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

First-order Predicate
Logic

Hoàng Anh Đức

Introduction

Basics of First-order
Predicate Logic

Syntax and Semantics

Variable Replacement and
Substitution

Quantifiers and Normal
Forms

Proof Calculi for Predicate
Logic

32 Automated Theorem
Provers

References

Automated Theorem Provers

Proof by Resolution Calculus



First-order Predicate
Logic

Hoàng Anh Đức

Introduction

Basics of First-order
Predicate Logic

Syntax and Semantics

Variable Replacement and
Substitution

Quantifiers and Normal
Forms

Proof Calculi for Predicate
Logic

33 Automated Theorem
Provers

References

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. $()$	$\text{Res}(1, 12, \emptyset)$

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.

Automated Theorem Provers

Proof By E Theorem Prover



Transformation in clause normal form language LOP (The syntax of LOP represents an extension of the PROLOG syntax for non Horn clauses.)

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

An input file for E

halbgr1.lop

```
1  <- eq(el,er). % query
2  eq( m(m(X,Y),Z), m(X,m(Y,Z)) ). % associativity of m
3  eq( m(el,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
```

First-order Predicate
Logic

Hoàng Anh Đức

Introduction

Basics of First-order
Predicate Logic

Syntax and Semantics

Variable Replacement and
Substitution

Quantifiers and Normal
Forms

Proof Calculi for Predicate
Logic

34 Automated Theorem
Provers

References

Automated Theorem Provers

Proof By E Theorem Prover



```
Run eprover --proof-object halbgr1.lop | epclextract
```

```
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'
```

Note:

- Due to the difference between versions of E, the output may be different from the above figure.
- Positive literals are identified by ++ and negative literals by --.
- $\text{pm}(a, b)$ stands for a resolution step between clause a and clause b . (pm means Paramodulation.)

First-order Predicate
Logic

Hoàng Anh Đức

Introduction

Basics of First-order
Predicate Logic

Syntax and Semantics

Variable Replacement and
Substitution

Quantifiers and Normal
Forms

Proof Calculi for Predicate
Logic

35

Automated Theorem
Provers

References

Automated Theorem Provers

Proof By E Theorem Prover



We'll walk through the prover's steps, focusing on intuition.

Starting Rules (Lines 0–4)

- **Line 0 (Symmetry):** If $x_2 = x_1$, then $x_1 = x_2$.

$$eq(x_1, x_2) \vee \neg eq(x_2, x_1)$$

- **Line 1 (Transitivity):** If $x_1 = x_3$ and $x_3 = x_2$, then $x_1 = x_2$.

$$eq(x_1, x_2) \vee \neg eq(x_1, x_3) \vee \neg eq(x_3, x_2)$$

- **Line 2 (Left Identity):** For any x , $e_l \cdot x = x$.

$$eq(m(e_l, x_1), x_1)$$

- **Line 3 (Negated Conjecture):** Assume $e_l \neq e_r$.

$$\neg eq(e_l, e_r)$$

- **Line 4 (Right Identity):** For any x , $x \cdot e_r = x$.

$$eq(m(x_1, e_r), x_1)$$

First-order Predicate
Logic

Hoàng Anh Đức

Introduction

Basics of First-order
Predicate Logic

Syntax and Semantics

Variable Replacement and
Substitution

Quantifiers and Normal
Forms

Proof Calculi for Predicate
Logic

36 Automated Theorem
Provers

References

Automated Theorem Provers

Proof By E Theorem Prover



First-order Predicate
Logic

Hoàng Anh Đức

Introduction

Basics of First-order
Predicate Logic

Syntax and Semantics

Variable Replacement and
Substitution

Quantifiers and Normal
Forms

Proof Calculi for Predicate
Logic

37 Automated Theorem
Provers

References

Simplifying the Rules (Lines 5–10)

- **Lines 5–6:** The prover rewrites Lines 0 and 1 into a standard form for processing. Think of it as organizing the rules.
- **Lines 7–10:** Converts axioms into a logical format:
 - Line 8: $[+ + eq(m(e_l, x_1), x_1)]$ — Restates $e_l \cdot x = x$.
 - Line 9: $\neg eq(e_l, e_r)$ — Restates $e_l \neq e_r$.
 - Line 10: Transitivity in logical form.

First Big Step (Line 11)

- The prover uses symmetry (Line 7) and left identity (Line 8) to derive:

$$[+ + eq(x_1, m(e_l, x_1))]$$

- **Intuition:** If $e_l \cdot x = x$, then $x = e_l \cdot x$. Like saying: “If $1 \cdot x = x$, then $x = 1 \cdot x$.” A small logical flip to rewrite equations.

Automated Theorem Provers

Proof By E Theorem Prover



First-order Predicate
Logic

Hoàng Anh Đức

Introduction

Basics of First-order
Predicate Logic

Syntax and Semantics

Variable Replacement and
Substitution

Quantifiers and Normal
Forms

Proof Calculi for Predicate
Logic

38 Automated Theorem
Provers

References

Combining Transitivity (Line 13)

- Combines transitivity (Line 10) with Line 11 to derive:

$$[+ + eq(x_1, x_2), - - eq(m(e_l, x_1), x_2)]$$

- **Intuition:** Either $x = y$ or $e_l \cdot x \neq y$. If $e_l \cdot x$ isn't equal to y , then x can't be y . Sets up a rule for testing equalities.

Automated Theorem Provers

Proof By E Theorem Prover



First-order Predicate
Logic

Hoàng Anh Đức

Introduction

Basics of First-order
Predicate Logic

Syntax and Semantics

Variable Replacement and
Substitution

Quantifiers and Normal
Forms

Proof Calculi for Predicate
Logic

39 Automated Theorem
Provers

References

Using the Right Identity (Line 14 and Line 16)

■ **Line 14:** $[+ + eq(m(x_1, e_r), x_1)]$ — Restates $x \cdot e_r = x$.

■ In Line 16, combines Line 13 with Line 14:

$$eq(x_1, m(x_1, e_r)) \vee \neg eq(m(e_l, x_1), m(x_1, e_r))$$

■ Since $m(x_1, e_r) = x_1$, simplifies to:

$$eq(x_1, x_1) \vee \neg eq(m(e_l, x_1), x_1)$$

■ Since $x_1 = x_1$ is true, this becomes:

$$\neg eq(m(e_l, x_1), x_1)$$

■ **Problem:** This says $e_l \cdot x \neq x$, contradicting Line 8 ($e_l \cdot x = x$). Like saying $1 \cdot x \neq x$, which can't be true.

Automated Theorem Provers

Proof By E Theorem Prover



First-order Predicate
Logic

Hoàng Anh Đức

Introduction

Basics of First-order
Predicate Logic

Syntax and Semantics

Variable Replacement and
Substitution

Quantifiers and Normal
Forms

Proof Calculi for Predicate
Logic

40 Automated Theorem
Provers

References

Tying It to the Goal (Line 15 and Line 16)

- **Line 15:** $[- - eq(e_r, e_l)]$ — Says $e_r \neq e_l$, from $e_l \neq e_r$ via symmetry.
- In Line 16, set $x_1 = e_r$ in $\neg eq(m(e_l, x_1), x_1)$:

$$\neg eq(m(e_l, e_r), e_r)$$

- **Why it matters:** Line 8 says $e_l \cdot e_r = e_r$, so $e_l \cdot e_r \neq e_r$ is false. If $e_l = e_r$, then $e_r = e_l$, contradicting $e_r \neq e_l$.
- **Result:** Empty clause (\square), meaning “this math doesn’t add up.” Thus, $e_l \neq e_r$ is impossible, so $e_l = e_r$.

Automated Theorem Provers

Proof By E Theorem Prover



First-order Predicate
Logic

Hoàng Anh Đức

Introduction

Basics of First-order
Predicate Logic

Syntax and Semantics

Variable Replacement and
Substitution

Quantifiers and Normal
Forms

Proof Calculi for Predicate
Logic

41 Automated Theorem
Provers

References

Exercise 12

Use the E theorem prover to solve [Ertel 2025], Exercise 3.9, p. 64. Prepare the LOP input file and run E (for example: `eprover -proof-object <file> | epclextract`). Collect the prover's output and ask an LLM (like ChatGPT) to translate the machine proof into concise, human-readable mathematical reasoning. What do you think about the output of the LLM?

References



First-order Predicate Logic

Hoàng Anh Đức

Introduction

Basics of First-order Predicate Logic

Syntax and Semantics

Variable Replacement and Substitution

Quantifiers and Normal Forms

Proof Calculi for Predicate Logic

Automated Theorem Provers

References



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

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



Gödel, Kurt (1931). "Diskussion zur Grundlegung der Mathematik: Erkenntnis 2." In: *Monatshefte für Mathematik Und Physik* 32, pp. 147–148.