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

Introduction

Basics of First-order 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

Introduction

- *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*.

2 Introduction

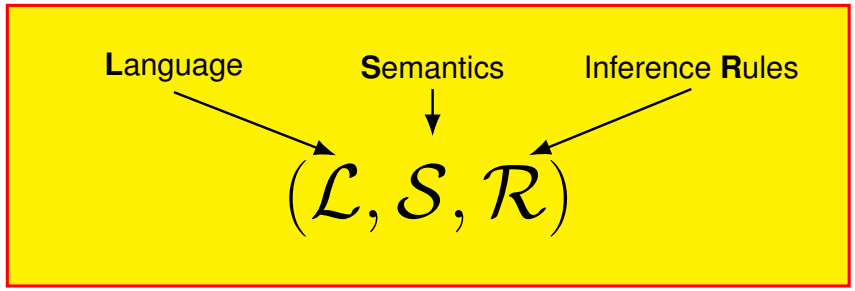
Basics of First-order Predicate Logic

Automated Theorem Provers

References

Basics of First-order Predicate Logic

Syntax and Semantics



Predicate Logic Formulas (Syntax)

- **Terms**
  - Simple Terms
    - Variables V
    - Constants K
  - Complex Terms
    - Function Symbols F: $f(t_1, t_2, \ldots, t_n)$
- **Logical Operators** Op: $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow, (,)$
- **Predicate Symbols** P: $p(t_1, t_2, \ldots, t_n)$
- **Quantifiers**: $\forall, \exists$

Predicate Logic Formulas (Semantics)

Interpretation $\mathbb{I}$ (mapping to real-world)

Variables V, Constants K	→ Names of Objects
Function Symbols F	→ Functions
Predicate Symbols P	→ Relations

Predicate Logic Formulas (Premise) —Derivation (Applying inference rules)→ Predicate Logic Formula (Conclusion)

Introduction

Basics of First-order Predicate Logic

3 Syntax and Semantics

Variable Replacement and Substitution

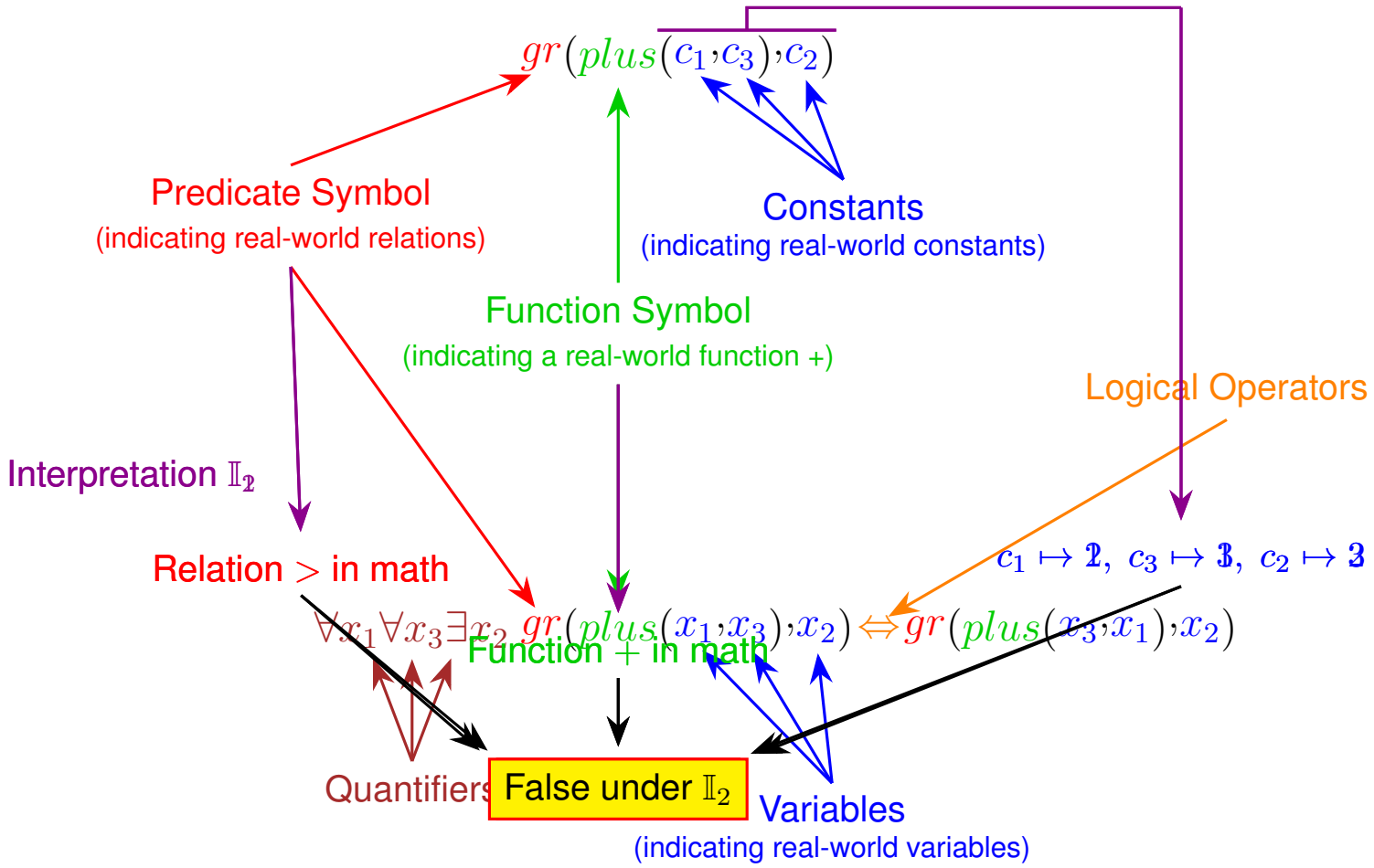
Quantifiers and Normal Forms

Proof Calculi for Predicate Logic

Automated Theorem Provers

References

Examples of Predicate Logic Formulas (SyntaxSemantics)



Basics of First-order Predicate Logic
Syntax and Semantics

We 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

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

Basics of First-order Predicate Logic
Syntax and Semantics

Predicate logic has four kinds of symbols: *variables* ($x, y, \ldots$), *constants* (names of specific objects), *function symbols* (stand for *functions*), and *predicate symbols* (stand for *relations*).

Quiz

In *gr*(*plus*$(c_1, c_3), c_2)$, which classification is correct?

(a) *gr*, *plus* are function symbols; c_1, c_2, c_3 constants

(b) *gr* is a predicate symbol, *plus* a function symbol, c_1, c_2, c_3 constants

(c) *gr*, *plus* are predicate symbols; c_1, c_2, c_3 variables

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

Function symbols *"stand for functions"* and predicate symbols *"stand for relations"*: a *function symbol returns an object*, a *predicate symbol returns true/false*.

Quiz

What do *plus*(c_1, c_3) and *gr*(c_1, c_3) each *return*?

(a) both return an object

(b) *plus*(c_1, c_3) returns an *object*; *gr*(c_1, c_3) returns *true/false*

(c) both return true/false

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

A *term* is an expression that *refers to an object*. The *atomic (simple) terms* are the *variables* and *constants*; a *complex term* is a function symbol applied to terms, $f(t_1, \ldots, t_n)$. A complex term is *just a complicated kind of name* for an object.

Quiz

Which of these is a *complex term*?

(a) c_2 (b) *plus*(c_1, c_3) (c) *gr*(*plus*$(c_1, c_3), c_2)$

A symbol is pure *syntax*: *"a complex term is just a complicated kind of name"*. Only an *interpretation* $\mathbb{I}$ gives it a real function or relation.

Quiz

Is the *function symbol plus* the same thing as the mathematical function $+$?

(a) Yes — they are identical

(b) No — $\mathbb{I}$ *assigns plus* a function (e.g. $\mapsto +$)

(c) No — *plus* is really a predicate symbol

From terms we build *formulas*. An *atomic formula* is a predicate symbol applied to terms, $p(t_1, \ldots, t_n)$. Larger formulas use the connectives $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$ and the *quantifiers* $\forall$ (for all) and $\exists$ (there exists). *A term names an object; a formula is true or false*.

Quiz

Which of these is a *well-formed formula*?

(a) $\mathit{plus}(c_1, c_3)$ (b) $\forall x_1\, \mathit{gr}(\mathit{plus}(x_1, x_3), x_2)$ (c) $\mathit{gr}(\forall x_1, c_2)$

First-order Predicate Logic

Hoàng Anh Đức

Introduction

Basics of First-order Predicate Logic

11 Syntax and Semantics

Variable Replacement and Substitution

Quantifiers and Normal Forms

Proof Calculi for Predicate Logic

Automated Theorem Provers

References

A *literal* is an *atomic formula or its negation*: $p(t_1, \ldots, t_n)$ (a *positive* literal) or $\neg p(t_1, \ldots, t_n)$ (a *negative* literal). Nothing else — no other connectives, no quantifiers.

Quiz

Which of these is a *literal*?

(a) $gr(c_1, c_2) \wedge gr(c_2, c_3)$ (b) $\neg gr(c_1, c_2)$ (c) $\forall x\, gr(x, c_2)$

A variable occurrence is *bound* if it lies in the *scope of a quantifier* $\forall x$ or $\exists x$; otherwise it is *free*. A formula with *no free variables* is a *closed formula* (a *first-order sentence*).

Quiz

In $\forall x_1\ gr(plus(x_1, x_3), x_2)$, which variables are *free*?

(a) x_1, x_3, x_2 (b) x_3, x_2 (c) none

As in propositional logic, a formula is in *conjunctive normal form (CNF)* when it is an *and of clauses*, where each *clause* is an *or of literals*. Note that these carry over to predicate logic *analogously* — the literals are now predicate literals.

Quiz

Which formula is in *CNF*?

(a) $\big(gr(x,c) \vee \neg p(x)\big) \wedge \big(p(x) \vee q(x)\big)$

(b) $p(x) \vee \big(q(x) \wedge r(x)\big)$

(c) $\neg\big(p(x) \wedge q(x)\big)$

Introduction

Basics of First-order Predicate Logic

13 Syntax and Semantics

Variable Replacement and Substitution

Quantifiers and Normal Forms

Proof Calculi for Predicate Logic

Automated Theorem Provers

References

A *Horn clause* is a clause of the form $\neg A_1 \vee \cdots \vee \neg A_m \vee B_1 \vee \cdots \vee B_n$, where $m, n \geq 0$, with *at most one positive literal* ($n \leq 1$). Thus its possible equivalent implication forms are $A_1 \wedge \cdots \wedge A_m \Rightarrow B$, $A_1 \wedge \cdots \wedge A_m \Rightarrow f$, and B. A *Horn formula* is a conjunction of Horn clauses; Horn formulas are important for rule-based reasoning.

Quiz

Which clause is a *Horn clause*?

(a) $\neg p(x) \vee \neg q(x) \vee r(x)$

(b) $p(x) \vee q(x) \vee \neg r(x)$

(c) $p(x) \vee q(x)$

First-order Predicate Logic

Hoàng Anh Đức

Introduction

Basics of First-order Predicate Logic

14 Syntax and Semantics

Variable Replacement and Substitution

Quantifiers and Normal Forms

Proof Calculi for Predicate Logic

Automated Theorem Provers

References

First-order Predicate Logic

Hoàng Anh Đức

Introduction

Basics of First-order Predicate Logic

15 Syntax and Semantics

Variable Replacement and Substitution

Quantifiers and Normal Forms

Proof Calculi for Predicate Logic

Automated Theorem Provers

References

An *interpretation* (*assignment*) $\mathbb{I}$ gives meaning to the symbols: it maps each *constant/variable* to an *object*, each *function symbol* to a *function*, and each *predicate symbol* to a *relation*.
As an example, choose $\mathbb{I}_1$: domain $\mathbb{N}$, *plus* $\mapsto +$, *gr* $\mapsto >$, $c_1 \mapsto 1, c_3 \mapsto 3, c_2 \mapsto 2$.

Quiz

Under $\mathbb{I}_1$, what does the *term* $plus(c_1, c_3)$ denote?

(a) *true* (b) 4 (c) 3

An *atomic formula* $p(t_1, \ldots, t_n)$ is *true under* $\mathbb{I}$ exactly when the interpreted arguments stand in the interpreted relation: $\big(\mathbb{I}(t_1), \ldots, \mathbb{I}(t_n)\big) \in \mathbb{I}(p)$. Connectives and quantifiers then behave as in propositional logic.

Quiz

Is $\varphi \equiv \mathit{gr}(\mathit{plus}(c_1, c_3), c_2)$ *true* under $\mathbb{I}_1$?

(a) True (b) False

An interpretation that makes F true is a *model* of F. Then F is *satisfiable* if it has *at least one* model, *valid* (a *tautology*) if it is true under *every* interpretation, and *unsatisfiable* if it has *no* model. These carry over unchanged from propositional logic.

Quiz

$\varphi \equiv gr(plus(c_1, c_3), c_2)$ is true under $\mathbb{I}_1$ but false under $\mathbb{I}_2$. So φ is:

(a) valid (b) satisfiable but not valid (c) unsatisfiable

$F \equiv G$ (*semantically equivalent*): the *same truth value under every interpretation*. $KB \models Q$ (*semantic entailment*): *every model of KB is also a model of Q*, i.e. $Mod(KB) \subseteq Mod(Q)$.

Quiz

What does $F \models G$ mean?

(a) F and G have exactly the same models

(b) every model of F is also a model of G

(c) F and G are both valid

Introduction

Basics of First-order Predicate Logic

18 Syntax and Semantics

Variable Replacement and Substitution

Quantifiers and Normal Forms

Proof Calculi for Predicate Logic

Automated Theorem Provers

References

Putting it together on Ertel's family-tree example [Ertel 2025], Example 3.2, p. 45: *child* is a 3-place *predicate symbol*, *eve*, *anne*, *oscar* are *constants*, and $\mathbb{I}(\mathit{child}(x, y, z)) = \mathit{true}$ iff $(\mathbb{I}(x), \mathbb{I}(y), \mathbb{I}(z)) \in \mathit{Child}$.

Quiz

Take $\mathbb{I}(\mathit{oscar}) =$ Oscar A., $\mathbb{I}(\mathit{eve}) =$ Eve A., $\mathbb{I}(\mathit{anne}) =$ Anne A., with $(\text{Eve A.}, \text{Anne A.}, \text{Oscar A.}) \in \mathit{Child}$. Is $\mathit{child}(\mathit{eve}, \mathit{anne}, \mathit{oscar})$ *true* under $\mathbb{I}$?

(a) True (b) False

Exercise 1

Use the family-tree vocabulary from Ertel's Example 3.2 [Ertel 2025], p. 45: *female*(x) means that x is female, and *child*(x, y, z) means that x is a child of parents y and z.

$$\begin{aligned} \mathit{Female} &= \{\mathit{karen}, \mathit{anne}, \mathit{mary}, \mathit{eve}, \mathit{isabelle}\}, \\ \mathit{Child} &= \{(\mathit{oscar}, \mathit{karen}, \mathit{frank}), (\mathit{mary}, \mathit{karen}, \mathit{frank}), \\ &\quad (\mathit{eve}, \mathit{anne}, \mathit{oscar}), (\mathit{henry}, \mathit{anne}, \mathit{oscar}), \\ &\quad (\mathit{isabelle}, \mathit{anne}, \mathit{oscar}), (\mathit{clyde}, \mathit{mary}, \mathit{oscarb})\}. \end{aligned}$$

(a) For each predicate symbol, give one true example and one false example, using only the constants shown above.

(b) Using the facts above, what knowledge base KB does this family-tree example form? Give one query Q that we hope to derive from KB. The query Q should not be the query already considered in the example.

First-order Predicate Logic
Hoàng Anh Đức
Introduction
Basics of First-order Predicate Logic
20 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

- 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

$$\begin{array}{rcll} \forall x & x = x & & \text{(reflexivity)} \\ \forall x\, \forall y & x = y \Rightarrow y = x & & \text{(symmetry)} \\ \forall x\, \forall y\, \forall z & x = y \wedge y = z \Rightarrow x = z & & \text{(transitivity)} \end{array}$$

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

$$\begin{array}{lll} \forall x\, \forall y & x = y \Rightarrow f(x) = f(y) & \text{(substitution axiom)} \\ \forall x\, \forall y & x = y \Rightarrow p(x) \Leftrightarrow p(y) & \text{(substitution axiom)} \end{array}$$

Introduction

Basics of First-order Predicate Logic

21 Syntax and Semantics

Variable Replacement and Substitution

Quantifiers and Normal Forms

Proof Calculi for Predicate Logic

Automated Theorem Provers

References

First-order Predicate Logic

Hoàng Anh Đức

Introduction

Basics of First-order Predicate Logic

22 Syntax and Semantics

Variable Replacement and Substitution

Quantifiers and Normal Forms

Proof Calculi for Predicate Logic

Automated Theorem Provers

References

Equality is a predicate, but it is a special one: it compares *terms*, and its substitution axioms let us replace a term by an equal term inside function and predicate expressions.

Quiz

Which expression is a well-formed atomic formula using equality? Assume a, b are constants, f, g are function symbols, and p, q are predicate symbols.

(a) $f(a) = g(b)$

(b) $p(a) = q(b)$

(c) $f(a) = \forall x\, p(x)$

The first three equality axioms say that equality is an *equivalence relation*.

Quiz

Suppose we know $a = b$ and $b = c$. Which conclusion follows from the equality axioms alone?

(a) $a = c$

(b) $f(a) = c$

(c) $p(a)$

The function-symbol substitution axiom says that equal inputs give equal outputs.

Quiz

Suppose $a = b$. Which formula is justified by the substitution axiom for a function symbol f?

(a) $f(a) = f(b)$

(b) $f(a) = b$

(c) $p(a) \Leftrightarrow p(b)$

First-order Predicate Logic

Hoàng Anh Đức

Introduction

Basics of First-order Predicate Logic

25 Syntax and Semantics

Variable Replacement and Substitution

Quantifiers and Normal Forms

Proof Calculi for Predicate Logic

Automated Theorem Provers

References

The predicate-symbol substitution axiom says that replacing a term by an equal term preserves truth.

Quiz

Suppose $a = b$ and $p(a)$ is true. What can we conclude using the substitution axiom for a predicate symbol p?

(a) $p(b)$

(b) $p(a) = p(b)$

(c) $a = p(b)$

Other mathematical relations can also be described by axioms, but they do *not* inherit the equality axioms automatically.

Quiz

Can the strict order predicate $<$ be defined by exactly the same axioms as equality: reflexivity, symmetry, transitivity, and substitution?

(a) Yes; every binary predicate should satisfy the equality axioms.

(b) No; $<$ is not reflexive or symmetric, so it needs different axioms.

(c) No; predicates cannot be described by axioms.

First-order Predicate Logic
Hoàng Anh Đức
Introduction
Basics of First-order Predicate Logic
26 Syntax and Semantics
Variable Replacement and Substitution
Quantifiers and Normal Forms
Proof Calculi for Predicate Logic
Automated Theorem Provers
References

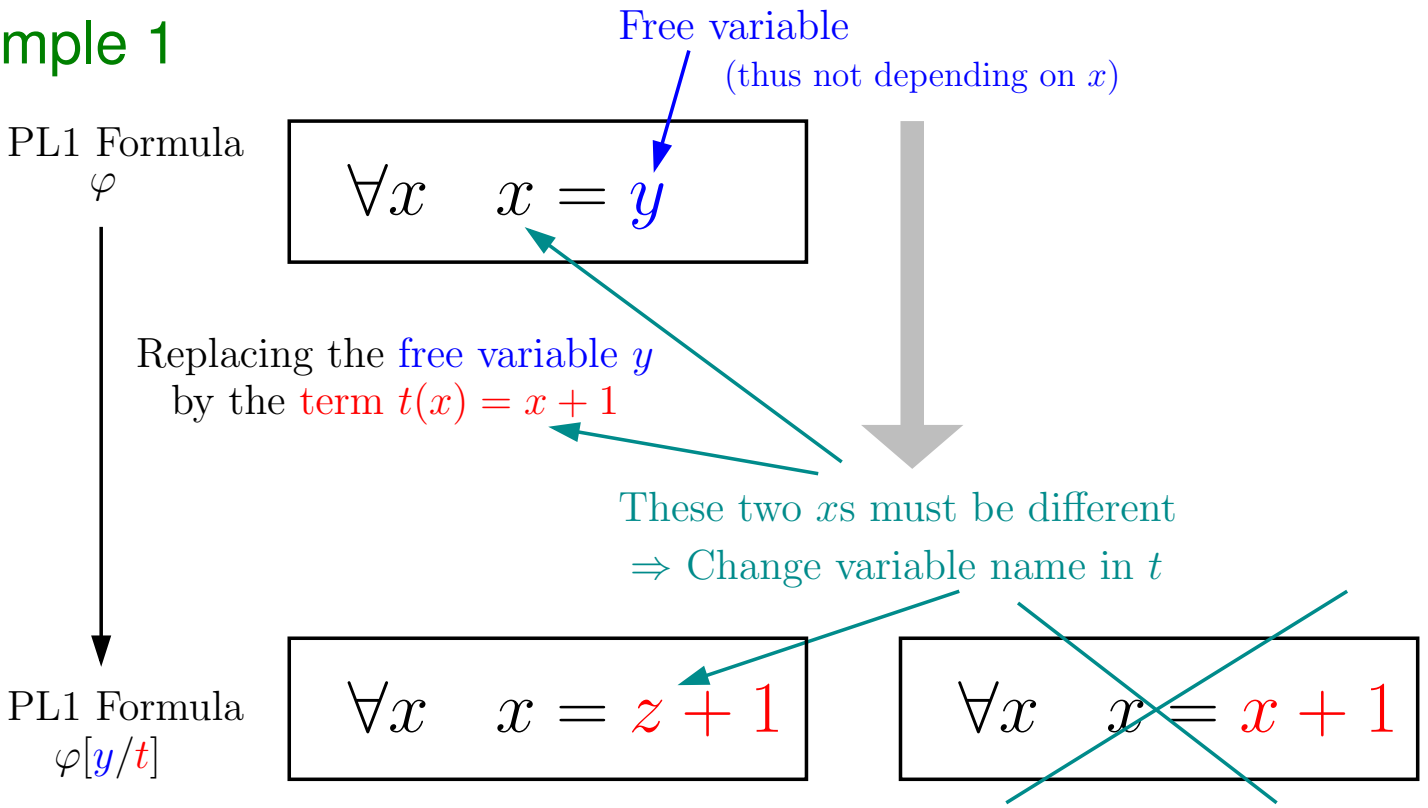
Basics of First-order Predicate Logic

Variable Replacement and Substitution

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



Basics of First-order Predicate Logic

Variable Replacement and Substitution

Introduction

Basics of First-order Predicate Logic

Syntax and Semantics

28 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
- $\sigma : x_1/t_1, x_2/t_2, \ldots, x_n/t_n$: a substitution that maps each variable x_i to the corresponding term t_i
- Also write $\sigma = \{x_1/t_1, x_2/t_2, \ldots, x_n/t_n\}$

- *Apply σ to a term t*, denoted by $\sigma(t)$ or $t[x_1/t_1, \ldots, x_n/t_n]$ to indicate $\sigma(t)$, means *simultaneously replacing every occurrence of each x_i in t by t_i*

First-order Predicate Logic

Hoàng Anh Đức

Introduction

Basics of First-order Predicate Logic

Syntax and Semantics

Variable Replacement and Substitution

29 Quantifiers and Normal Forms

Proof Calculi for Predicate Logic

Automated Theorem Provers

References

Negating Quantified Formulas

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

Exercise 2

Move each negation inward until it applies only to atomic formulas:

(a) $\neg\forall x\,\exists y\,R(x,y)$

(b) $\neg\exists x\,\forall y\,\mathit{likes}(y,x)$

(c) $\neg\forall x\,\forall y\,\forall z\,\big(\mathit{child}(x,y,z) \Rightarrow \mathit{child}(x,z,y)\big)$

First-order Predicate Logic
Hoàng Anh Đức
Introduction
Basics of First-order Predicate Logic
Syntax and Semantics
Variable Replacement and Substitution
30 Quantifiers and Normal Forms
Proof Calculi for Predicate Logic
Automated Theorem Provers
References

Prenex Normal Form (PNF)

A formula is in *prenex normal form* if it is of the form $Q_1 x_1 Q_2 x_2 \ldots Q_n x_n \psi$, 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!]**

Exercise 3

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
- The resulting formula is no longer equivalent to the original one, but its satisfiability remains unchanged: it is satisfiable if and only if the original formula is

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

Introduction
Basics of First-order Predicate Logic
Syntax and Semantics
Variable Replacement and Substitution
31 Quantifiers and Normal Forms
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*

Introduction

Basics of First-order Predicate Logic

Syntax and Semantics

Variable Replacement and Substitution

Quantifiers and Normal Forms

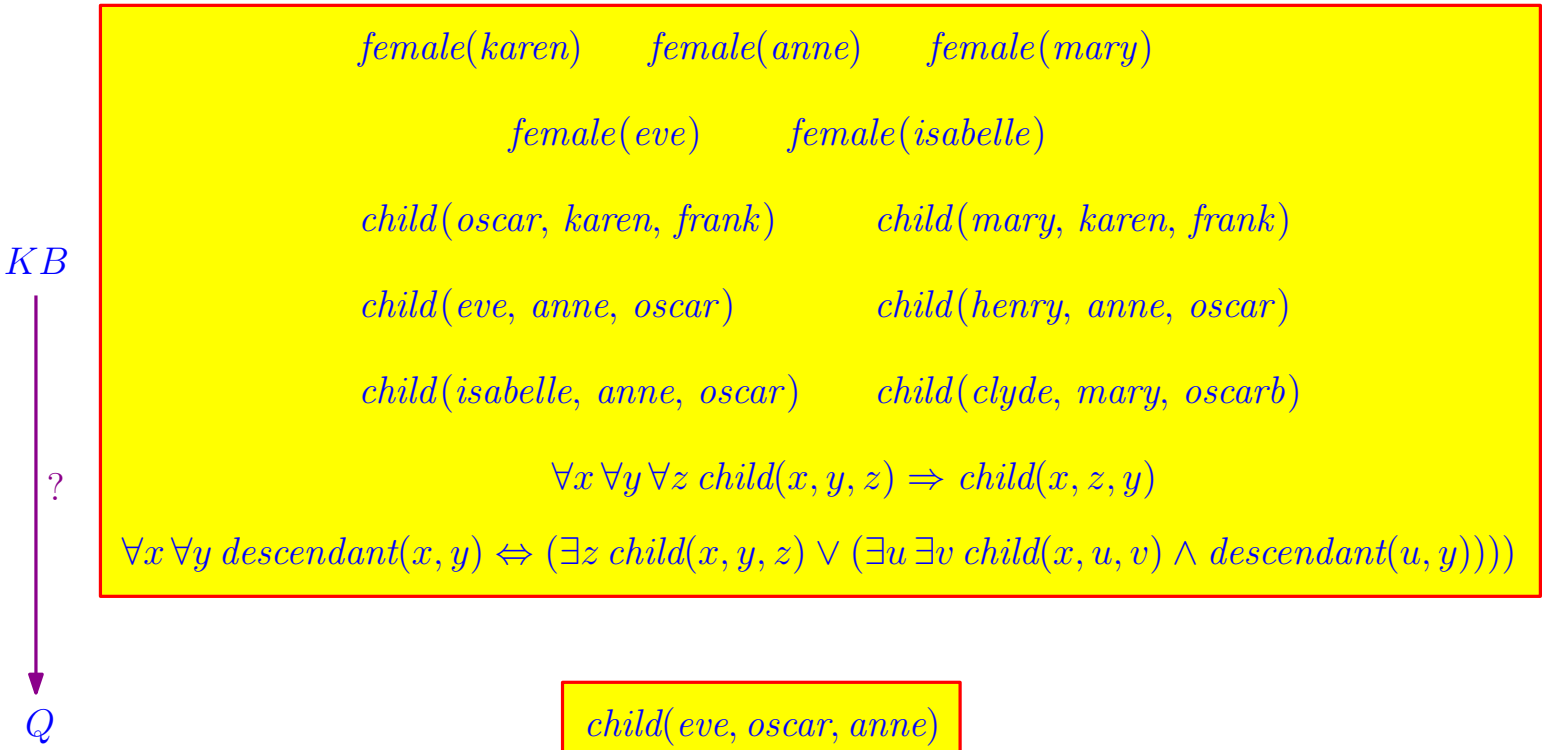
32 Proof Calculi for Predicate Logic

Automated Theorem Provers

References

Basics of First-order Predicate Logic

Proof Calculi for Predicate Logic



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

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
34 Proof Calculi for Predicate Logic
Automated Theorem Provers
References

Exercise 4

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 $KB \nvdash Q$ using only Modus Ponens and $\forall$-Elimination)

(**Hint:** Think about what appears in the knowledge base but not in the inference rules)

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
35 Proof Calculi for Predicate Logic
Automated Theorem Provers
References

Theorem 2 (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 $KB \vdash \varphi$.

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)

Introduction

Basics of First-order Predicate Logic

Syntax and Semantics

Variable Replacement and Substitution

Quantifiers and Normal Forms

36 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 $KB \not\models Q$
- *In PL1:* deriving the empty clause still proves $KB \models Q$, but if no proof exists, the search may continue forever rather than report failure.

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
37 Proof Calculi for Predicate Logic
Automated Theorem Provers
References

Generalized Resolution Rule for PL1

The resolution rule for two clauses in CNF reads

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

where σ *is the MGU of* B *and* B'

What does this definition mean?

- Parent clauses: $A_1 \vee \cdots \vee A_m \vee B$ and $\neg B' \vee C_1 \vee \cdots \vee C_n$
- Before finding the MGU, *rename variables apart if necessary*; clauses should not accidentally share variable names.
- $\sigma(B) = \sigma(B')$ means that B and B' are matched by applying the MGU σ **[What is σ? How to find it?]**
- Apply σ to both clauses, delete the complementary pair, and keep the remaining literals as the resolvent

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

38 Proof Calculi for Predicate Logic

Automated Theorem Provers

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
39 Proof Calculi for Predicate Logic
Automated Theorem Provers
References

Exercise 5

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

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”

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

40 Proof Calculi for Predicate Logic

Automated Theorem Provers

References

Basics of First-order Predicate Logic

Proof Calculi for Predicate Logic

KB

$?$

Q

$female(karen)$ $female(anne)$ $female(mary)$

$female(eve)$ $female(isabelle)$

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

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

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

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

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

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

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

$\neg B'$

$\neg child(x,y,z) \vee child(x,z,y)$

$|||$

B

$child(eve,\ anne,\ oscar)$, $child(x,y,z) \Rightarrow child(x,z,y)$ $\sigma(B) = \sigma(B')$

Gen. Res. ——————————————————

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

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

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

41 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

42 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. $child(eve, anne, oscar)$	KB
2. $\neg child(x, y, z) \vee child(x, z, y)$	KB
3. $\neg child(eve, oscar, anne)$	$\neg Q$
4. $child(eve, oscar, anne)$	GenRes$(1, 2)$
5. $()$	GenRes$(3, 4)$

Note: The previous slide demonstrates only GenRes$(1, 2)$.

Exercise 6

Write a resolution proof for the query $Q = \exists y\, \mathit{knows}(\mathit{henry}, y)$ with respect to the knowledge base $KB = \forall x\, \mathit{knows}(x, \mathit{mother}(x))$, where $\mathit{mother}(x)$ is a function that returns the mother of x, $\mathit{knows}(x, y)$ means "x knows y", and *henry* is a constant representing a person named Henry.

Exercise 7

Use the resolution calculus to prove that the formula $R = \forall x \left[(\neg \mathit{shaves}(\mathit{barber}, x) \vee \neg \mathit{shaves}(x, x)) \wedge (\mathit{shaves}(x, x) \vee \mathit{shaves}(\mathit{barber}, x))\right]$ (a.k.a the Russell's paradox) is unsatisfiable, where $\mathit{shaves}(x, y)$ means "x shaves y" and *barber* is a constant representing a barber. (**Hint:** Remeber that you can also use the Factorization rule along with the Generalized Resolution rule.)

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
43 Proof Calculi for Predicate Logic
Automated Theorem Provers
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
44 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 8

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

Introduction
Basics of First-order Predicate Logic
Syntax and Semantics
Variable Replacement and Substitution
Quantifiers and Normal Forms
45 Proof Calculi for Predicate Logic
Automated Theorem Provers
References

Exercise 9

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

- An *automated theorem prover* is a software tool that uses logical reasoning to automatically search for proofs of 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* [Schulz 2002] to solve some problems in first-order predicate logic

Automated Theorem Provers

Figure: E Theorem Prover

https://wwwlehre.dhbw-stuttgart.de/~sschulz/E/E.html

Automated Theorem Provers

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

Proof of Theorem 3.

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

$$e_l \cdot x = x \tag{1}$$

$$x \cdot e_r = x \tag{2}$$

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

$$e_l \cdot e_r = e_r \tag{3}$$

$$e_l \cdot e_r = e_l \tag{4}$$

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)

$$(x = x)_5 \quad \text{reflexive}$$
$$(\neg x = y \vee y = x)_6 \quad \text{symmetry}$$
$$(\neg x = y \vee \neg y = z \vee x = z)_7 \quad \text{transitivity}$$
$$(\neg x = y \vee m(x, z) = m(y, z))_8 \quad \text{substitution for } m$$
$$(\neg x = y \vee m(z, x) = m(z, y))_9 \quad \text{substitution for } m$$

A resolution proof may be as follows.

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

For example, 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 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.

Automated Theorem Provers

Proof By E Theorem Prover

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

$$(\neg A_1 \vee \cdots \vee \neg A_m \vee B_1 \vee \cdots \vee B_n) \mapsto \texttt{B}_1;\ldots;\texttt{B}_\texttt{n}\texttt{<-}\texttt{A}_1,\ldots,\texttt{A}_\texttt{m}$$

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
```

Automated Theorem Provers

Reading E Output: Conventions

The next slide shows a proof object extracted from E's proof output. E represents clauses as disjunctions of literals, and its proof objects list dependencies before the steps that use them [Schulz 2019], Sec. 3.1 and Sec. 6.4.

- A bracketed list is one *clause*; commas separate literals in an OR-list:

$$\texttt{[++A,--B,--C]} \quad \text{means} \quad A \vee \neg B \vee \neg C.$$

- Equivalently, read the same clause as

$$B \wedge C \Rightarrow A.$$

- ++ marks a *positive literal*; -- marks a *negative literal*.
- A field near the end gives the *reason*; later rows may cite earlier rows or use short E rule names. We will read those rule names when they appear in the output.

Automated Theorem Provers

Reading the E Output

Output from E 3.3.5:

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

Rows 0–16: E simplifies the input clauses, applies paramodulation and simplification, and derives the empty clause `[]`; therefore the negated goal is unsatisfiable and the theorem is proved.

Exercise 10

Use the E theorem prover to solve [Ertel 2025], Exercise 3.9, p. 66. 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

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://wwwlehre.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://wwwlehre.dhbw-stuttgart.de/~sschulz/E/E.html.

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