智能系统原理与开发
第03章 谓词逻辑

李斌
复旦大学 计算机科学技术学院


Propositional logic

Propositional calculus symbols（命题演算符号）

- Propositional symbols
  P, *Q*, *R*, *S*, ...
- Truth symbols
  true, *false*
- Connectives（连接词）
  ∧ conjunction（合取）
  ∨ disjunction（析取）
  ¬ not（否）
  → implication（蕴含）
  ≡ equivalence（等价）

Propositional logic

Propositional calculus sentences（命题演算语句）

- Every propositional symbol and truth symbol is sentences.
  $true$, P, Q
- The ***negation*** of a sentence is a sentence.
  $\neg false$, $\neg P$
- The ***conjunction***, or ***and***, of two sentences is a sentence.
  $P \wedge \neg P$
- The ***disjunction***, or ***or***, of two sentences is a sentence.
  $P \vee \neg P$

Propositional logic

Propositional calculus sentences（命题演算语句）

- The implication of one sentence from another is a sentence.
  $P \rightarrow Q$
- The ***equivalence*** of two sentences is a sentence.
  $P \vee Q \equiv R$

Legal sentences are also called ***well-formed formulas*** （合式公式）or ***WFFs***.

Propositional logic

Propositional calculus semantics（命题演算语义）

The ***interpretation*** of a set of propositions is the assignment of a truth value, either *true* or *false*, to each propositional symbol.

- The truth assignment of ***negation***, ¬*P*, where *P* is any propositional symbol, is *false* if the assignment to *P* is *true*, a *true* if the assignment to *P* is *false*.
- The truth assignment of ***conjunction***, ∧ is *true* only when both conjuncts have *true* value; otherwise it is *false*.
- The truth assignment of ***disjunction***, ∨ is *false* only when both disjuncts have *false* value; otherwise it is *true*.

Propositional logic

Propositional calculus semantics（命题演算语义）

The ***interpretation*** of a set of propositions is the assignment of a truth value, either *true* or *false*, to each propositional symbol.

- The truth assignment of ***implication***, →, is *false* only when the premise or symbol before the implication is *true* and the truth value of the consequent or symbol after the implication is *false*; otherwise it is *true*.
- The truth assignment of ***equivalence***, ≡, is *true* only when both expressions have the same truth assignment for all possible interpretations; otherwise it is *false*.

Presumption

It has rained ⟶ Ground is wet

Truth table

It has rained	*Ground is wet*	*Formula*
false	*false*	*true*
false	*true*	*true*
true	*false*	*false*
true	*true*	*true*

Propositional logic

Equivalences

- $\neg(\neg P) \equiv P$
- $(P \vee Q) \equiv (\neg P \rightarrow Q)$
- $(P \rightarrow Q) \equiv (\neg Q \rightarrow \neg P)$ (The ***contrapositive*** law)
- $\neg(P \vee Q) \equiv (\neg P \wedge \neg Q)$ (The ***De Morgan's*** law)
  $\neg(P \wedge Q) \equiv (\neg P \vee \neg Q)$
- $(P \vee Q) \equiv (Q \vee P)$ (The ***commutative*** laws)
  $(P \wedge Q) \equiv (Q \wedge P)$
- $((P \vee Q) \vee R)) \equiv (P \vee (Q \vee R))$ (The ***associative*** laws)
  $((P \wedge Q) \wedge R)) \equiv (P \wedge (Q \wedge R))$
- $P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$ (The ***distributive*** laws)
  $P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R)$

Proof

$(\neg P \vee Q) \equiv (P \rightarrow Q)$

Truth table

P	Q	$\neg P$	$\neg P \vee Q$	$P \rightarrow Q$	$(\neg P \vee Q) \equiv (P \rightarrow Q)$
true	*true*	*false*	*true*	*true*	*true*
true	*false*	*false*	*false*	*false*	*true*
false	*true*	*true*	*true*	*true*	*true*
false	*false*	*true*	*true*	*true*	*true*

命题逻辑 vs 一阶逻辑

All men are mortal

he is a man ⟶ he will die

Socrates is man

Socrates is a man

Will Socrates die?

First order predicate calculus

$$\left.\begin{array}{l} \forall x(man(x) \rightarrow mortal(x)) \\ man(Socrates) \end{array}\right\} \Longrightarrow mortal(Socrates)$$

First-order logic

Symbols and terms

- ***Truth symbols*** （真值符号）
  true and *false*.
- ***Constant symbols***（常量符号）
- ***Variable symbols***（变量符号）
- ***Function symbols*** （函数符号）
  with ***arity*** indicating the number of elements

First-order logic

Predicates and atomic sentences（谓词与原子语句）

- ***Predicate symbols*** with an associated non-negative integer referred to as the ***arity*** or ***argument number*** for the predicates.
- An ***atomic sentence*** is a predicate constant of arity n, followed by n terms, t_1, t_2, ..., t_n, enclosed in parentheses and separated by commas.
- The truth values, *true* and *false*, are atomic sentences.

Atomic Sentence	Interpretation
Tet(a)	a is a tetrahedron
Cube(a)	a is a cube
SameSize(a, b)	a is the same size as b
SameShape(a, b)	a is the same shape as b
Larger(a, b)	a is larger than b

First-order logic

Logical connectives

- $\wedge$ conjunction
- $\vee$ disjunction
- $\neg$ not
- $\rightarrow$ implication
- $\equiv$ equivalence
- $\exists$ existential quantifier（存在量词）
- $\forall$ universal quantifier（全程量词）

First-order logic

Predicate calculus sentences

- Every ***atomic*** sentence is a sentence.
- If s is a sentence, then so is its ***negation***, $\neg s$.
- If s_1 and s_2 is a sentence, then so is their ***conjunction***, $s_1 \wedge s_2$.
- If s_1 and s_2 is a sentence, then so is their ***disjunction***, $s_1 \vee s_2$.
- If s_1 and s_2 is a sentence, then so is their ***implication***, $s_1 \rightarrow s_2$.
- If s_1 and s_2 is a sentence, then so is their ***equivalence***, $s_1 \equiv s_2$.
- If x is a variable and s is a sentence, then $\forall x\ s$, is a sentence.
- If x is a variable and s is a sentence, then $\exists x\ s$, is a sentence.

Family relationships

mother(*eve*, *abel*)
mother(*eve*, *cain*)
father(*adam*, *abel*)
father(*adam*, *cain*)

$\forall x \forall y((father(x,y) \vee mother(x,y) \rightarrow parent(x,y))$
$\forall x \forall y \forall z((parent(x,y) \wedge parent(x,z) \rightarrow sibling(y,z))$

Intuitively, it is clear that these implications can be used to infer facts such as

sibling(*cain*, *abel*)

First-order logic

Interpretation（解释）

Let the domain **D**（论域）be a nonempty set. An ***interpretation*** over **D** is an assignment of the entities of **D** to each of the constant, variable, predicate, and function symbols of a predicate calculus expression, such that:

- Each ***constant*** is assigned an ***element*** of **D**.
- Each ***variable*** is assigned to a ***nonempty subset*** of **D**; these are the allowable substitutions for that variable.
- Each ***function*** *f* of arity *m* is defined on *m* arguments of **D** and defines a mapping from $\mathbf{D}^m$ into **D**.
- Each ***predicate*** *p* of arity *n* is defined on *n* arguments of **D** and defines a mapping from $\mathbf{D}^n$ into {*true*, *false*}.

First-order logic

Truth value of predicate calculus expressions

Assume an expression *E* and an interpretation **I** for *E* over a nonempty domain **D**. The truth value for *E* is determined by:

- The value of a ***constant*** is the element of **D** it is assigned to by **I**.
- The value of a ***variable*** is the set of elements of **D** it is assigned to by **I**.
- The value of a ***function*** expression is that element of **D** obtained by evaluating the function for the parameter values assigned by the interpretation **I**.
- The value of an ***atomic sentence*** is either *true* or *false*, as determined by the interpretation **I**.

First-order logic

Truth value of predicate calculus expressions

Assume an expression *E* and an interpretation **I** for *E* over a nonempty domain **D**. The truth value for *E* is determined by:

- The value of expressions using $\neg, \wedge, \vee, \rightarrow$, and $\equiv$ is determined from the value of their operands.
- The value of $\forall x\ s$ is *true* if *s* is *true* for all assignments to *x* under **I**, and it is *false* otherwise.
- The value of $\exists x\ s$ is *true* if there is an assignment to *x* in the interpretation **I** under which *s* is *true*; otherwise it is *false*.

First-order logic

First-order predicate calculus

- First-order predicate calculus allows ***quantified variables*** to refer to objects in the domain of discourse and not to ***predicates*** or ***functions***.

$\forall(likes)likes(george,kate)$

Higher-order predicate calculi

Examples

- If it does not rain on Monday, Tom will go to the mountains.

  $\neg weather(rain,monday) \rightarrow go(tom,mountains)$
- Emma is a Doberman pinscher and a good dog.

  $gooddog(emma) \wedge isa(emma,doberman)$
- All basketball player are tall.

  $\forall x(basketball\_player(x) \rightarrow tall(x))$
- Some people like anchovies.

  $\exists x(person(x) \wedge likes(x,anchovies))$
- Nobody likes taxes.

  $\neg\exists x(likes(x,taxes))$

Mathematics

Constant: 0, 5.4, π

Variable: x, y, z

Function: $sin(x)$, $log(base, y)$, $sum(x, y)$

} ***Item***

Predicate: $equal(t_1, t_2)$, $less(t_1, t_2)$, $congruence(t_1, t_2)$

} ***Atomic Sentence***

Sentence:

$$\forall x \forall y \forall z(equal(x,y) \wedge equal(y,z) \rightarrow equal(x,z))$$

$$\forall x \forall y(\neg equal(x,y) \wedge less(x,y) \rightarrow \exists z(less(z,y) \wedge less(x,z)))$$

Semantics

Interpretation

Inference

Axiom + Theorem (a part) ⇨ ***New Theorem***

红楼梦人物关系

Constant: 贾宝玉, 史湘云、刘姥姥

Variable: x, y, z

Function: $subtract(x)$, $multiply(x, y)$, $sum(x, y)$

Predicate: $father(t_1, t_2)$, $mother(t_1, t_2)$, $parent(t_1, t_2)$

} ***Atomic Sentence***

Sentence:

$$\forall x \forall y((father(x,y) \vee mother(x,y) \rightarrow parent(x,y))$$

$$\forall x \forall y \forall z((parent(x,y) \wedge parent(x,z) \rightarrow sibling(y,z))$$

Semantics

Interpretation

Inference

Facts + Sentences ⇨ ***New Sentence***

Inference（推理）

- The ability to ***infer*** new correct expressions from a set of true assertions is an important feature of the predicate calculus.
- These new expressions are correct in that they are ***consistent with***（相容）all previous interpretations of the original set of expressions.
- An interpretation that makes a sentence true is said to ***satisfy***（满足）that sentence.
- An expression *x* ***logically follows*** from a set of predicate calculus expressions ***S*** if every interpretation that ***satisfies*** *S* also ***satisfies*** *x*.
- Inference rules produce new sentences based on the ***syntactic form*** of given logical assertions.

First-order logic

Satisfy, model, valid, inconsistent

For a predicate calculus expression *x* and an interpretation **I**:

- If *x* has a ***value*** of *true* under **I** and a particular variable assignment, then **I** is said to ***satisfy*** *x*.
- If **I** satisfies *x* for all variable assignments, then **I** is a ***model*** of *x*.
- *x* is ***satisfiable*** if and only if there exist an interpretation and variable assignment that satisfy it; otherwise, it is ***unsatisfiable***.

First-order logic

Satisfy, model, valid, inconsistent

For a predicate calculus expression x and an interpretation **I**:

- A set of expressions are ***satisfiable*** if and only if there exist an interpretation and variable assignment that satisfy every element.
- If a set of expressions are not satisfiable, it is said to be ***inconsistent***（不相容）.
- If x has a value *true* for all possible interpretations, x is said to be ***valid***（有效的）.

Inconsistent: $\exists x\,(P(x) \wedge \neg P(x))$
Valid: $(P(x) \vee \neg P(x))$

Proof procedure

A ***proof procedure*** is a combination of an inference rule and an algorithm for applying that rule to a set of logical expressions to generate new sentences.

Logical follows, sound, and complete

- A predicate calculus expression x ***logically follows*** from a set ***S*** of predicate calculus expression if every interpretation and variable assignment that satisfies ***S*** also satisfies x.
- An inference rule is ***sound*** if every predicate calculus expression produced by the rule from a set of ***S*** of predicate calculus expressions also logically follows from ***S***.（soundness可靠性）
- An inference rule is ***complete*** if, given a set ***S*** of predicate calculus expressions, the rule can infer every expression that logically follows from ***S***.（completeness完备性）

Rules

Modus ponens, modus tollens, elimination

- If the sentences P and $P \rightarrow Q$ are known to be *true*, then ***modus ponens*** let us infer Q.
- Under the inference rule ***modus tollens***, if $P \rightarrow Q$ are known to be *true* and Q is known to be *false*, we can infer $\neg P$.
- ***Elimination*** allows us to infer the truth of either of the conjuncts from the truth of a conjunctive sentence. For instance, $P \wedge Q$ let us conclude P and Q are *true*.

肯定前件（拉丁语：modus ponens）
否定后件（拉丁语：modus tollens）

Rules

Introduction, universal instantiation

- ***Introduction*** lets us infer the truth of a conjunction from the truth of its conjuncts. For instance, if P and Q are *true*, then $P \wedge Q$.
- ***Universal instantiation*** states that if any universally quantified variable in a *true* sentence is replaced by any appropriate term from the domain, the result is a *true* sentence. Thus, if a is from the domain of x, $\forall x\, P(x)$ lets us infer $P(a)$.

消去（elimination）
引入（introduction）
全称实例化（universal instantiation）

Example

$\forall x(man(x) \rightarrow mortal(x))$
$man(Socrates)$

Unification ***{ Socrates/x }***
合一（unification）

$man(x) \rightarrow mortal(x)$
$man(Socrates) \rightarrow mortal(Socrates)$ (***Universal instantiation***)

$man(Socrates)$
$mortal(Socrates)$ (***Modus ponens***)

Unification

Most general unifier (mgu)

If s is any unifier of expressions E and g is the ***most general unifier*** of that set of expressions, then for s applied to E there exists another unifier s' such that $Es = Egs'$, where Es and Egs' are the ***composition*** of unifiers applied to the expression E.

- Any constant is considered a ***ground instance*** and may not be replaced. Neither can two different ground instances be substituted with one variable.
- A variable cannot be unified with a term containing that variable. 即x不可以被p(x)替代

最一般合一式（most general unifier）

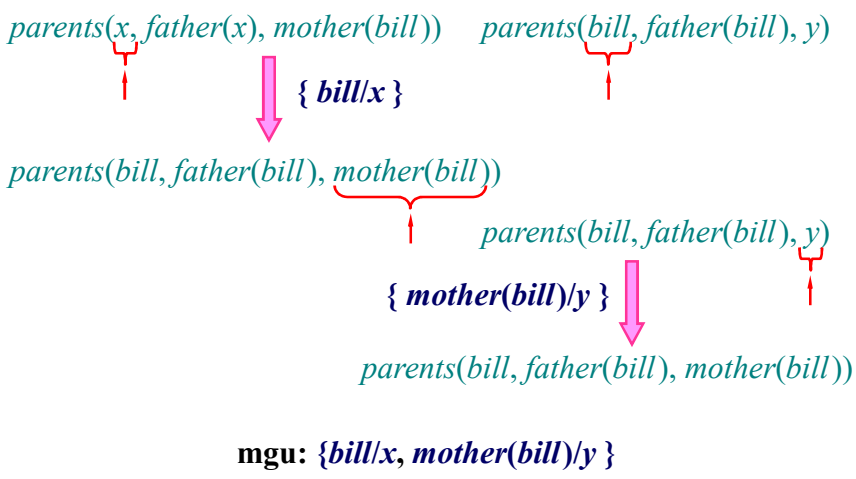
Unification

Most general unifier (mgu)

If *s* is any unifier of expressions *E* and *g* is the ***most general unifier*** of that set of expressions, then for *s* applied to *E* there exists another unifier *s'* such that *Es* = *Egs'*, where *Es* and *Egs'* are the ***composition*** of unifiers applied to the expression *E*.

- It is important that any unifying substitution be made ***consistently*** across all occurrences of the variable in both expressions being matched.
- The unifier must be as ***general*** as possible: that the most general unifier be found for the two expressions.

Unification example



mgu: {*bill*/*x*, *mother*(*bill*)/*y* }

A logic-based financial advisor

The function of the advisor is to help a user decider whether to invest in a savings account or the stock market.

- Individuals with an ***inadequate savings account*** should always make increasing the amount saved their first priority.
- Individuals with an ***adequate savings account*** and an ***adequate income*** should consider a riskier but potentially more profitable investment in the stock market.
- Individuals with a ***lower income*** who already have an ***adequate savings account*** may want to consider splitting their surplus income between savings and stocks.

A logic-based financial advisor

[1] $savings\_account(inadequate) \rightarrow investment(savings)$

[2] $savings\_account(adequate) \wedge income(adequate) \rightarrow investment(stocks)$

[3] $savings\_account(adequate) \wedge income(inadequate) \rightarrow investment(combinaton)$

A logic-based financial advisor

[4] $\forall x(amount\_saved(x) \wedge \exists y(dependents(y) \wedge greater(x, minsavings(y))) \rightarrow savings\_account(adequate))$

[5] $\forall x(amount\_saved(x) \wedge \exists y(dependents(y) \wedge \neg greater(x, minsavings(y))) \rightarrow savings\_account(inadequate))$

[6] $\forall x(earnings(x, steady) \wedge \exists y(dependents(y) \wedge greater(x, minincome(y))) \rightarrow income(adequate))$

[7] $\forall x(earnings(x, steady) \wedge \exists y(dependents(y) \wedge \neg greater(x, minincome(y))) \rightarrow income(inadequate))$

[8] $\forall x(earnings(x, unsteady) \rightarrow income(inadequate))$

A logic-based financial advisor

[9] $amount\_saved(22000)$

[10] $earnings(25000, steady)$

[11] $dependents(3)$

Functions:

$minsavings(x) = 5000 \times x$

$minincome(x) = 15000 + (4000 \times x)$

A logic-based financial advisor

[10] $earnings(25000, steady)$

[11] $dependents(3)$

[7] $\forall x(earnings(x, steady) \wedge \exists y(dependents(y) \wedge \neg greater(x, minincome(y))) \rightarrow income(inadequate))$

{ 25000/*x*, 3/*y* }

[12] $earnings(25000, steady) \wedge dependents(3) \wedge \neg greater(25000, minincome(3)) \rightarrow income(inadequate)$

[13] $income(inadequate)$

A logic-based financial advisor

[9] $amount\_saved(22000)$

[11] $dependents(3)$

[4] $\forall x(amount\_saved(x) \wedge \exists y(dependents(y) \wedge greater(x, minsavings(y))) \rightarrow savings\_account(adequate))$

{ 22000/*x*, 3/*y* }

[14] $amount\_saved(22000) \wedge dependents(3) \wedge greater(22000, minsavings(3)) \rightarrow savings\_account(adequate)$

[15] $savings\_account(adequate)$

A logic-based financial advisor

[13] *income*(*inadequate*)

[15] *savings _ account*(*adequate*)

[3] *savings _ account*(*adequate*) ∧ *income*(*inadequate*)
→ *investment*(*combinaton*)

[16] *investment*(*combination*)

Answer:

Individuals with a lower income who already have an adequate savings account may want to consider splitting their surplus income between savings and stocks (***combination***).

Resolution refutation proofs

- Put the premises or axioms into ***clause form***.
- Add the ***negation*** of what is to be proved, in clause form, to the set of axioms.
- ***Resolve*** these clauses together, producing new clauses that logically follow from them.
- Produce a ***contradiction*** by generating the ***empty clause***.
- The ***substitutions*** used to produce the empty clause are those under which the opposite of the negated goal is true.

Resolution refutation（归结反驳），即反证法

证明：A→B ←→ ~A∨B ←→ ~(A^~B) ←→ 证明A^~B矛盾，如果矛盾，结论成立

1、将待证明问题A→B转化为其逆命题A^~B

2、求合取范式，得到子句集（构成合取范式的有限个简单析取式的集合就是子句集）

3、对子句集进行归结，得到空子句

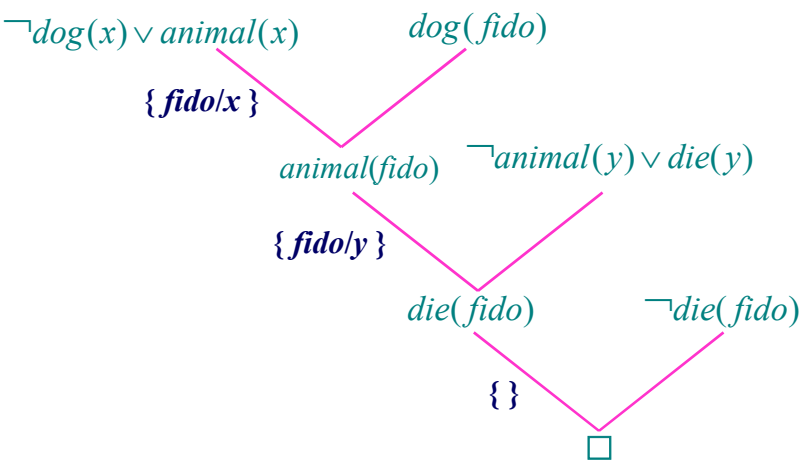
Example

- All dogs are animals
  $\forall x(dog(x) \rightarrow animal(x))$
  Clause Form
  $\neg dog(x) \vee animal(x)$
- All animals will die
  $\forall y(animal(y) \rightarrow die(y))$
  $\neg animal(y) \vee die(y)$
- Fido is a dog
  $dog(fido)$
  $dog(fido)$

Question: ***Will Fido die?***

$die(fido)$
$\neg die(fido)$

Example



Producing the clause form

$$\forall x([a(x) \wedge b(x)] \rightarrow [c(x,i) \wedge \exists y(\exists z([c(y,z)] \rightarrow d(x,y)))]) \vee \forall x(e(x))$$

1. We eliminate the $\rightarrow$ by using the equivalent form.

$$a \rightarrow b \equiv \neg a \vee b$$

$$\forall x(\neg[a(x) \wedge b(x)] \vee [c(x,i) \wedge \exists y(\exists z[\neg c(y,z)] \vee d(x,y))]) \vee \forall x(e(x))$$

Producing the clause form

$$\forall x(\neg[a(x) \wedge b(x)] \vee [c(x,i) \wedge \exists y(\exists z[\neg c(y,z)] \vee d(x,y))]) \vee \forall x(e(x))$$

2. We reduce the scope of negation. This may be accomplished using a number of the transformations.

$$\neg(\neg a) \equiv a$$
$$\neg\exists x(a(x)) \equiv \forall x(\neg a(x))$$
$$\neg\forall x(b(x)) \equiv \exists x(\neg b(x))$$
$$\neg(a \wedge b) \equiv \neg a \vee \neg b$$
$$\neg(a \vee b) \equiv \neg a \wedge \neg b$$

$$\forall x([\neg a(x) \vee \neg b(x)] \vee [c(x,i) \wedge \exists y(\exists z[\neg c(y,z)] \vee d(x,y))]) \vee \forall x(e(x))$$

Producing the clause form

$$\forall x([\neg a(x) \vee \neg b(x)] \vee [c(x,i) \wedge \exists y(\exists z[\neg c(y,z)] \vee d(x,y))]) \vee \forall x(e(x))$$

3. We standardize by renaming all variables so that variables bound by different quantifiers have unique names.

$$\forall x(a(x)) \vee \forall x(b(x)) \equiv \forall x(a(x)) \vee \forall y(b(y))$$

$$\forall x([\neg a(x) \vee \neg b(x)] \vee [c(x,i) \wedge \exists y(\exists z[\neg c(y,z)] \vee d(x,y))]) \vee \forall w(e(w))$$

Producing the clause form

$$\forall x([\neg a(x) \vee \neg b(x)] \vee [c(x,i) \wedge \exists y(\exists z[\neg c(y,z)] \vee d(x,y))]) \vee \forall w(e(w))$$

4. Move all quantifiers to the left without changing their order.

$$\forall x \exists y \exists z \forall w([\neg a(x) \vee \neg b(x)] \vee [c(x,i) \wedge (\neg c(y,z) \vee d(x,y))]) \vee e(w)$$

Producing the clause form

$$\forall x\exists y\exists z\forall w([\neg a(x)\vee\neg b(x)]\vee[c(x,i)\wedge(\neg c(y,z)\vee d(x,y))])\vee e(w)$$

5. All existential quantifiers are eliminated by a process called ***skolemization***（斯科伦化）.

$$\forall x\exists y(mother(x,y)) \qquad \forall x(mother(x,m(x))$$

$$\forall x\forall y\exists z\forall w(foo(x,y,z,w)) \qquad \forall x\forall y\forall w(foo(x,y,f(x,y),w))$$

$$\forall x\forall w([\neg a(x)\vee\neg b(x)]\vee[c(x,i)\wedge(\neg c(f(x),g(x))\vee d(x,f(x)))])\vee e(w)$$

Skolem normal form（斯科伦范式）：对前束合取范式消去所有量词
1）如果存在量词前没有全称量词，则用常量替换并消去存在量词
2）如果存在量词前有全称量词，则用全称量词的函数替代并消去存在量词
3）直接消去全称量词

Producing the clause form

$$\forall x\forall w([\neg a(x)\vee\neg b(x)]\vee[c(x,i)\wedge(\neg c(f(x),g(x))\vee d(x,f(x)))])\vee e(w)$$

6. Drop all universal quantification. By this point only universally quantified variables exit with no variable conflicts.

$$[\neg a(x)\vee\neg b(x)]\vee[c(x,i)\wedge(\neg c(f(x),g(x))\vee d(x,f(x)))]\vee e(w)$$

Producing the clause form

$$[\neg a(x) \vee \neg b(x)] \vee [c(x,i) \wedge (\neg c(f(x),g(x)) \vee d(x,f(x)))] \vee e(w)$$

7. We convert the expression to the conjunct of disjuncts form. This requires using the associative and distributive properties of $\wedge$ and $\vee$.

$$a \vee (b \vee c) \equiv (a \vee b) \vee c$$
$$a \wedge (b \wedge c) \equiv (a \wedge b) \wedge c$$
$$a \vee (b \wedge c) \equiv (a \vee b) \wedge (a \vee c)$$

$$[\neg a(x) \vee \neg b(x) \vee c(x,i) \vee e(w)] \wedge$$
$$[\neg a(x) \vee \neg b(x) \vee \neg c(f(x),g(x)) \vee d(x,f(x)) \vee e(w)]$$

Producing the clause form

$$[\neg a(x) \vee \neg b(x) \vee c(x,i) \vee e(w)] \wedge$$
$$[\neg a(x) \vee \neg b(x) \vee \neg c(f(x),g(x)) \vee d(x,f(x)) \vee e(w)]$$

8. Now call each conjunct a separate clause

$$[\neg a(x) \vee \neg b(x) \vee c(x,i) \vee e(w)]$$
$$[\neg a(x) \vee \neg b(x) \vee \neg c(f(x),g(x)) \vee d(x,f(x)) \vee e(w)]$$

Producing the clause form

$[\neg a(x) \vee \neg b(x) \vee c(x,i) \vee e(w)]$

$[\neg a(x) \vee \neg b(x) \vee \neg c(f(x),g(x)) \vee d(x,f(x)) \vee e(w)]$

9. The final step is to standardize the variables apart again. This requires giving the variable in each clause generated by step 8 different name.

$[\neg a(x) \vee \neg b(x) \vee c(x,i) \vee e(w)]$

$[\neg a(u) \vee \neg b(u) \vee \neg c(f(u),g(u)) \vee d(u,f(u)) \vee e(v)]$

MGU Revisited

All existential quantifiers are eliminated by a process called ***skolemization***（斯科伦化）.

$\forall x \exists y(mother(x,y))$ $\forall x(mother(x,m(x))$

$\forall x \forall y \exists z \forall w(foo(x,y,z,w))$ $\forall x \forall y \forall w(foo(x,y,f(x,y),w))$

Examples（p谓词、f/g函数、小写常量、斜体小写变量）

[1] $\forall x$(p(f(g(x,a),x)) $\forall z$(p(z,b))

mgu: {b/x, f(g(b,a)/z}

p(f(g(b,a)),b)

[2] $\forall x \forall y$(p(f(x,x),y) $\forall z$(p(f(a,z),b)

mgu:{a/x, a/z,b/y}

p(f(a,a),b)

MGU Revisited

Examples（p谓词、f/g函数、小写常量、斜体小写变量）

[3] $\forall x(p(x,x))$ p(a,b)

no unifier

[4] $\forall x \forall y \forall z(p(f(x,y),z)$ $\forall z \forall y(p(z,f(a,y))$

mgu: $\{f(a,y)/z, a/x\}$

p(f(a,*y*),f(a,*y*))

unifier: $\{f(a,b)/z, a/x, b/y\}$

p(f(a,b),f(a,b))

If *s* is any unifier of expressions *E* and *g* is the ***most general unifier*** of that set of expressions, then for *s* applied to *E* there exists another unifier *s′* such that *Es* = *Egs′*, where *Es* and *Egs′* are the ***composition*** of unifiers applied to the expression *E*.

Story of happy student

Anyone passing his history exams and winning the lottery is happy. But anyone who studies or is lucky can pass all his exams. John did not study but he is lucky. Anyone who is lucky wins the lottery. Is John happy.

Story of happy student

- Anyone passing history exams and winning the lottery is happy.
  $\forall x(pass(x,history) \wedge win(x,lottery) \rightarrow happy(x))$
- Anyone who studies or is lucky can pass all his exams.
  $\forall x \forall y(study(x) \vee lucky(x) \rightarrow pass(x,y))$
- John did not study but he is lucky.
  $\neg study(john) \wedge lucky(john)$
- Anyone who is lucky wins the lottery
  $\forall x(lucky(x) \rightarrow win(x,lottery))$

Question: ***Is John happy?***

$\neg happy(john)$

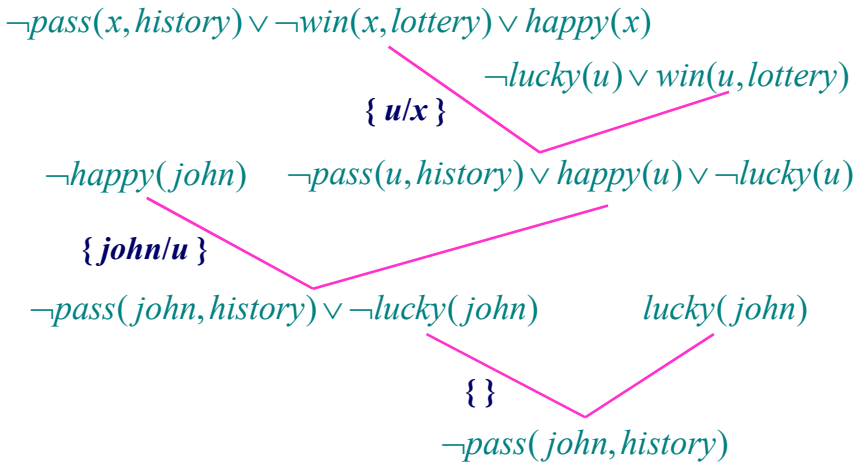
Story of happy student

- Anyone passing history exams and winning the lottery is happy.
  $\neg pass(x,history) \vee \neg win(x,lottery) \vee happy(x)$
- Anyone who studies or is lucky can pass all his exams.
  $\neg study(y) \vee pass(y,z) \quad \neg lucky(w) \vee pass(w,v)$
- John did not study but he is lucky.
  $\neg study(john) \qquad lucky(john)$
- Anyone who is lucky wins the lottery
  $\neg lucky(u) \vee win(u,lottery)$

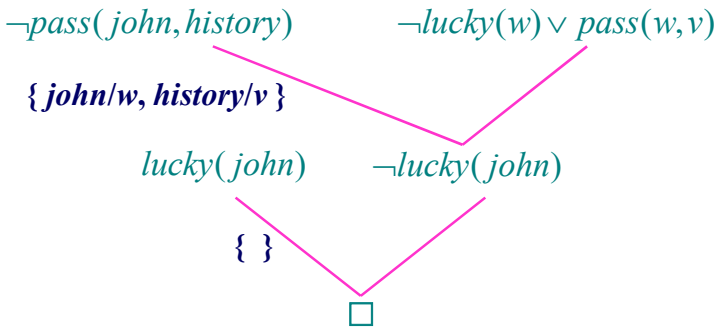
Question: ***Is John happy?***

$\neg happy(john)$

Story of happy student



Story of happy student



Story of exciting life

All people who are not poor and are smart are happy. Those people who read are not stupid. John can read and is wealthy. Happy people have exciting life. Can anyone be found with an exciting life?

We assume

$\forall x(smart(x) \equiv \neg stupid(x))$

$\forall y(wealthy(y) \equiv \neg poor(y))$

Story of exciting life

- All people who are not poor and are smart are happy.
  $\forall x(\neg poor(x) \wedge smart(x) \rightarrow happy(x))$
- Those people who read are not stupid.
  $\forall y(read(y) \rightarrow smart(y))$
- John can read and is wealthy.
  $\neg poor(john) \wedge read(john)$
- Happy people have exciting life
  $\forall z(happy(z) \rightarrow exciting(z))$

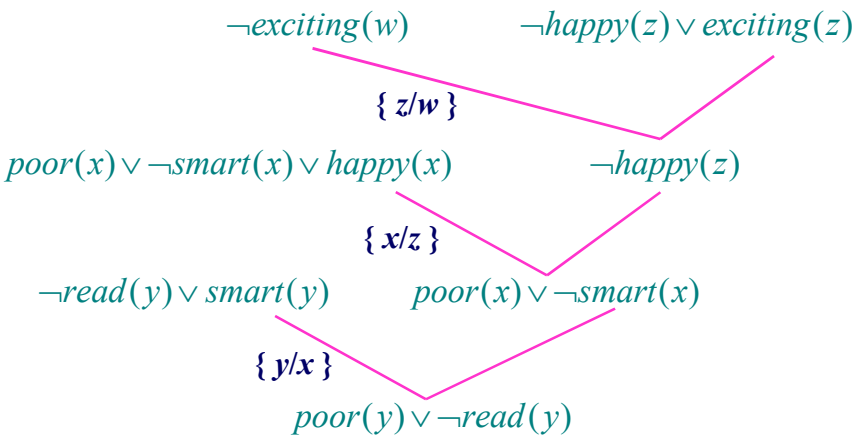
Question: ***Can anyone be found with an exciting life?***
$\neg \exists w(exciting(w))$

Story of exciting life

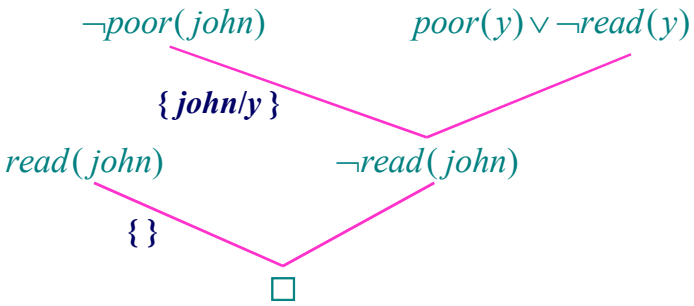
- All people who are not poor and are smart are happy.
  $poor(x) \vee \neg smart(x) \vee happy(x)$
- Those people who read are not stupid.
  $\neg read(y) \vee smart(y)$
- John can read and is wealthy.
  $\neg poor(john)$ $read(john)$
- Happy people have exciting life
  $\neg happy(z) \vee exciting(z)$

Question: ***Can anyone be found with an exciting life?***
$\neg exciting(w)$

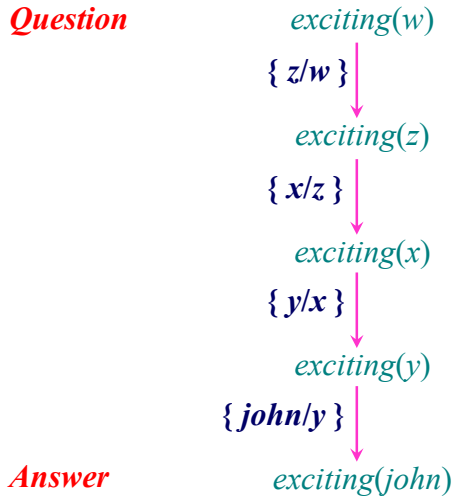
Story of exciting life



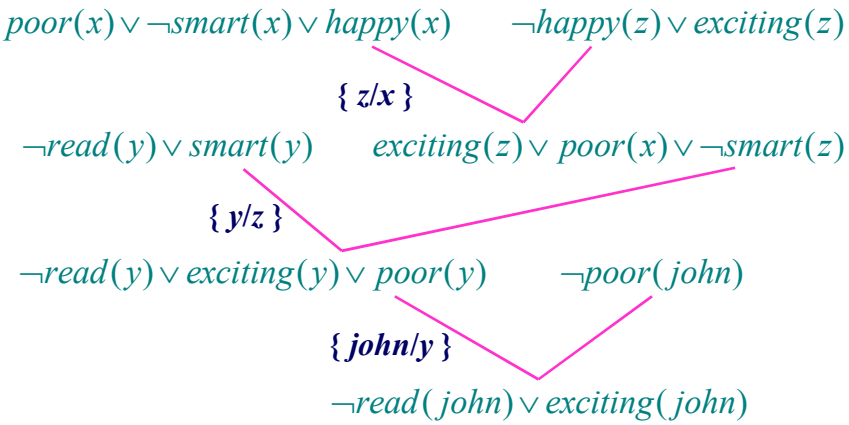
Story of exciting life



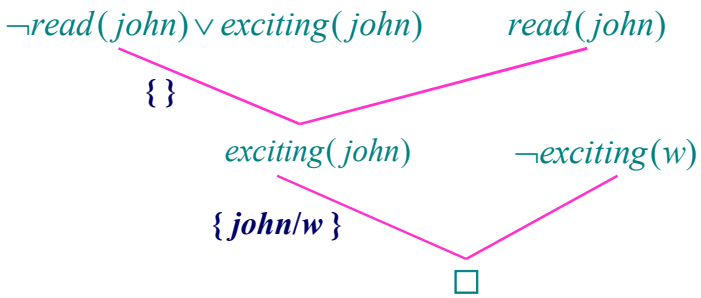
Answer extraction from refutation



Another resolution refutation



Another resolution refutation



Story of grandparent

Everyone has a parent. The parent of a parent is a grandparent. Given the person John, prove that John has a grandparent.

- Everyone has a parent

  $\forall x \exists y parent(x, y)$

- A parent of a parent is a grandparent

  $\forall x \forall y \forall z (parent(x, y) \wedge parent(y, z) \rightarrow grandparent(x, z))$

  Question: ***Has John a grandparent?***

  $\neg \exists w (grandparent(john, w))$

Story of grandparent

Everyone has a parent. The parent of a parent is a grandparent. Given the person John, prove that John has a grandparent.

- Everyone has a parent

  $parent(x, pa(x))$

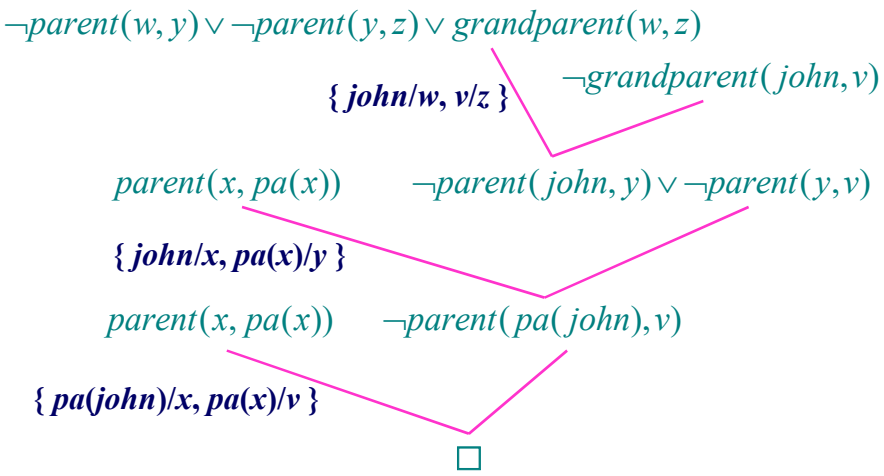
- A parent of a parent is a grandparent

  $\neg parent(w, y) \vee \neg parent(y, z) \vee grandparent(w, z)$

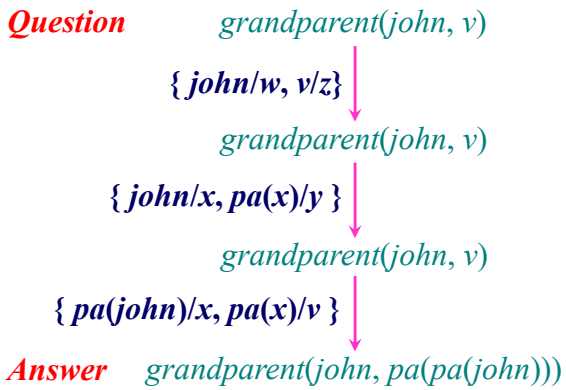
  Question: ***Has John a grandparent?***

  $\neg grandparent(john, v)$

Story of grandparent



Answer extraction from refutation



Horn clauses

A Horn clause contains ***at most one positive literal***

$a \vee \neg b_1 \vee \neg b_2 \vee \square \vee \neg b_n$

To emphasize the key role of the one positive literal in resolutions, we generally write Horn clauses as ***implications*** with the ***positive literal as the conclusion***

$a \leftarrow b_1 \wedge b_2 \wedge \square \wedge b_n$

Horn（霍恩）子句：

$L_1 \vee L_2 \vee \ldots \vee L_n$中如果至多含一个正文字，那么该子句称为Horn子句。

Horn子句$P \vee \neg Q_1 \vee \neg Q_2 \vee \ldots \vee \neg Q_n$通常表示为：$P \leftarrow Q_1 \wedge Q_2 \wedge \ldots \wedge Q_n$

Three forms of Horn clauses

- ***Goal*** clause（询问）

  $\leftarrow a_1 \wedge a_2 \wedge \square \wedge a_n$

- ***Definite*** clause or ***Facts***（事实）

  $a_1 \leftarrow \qquad a_2 \leftarrow \qquad a_n \leftarrow$

- ***Rules*** relation（规则）

  $a_1 \leftarrow b_1 \wedge b_2 \wedge \square \wedge b_n$

- Resolution of two Horn clauses always results in a Horn clause
- Resolution of a goal clause and a definite clause is always a goal clause

Prolog

Given a ***goal***

$\leftarrow a_1 \wedge a_2 \wedge \square \wedge a_n$

Prolog interpreter sequentially searches for the ***first clause*** in logic program whose head unifies with a_1. if

$a^1 \leftarrow b_1 \wedge b_2 \wedge \square \wedge b_n$

is the ***reducing clause*** with the unification ξ, the goal clause then becomes

$\leftarrow (b_1 \wedge b_2 \wedge \square \wedge b_n \wedge a_2 \wedge \square \wedge a_n)\xi$

Logic programming language Prolog is based on computing with Horn clauses

Prolog

The Prolog interpreter then continues by trying to reduce the ***leftmost goal***, b_1

$b^1 \leftarrow c_1 \wedge c_2 \wedge \square \wedge c_p$

under unification ζ. the ***goal*** then becomes

$\leftarrow (c_1 \wedge c_2 \wedge \square \wedge c_p \wedge b_2 \wedge \square \wedge b_m \wedge a_2 \wedge \square \wedge a_n)\xi\zeta$

if the goal is reduced to the null clause then the ***composition of unifications*** that made the reductions

$\leftarrow (\square)\xi\zeta \cdots \varsigma$

provide an ***interpretation*** under which the original goal clause was true.

Syntax for Prolog

English	Predicate calculus	Prolog
and	$\wedge$	,
or	$\vee$	;
only if	$\rightarrow$	:-
not	$\neg$	**not**

Prolog(**Pro**gramming in **log**ic)语言是以Horn子句逻辑为基础的高级程序设计语言。

Example

likes(*george*, *kate*)
likes(*george*, *susie*)
likes(*george*, *wine*)
likes(*susie*, *wine*)
likes(*kate*, *gin*)

?- *likes*(*george*, *susie*).
yes
?- *likes*(*george*, *gin*).
no
?- *likes*(*george*, *x*).
x = *kate*
;
x = *susie*
;
x = *wine*
;
no

Example

likes(*george*, *kate*)
likes(*george*, *susie*)
likes(*george*, *wine*)
likes(*susie*, *wine*)
likes(*kate*, *gin*)
friends(*x*, *y*) :- *likes*(*x*, *z*), *likes*(*y*, *z*)

?- *friends*(*george*, *susie*).
yes

Example

ancestor(X, Y) :- parent(X, Y)
ancestor(X, Y) :- ancestor(X, Z), ancestor(Z, Y)
parent(ann, mary)
parent(ann, susan)
parent(mary, bob)
parent(susan, john)

?- ancestor(ann, susan)
yes
?- ancestor(ann, john)
yes

Example

r:- p, s
s:- p, q
p:-
q:-
?- r

(1) r:- p, s
(2) s:- p, q
(3) p:-
(4) q:-
(5) :- r
(6) s:- q (2)(3)归结
(7) s:- (4)(6)归结
(8) r:- s (1)(3)归结
(9) r:- (7)(8)归结
(10) :- (5)(9)归结

Example

(1) cousin(x, y):- parent(u, x), parent(v, y), brother(u, v)
(2) parent(贾政, 贾宝玉):-
(3) parent(贾敏, 林黛玉):-
(4) brother(贾政, 贾敏):-
(5) :- cousin(贾宝玉, y)

(6) :- parent(u, 贾宝玉), parent(v,y), brother(u,v) (1)(5)归结贾宝玉/x
(7) :- parent(v,y), brother(贾政,v) (2)(6)归结贾政/u
(8) :- parent(贾敏,y) (4)(7)归结贾敏/v
(9) :- (3)(8)归结林黛玉/y

Categories of logic systems

- Very expressive but ***undecidable*** logics, typically variants of first- or higher-order logics;
- Quantifier-free formalisms of low computational complexity such as classical propositional logic.
- ***Decidable*** logics with restricted quantification located between propositional and first-order logics.

Thanks