形式化方法导引

第 2 章 经典数理逻辑-问题定义

黄文超

https://faculty.ustc.edu.cn/huangwenchao

$\longrightarrow$ 教学课程 $\longrightarrow$ 形式化方法导引

本章内容

- Define verification
  - $\mathcal{M} \models \phi$
- A method of define $\mathcal{M}$ and ϕ: Logics
  - Propositional logic
  - Predicate logic
  - Higher-order logic

回顾: 定义: Verifier

A *verifier* for a language A is an algorithm V, where

$$A = \{w \mid V \text{ accepts } \langle w, c\rangle \text{ for some string } c\}.$$

回顾: 验证过程

(1) 构建模型 w. **(2)** 设计规约 A. **(3)** (手动或自动) 构建证明 c
(4) 使用验证器 V, 输入 c, 输出是否$w \in A$

Verifier —— Logic

回顾: 验证过程

(1) 构建模型 w. **(2)** 设计规约 A. **(3)** (手动或自动) 构建证明 c
(4) 使用验证器 V, 输入 c, 输出是否$w \in A$

定义: Verification in Logics

Most logics used in the design, specification, and verification of computer systems fundamentally deal with a *satisfaction relation*:

$$\mathcal{M} \vDash \phi$$

- $\mathcal{M}$ is some sort of situation or *model* of a system
- ϕ is a *specification*, a formula of that logic, expressing what should be true in situation $\mathcal{M}$.
- At the *heart* of this set-up is that one can often *specify and implement algorithms* for computing $\vDash$.

回顾: 验证过程

(1) 构建模型 w. **(2)** 设计规约 A. **(3)** (手动或自动) 构建证明 c
(4) 使用验证器 V, 输入 c, 输出是否$w \in A$

定义: Verification in Logics

Most logics used in the design, specification, and verification of computer systems fundamentally deal with a *satisfaction relation*:

$$\mathcal{M} \vDash \phi$$

- $\mathcal{M}$ is some sort of situation or *model* of a system
- ϕ is a *specification*, a formula of that logic, expressing what should be true in situation $\mathcal{M}$.
- At the *heart* of this set-up is that one can often *specify and implement algorithms* for computing $\vDash$.

Verifier —— Logic

回顾: 验证过程

(1) 构建模型 w. **(2)** 设计规约 A. **(3)** (手动或自动) 构建证明 c
(4) 使用验证器 V, 输入 c, 输出是否$w \in A$

定义: Verification in Logics

Most logics used in the design, specification, and verification of computer systems fundamentally deal with a *satisfaction relation*:

$$\mathcal{M} \vDash \phi$$

- $\mathcal{M}$ is some sort of situation or *model* of a system
- ϕ is a *specification*, a formula of that logic, expressing what should be true in situation $\mathcal{M}$.
- At the *heart* of this set-up is that one can often *specify and implement algorithms* for computing $\vDash$.

Verifier —— Logic

回顾: 验证过程

(1) 构建模型 w. **(2)** 设计规约 A. **(3)** (手动或自动) 构建证明 c
(4) 使用验证器 V, 输入 c, 输出是否$w \in A$

定义: Verification in Logics

Most logics used in the design, specification, and verification of computer systems fundamentally deal with a *satisfaction relation*:

$$\mathcal{M} \models \phi$$

- $\mathcal{M}$ is some sort of situation or *model* of a system
- ϕ is a *specification*, a formula of that logic, expressing what should be true in situation $\mathcal{M}$.
- At the *heart* of this set-up is that one can often *specify and implement algorithms* for computing $\models$.

Verifier —— Logic

回顾: 验证过程

(1) 构建模型 w. **(2)** 设计规约 A. **(3)** (手动或自动) 构建证明 c
(4) 使用验证器 V, 输入 c, 输出是否$w \in A$

定义: Verification in Logics

Most logics used in the design, specification, and verification of computer systems fundamentally deal with a *satisfaction relation*:

$$\mathcal{M} \models \phi$$

- $\mathcal{M}$ is some sort of situation or *model* of a system
- ϕ is a *specification*, a formula of that logic, expressing what should be true in situation $\mathcal{M}$.
- At the *heart* of this set-up is that one can often *specify and implement algorithms* for computing $\models$.

Verifier —— Logic

定义: Verification in Logics

Most logics used in the design, specification and verification of computer systems fundamentally deal with a *satisfaction relation*:

$$\mathcal{M} \models \phi$$

- $\mathcal{M}$ is some sort of situation or *model* of a system
- ϕ is a *specification*, a formula of that logic, expressing what should be true in situation $\mathcal{M}$.
- At the *heart* of this set-up is that one can often *specify and implement algorithms* for computing $\models$.

下一个问题:

- 问: 如何统一化定义 $\mathcal{M}$ 和 ϕ? 答: *Logics*
- 问: 如何支持 $\models$ 和 algorithms? 答: *Rules*

Verifier —— Logic

定义: Verification in Logics

Most logics used in the design, specification and verification of computer systems fundamentally deal with a *satisfaction relation*:

$$\mathcal{M} \vDash \phi$$

- $\mathcal{M}$ is some sort of situation or *model* of a system
- ϕ is a *specification*, a formula of that logic, expressing what should be true in situation $\mathcal{M}$.
- At the *heart* of this set-up is that one can often *specify and implement algorithms* for computing $\vDash$.

下一个问题:

- 问: 如何统一化定义 $\mathcal{M}$ 和 ϕ? 答: *Logics*
- 问: 如何支持 $\vDash$ 和 algorithms? 答: *Rules*

Verifier —— Logic

定义: Verification in Logics

Most logics used in the design, specification and verification of computer systems fundamentally deal with a *satisfaction relation*:

$$\mathcal{M} \vDash \phi$$

- $\mathcal{M}$ is some sort of situation or *model* of a system
- ϕ is a *specification*, a formula of that logic, expressing what should be true in situation $\mathcal{M}$.
- At the *heart* of this set-up is that one can often *specify and implement algorithms* for computing $\vDash$.

下一个问题:

- 问: 如何统一化定义 $\mathcal{M}$ 和 ϕ? 答: *Logics*
- 问: 如何支持 $\vDash$ 和 algorithms? 答: *Rules*

定义: Verification in Logics

Most logics used in the design, specification and verification of computer systems fundamentally deal with a *satisfaction relation*:

$$\mathcal{M} \models \phi$$

- $\mathcal{M}$ is some sort of situation or *model* of a system
- ϕ is a *specification*, a formula of that logic, expressing what should be true in situation $\mathcal{M}$.
- At the *heart* of this set-up is that one can often *specify and implement algorithms* for computing $\models$.

下一个问题:

- 问: 如何统一化定义 $\mathcal{M}$ 和 ϕ? 答: *Logics*
- 问: 如何支持 $\models$ 和 algorithms? 答: *Rules*

Verifier —— Logic

定义: Verification in Logics

Most logics used in the design, specification and verification of computer systems fundamentally deal with a *satisfaction relation*:

$$\mathcal{M} \models \phi$$

- $\mathcal{M}$ is some sort of situation or *model* of a system
- ϕ is a *specification*, a formula of that logic, expressing what should be true in situation $\mathcal{M}$.
- At the *heart* of this set-up is that one can often *specify and implement algorithms* for computing $\models$.

下一个问题:

- 问: 如何统一化定义 $\mathcal{M}$ 和 ϕ? 答: *Logics*
- 问: 如何支持 $\models$ 和 algorithms? 答: *Rules*

Verifier —— Logic

定义: Verification in Logics

Most logics used in the design, specification and verification of computer systems fundamentally deal with a *satisfaction relation*:

$$\mathcal{M} \vDash \phi$$

- $\mathcal{M}$ is some sort of situation or *model* of a system
- ϕ is a *specification*, a formula of that logic, expressing what should be true in situation $\mathcal{M}$.
- At the *heart* of this set-up is that one can often *specify and implement algorithms* for computing $\vDash$.

下一个问题:

- 问: 如何统一化定义 $\mathcal{M}$ 和 ϕ? 答: *Logics*
- 问: 如何支持 $\vDash$ 和 algorithms? 答: *Rules*

定义 $\mathcal{M}$ 和 ϕ? Logics

回顾: 如何定义一个问题? – 问题 3

Given a set S, a machine M, and $x \in S$, compute whether $x \in L(M)$.

- M is a *machine*, e.g., *finite automaton*.
- $L(M)$ is the *language* of M.

Define a special group of languages: Logics

- Propositional logic (命题逻辑)
- Predicate logic (谓词逻辑)
  - a.k.a., First-order Logic (一阶逻辑)
- Higher-Order Logic (高阶逻辑)

定义 $\mathcal{M}$ 和 ϕ? Logics

回顾: 如何定义一个问题? – 问题 3

Given a set S, a machine M, and $x \in S$, compute whether $x \in L(M)$.

- M is a *machine*, e.g., *finite automaton*.
- $L(M)$ is the *language* of M.

Define a special group of languages: Logics

- Propositional logic (命题逻辑)
- Predicate logic (谓词逻辑)
  - a.k.a., First-order Logic (一阶逻辑)
- Higher-Order Logic (高阶逻辑)

定义: Propositions (命题)

Declarative sentences which one can argue as being *true* or *false*, e.g.,

- The sum of the numbers 3 and 5 equals 8.
- Jane reacted violently to Jack’ s accusations.
- Every even natural number >2 is the sum of two prime numbers.
- All Martians like pepperoni on their pizza.

- 问题: 过于繁杂...
  - 解决方法: 从原子开始组建...

定义: Atomic Propositions (原子命题)

Propositions which is *indecomposable*, e.g.,

- The number 5 is even.

定义: Propositions (命题)

Declarative sentences which one can argue as being *true* or *false*, e.g.,

- The sum of the numbers 3 and 5 equals 8.
- Jane reacted violently to Jack’ s accusations.
- Every even natural number >2 is the sum of two prime numbers.
- All Martians like pepperoni on their pizza.

- 问题: 过于繁杂...
  - 解决方法: 从原子开始组建...

定义: Atomic Propositions (原子命题)

Propositions which is *indecomposable*, e.g.,

- The number 5 is even.

定义: Propositions (命题)

Declarative sentences which one can argue as being *true* or *false*, e.g.,

- The sum of the numbers 3 and 5 equals 8.
- Jane reacted violently to Jack' s accusations.
- Every even natural number >2 is the sum of two prime numbers.
- All Martians like pepperoni on their pizza.

- 问题: 过于繁杂...
  - 解决方法: 从原子开始组建...

定义: Atomic Propositions (原子命题)

Propositions which is *indecomposable*, e.g.,

- The number 5 is even.

定义: Propositions (命题)

Declarative sentences which one can argue as being *true* or *false*.

定义: Atomic Propositions (原子命题)

Propositions which is *indecomposable*.

Symbols representing Atomic Propositions

We assign certain distinct symbols $p, q, r, \ldots$, or sometimes $p_1, p_2, p_3, \ldots$ to each of these atomic sentences

Symbols representing Logical Operators

$\{\neg, \vee, \wedge, \rightarrow\}$

- We can then code up more complex sentences in a compositional way

定义: Propositions (命题)

Declarative sentences which one can argue as being *true* or *false*.

定义: Atomic Propositions (原子命题)

Propositions which is *indecomposable*.

Symbols representing Atomic Propositions

We assign certain distinct symbols $p, q, r, \ldots$, or sometimes $p_1, p_2, p_3, \ldots$ to each of these atomic sentences

Symbols representing Logical Operators

$\{\neg, \vee, \wedge, \rightarrow\}$

- We can then code up more complex sentences in a compositional way

定义: Propositions (命题)

Declarative sentences which one can argue as being *true* or *false*.

定义: Atomic Propositions (原子命题)

Propositions which is *indecomposable*.

Symbols representing Atomic Propositions

We assign certain distinct symbols $p, q, r, \ldots$, or sometimes $p_1, p_2, p_3, \ldots$ to each of these atomic sentences

Symbols representing Logical Operators

$\{\neg, \vee, \wedge, \rightarrow\}$

- We can then code up more complex sentences in a compositional way

Preparation: given the following atomic sentences

- p: 'I won the lottery last week.'
- q: 'I purchased a lottery ticket.'
- r: 'I won last week' s sweepstakes.'

Definition of the Logical Operators

$\neg$: *Negation*. $\neg p$ denotes *negation of* p
- i.e., it is *not true* that I won the lottery last week.

$\vee$: *Disjunction* (析取). $p \vee r$ denotes *at least one* of $\{p, r\}$ is true.
- i.e., I won the lottery last week, or I won last week' s sweepstakes

$\wedge$: *Conjunction* (合取). $p \wedge r$ denotes *both* p and r are true.
- i.e., Last week I won the lottery and the sweepstakes.

$\rightarrow$: *Implication* (蕴含). $p \rightarrow q$ denotes q is a *logical consequence* of p.
- i.e., If I won the lottery last week, then I purchased a lottery ticket.

Preparation: given the following atomic sentences

- p: 'I won the lottery last week.'
- q: 'I purchased a lottery ticket.'
- r: 'I won last week' s sweepstakes.'

Definition of the Logical Operators

$\neg$: *Negation*. $\neg p$ denotes *negation of p*
- i.e., it is *not true* that I won the lottery last week.

$\vee$: *Disjunction* (析取). $p \vee r$ denotes *at least one* of $\{p, r\}$ is true.
- i.e., I won the lottery last week, or I won last week' s sweepstakes

$\wedge$: *Conjunction* (合取). $p \wedge r$ denotes *both* p and r are true.
- i.e., Last week I won the lottery and the sweepstakes.

$\rightarrow$: *Implication* (蕴含). $p \rightarrow q$ denotes q is a *logical consequence* of p.
- i.e., If I won the lottery last week, then I purchased a lottery ticket.

定义 $\mathcal{M}$ 和 ϕ? Logics

Preparation: given the following atomic sentences

- p: 'I won the lottery last week.'
- q: 'I purchased a lottery ticket.'
- r: 'I won last week' s sweepstakes.'

Definition of the Logical Operators

$\neg$: *Negation*. $\neg p$ denotes *negation of p*

- i.e., it is *not true* that I won the lottery last week.

$\vee$: *Disjunction* (析取). $p \vee r$ denotes *at least one* of $\{p, r\}$ is true.

- i.e., I won the lottery last week, or I won last week' s sweepstakes

$\wedge$: *Conjunction* (合取). $p \wedge r$ denotes *both* p and r are true.

- i.e., Last week I won the lottery and the sweepstakes.

$\rightarrow$: *Implication* (蕴含). $p \rightarrow q$ denotes q is a *logical consequence* of p.

- i.e., If I won the lottery last week, then I purchased a lottery ticket.

Preparation: given the following atomic sentences

- p: 'I won the lottery last week.'
- q: 'I purchased a lottery ticket.'
- r: 'I won last week' s sweepstakes.'

Definition of the Logical Operators

$\neg$: *Negation*. $\neg p$ denotes *negation of* p

- i.e., it is *not true* that I won the lottery last week.

$\vee$: *Disjunction* (析取). $p \vee r$ denotes *at least one* of $\{p, r\}$ is true.

- i.e., I won the lottery last week, or I won last week' s sweepstakes

$\wedge$: *Conjunction* (合取). $p \wedge r$ denotes *both* p and r are true.

- i.e., Last week I won the lottery and the sweepstakes.

$\to$: *Implication* (蕴含). $p \to q$ denotes q is a *logical consequence* of p.

- i.e., If I won the lottery last week, then I purchased a lottery ticket.

Preparation: given the following atomic sentences

- p: 'I won the lottery last week.'
- q: 'I purchased a lottery ticket.'
- r: 'I won last week' s sweepstakes.'

Definition of the Logical Operators

$\neg$: *Negation*. $\neg p$ denotes *negation of* p

- i.e., it is *not true* that I won the lottery last week.

$\vee$: *Disjunction* (析取). $p \vee r$ denotes *at least one* of $\{p, r\}$ is true.

- i.e., I won the lottery last week, or I won last week' s sweepstakes

$\wedge$: *Conjunction* (合取). $p \wedge r$ denotes *both* p and r are true.

- i.e., Last week I won the lottery and the sweepstakes.

$\rightarrow$: *Implication* (蕴含). $p \rightarrow q$ denotes q is a *logical consequence* of p.

- i.e., If I won the lottery last week, then I purchased a lottery ticket.

回顾：自动机

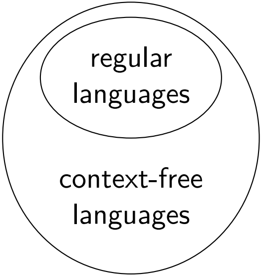
如何定义一个问题？ – 问题 3

Given a set S, a machine M, and $x \in S$, compute whether $x \in L(M)$.

- M is a *machine*, e.g., *finite automaton*.
- $L(M)$ is the *language* of M.

M 的类型？

- regular languages
  - 例：正则表达式匹配、词法分析
- *context-free languages*
  - 例：语法分析



Define *Propositional logic* using a *context-free language*

- Backus-Naur Form (BNF) (巴科斯范式)

回顾：自动机

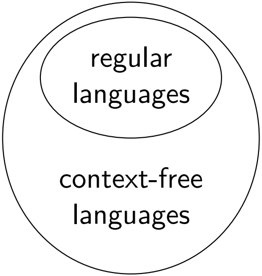
如何定义一个问题？ – 问题 3

Given a set S, a machine M, and $x \in S$, compute whether $x \in L(M)$.

- M is a *machine*, e.g., *finite automaton*.
- $L(M)$ is the *language* of M.

M 的类型？

- regular languages
  - 例：正则表达式匹配、词法分析
- *context-free languages*
  - 例：语法分析



Define *Propositional logic* using a *context-free language*

- Backus-Naur Form (BNF) (巴科斯范式)

定义: Propositional Logic in *BNF*

$$\phi ::= p \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \rightarrow \phi)$$

where p stands for any atomic proposition and each occurrence of ϕ to the right of $::=$ stands for any already constructed formula.

Well-formed formula, 例:

$$(((\neg p) \wedge q) \rightarrow (p \wedge (q \vee (\neg r))))$$

Not well-formed formula, 例:

$$(\neg)() \vee pq \rightarrow$$

定义: Propositional Logic in *BNF*

$$\phi ::= p \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \rightarrow \phi)$$

where p stands for any atomic proposition and each occurrence of ϕ to the right of $::=$ stands for any already constructed formula.

Well-formed formula, 例:

$$(((\neg p) \wedge q) \rightarrow (p \wedge (q \vee (\neg r))))$$

Not well-formed formula, 例:

$$(\neg)() \vee pq \rightarrow$$

定义: Propositional Logic in *BNF*

$$\phi ::= p \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \rightarrow \phi)$$

where p stands for any atomic proposition and each occurrence of ϕ to the right of $::=$ stands for any already constructed formula.

Well-formed formula, 例:

$$(((\neg p) \wedge q) \rightarrow (p \wedge (q \vee (\neg r))))$$

Not well-formed formula, 例:

$$(\neg)() \vee pq \rightarrow$$

- 问题: 怎样利用 atomic propositions 和 logical operators 来计算 ϕ?
  - 基本方法: 使用 Bool 真值表

定义: **T** and **F**

- The set of truth values contains two elements **T** and **F**, where **T** represents ‘true’ and **F** represents ‘false’ .
- A *valuation* or *model* of a formula ϕ is an assignment of each propositional atom in ϕ to a truth value.

ϕ	ψ	$\phi \land \psi$
T	T	T
T	F	F
F	T	F
F	F	F

ϕ	ψ	$\phi \lor \psi$
T	T	T
T	F	T
F	T	T
F	F	F

ϕ	ψ	$\phi \to \psi$
T	T	T
T	F	F
F	T	T
F	F	T

ϕ	$\neg\phi$
T	F
F	T

$\top$
T

$\bot$
F

- 问题: 怎样利用 atomic propositions 和 logical operators 来计算 ϕ?
  - 基本方法: 使用 Bool 真值表

定义: **T** and **F**

- The set of truth values contains two elements **T** and **F**, where **T** represents 'true' and **F** represents 'false'.
- A *valuation* or *model* of a formula ϕ is an assignment of each propositional atom in ϕ to a truth value.

ϕ	ψ	$\phi \wedge \psi$
T	**T**	**T**
T	**F**	**F**
F	**T**	**F**
F	**F**	**F**

ϕ	ψ	$\phi \vee \psi$
T	T	T
T	F	T
F	T	T
F	F	F

ϕ	ψ	$\phi \rightarrow \psi$
T	T	T
T	F	F
F	T	T
F	F	T

ϕ	$\neg\phi$
T	F
F	T

$\top$
T

$\bot$
F

- 问题: 怎样利用 atomic propositions 和 logical operators 来计算 ϕ?
  - 基本方法: 使用 Bool 真值表

定义: **T** and **F**

- The set of truth values contains two elements **T** and **F**, where **T** represents ‘true’ and **F** represents ‘false’ .
- A *valuation* or *model* of a formula ϕ is an assignment of each propositional atom in ϕ to a truth value.

ϕ	ψ	$\phi \wedge \psi$
T	**T**	**T**
T	**F**	**F**
F	**T**	**F**
F	**F**	**F**

ϕ	ψ	$\phi \vee \psi$
T	**T**	**T**
T	**F**	**T**
F	**T**	**T**
F	**F**	**F**

ϕ	ψ	$\phi \rightarrow \psi$
T	**T**	**T**
T	**F**	**F**
F	**T**	**T**
F	**F**	**T**

ϕ	$\neg\phi$
T	**F**
F	**T**

$\top$
T

$\bot$
F

- 问题: 怎样利用 atomic propositions 和 logical operators 来计算 ϕ?
  - 基本方法: 使用 Bool 真值表

定义: **T** and **F**

- The set of truth values contains two elements **T** and **F**, where **T** represents 'true' and **F** represents 'false' .
- A *valuation* or *model* of a formula ϕ is an assignment of each propositional atom in ϕ to a truth value.

ϕ	ψ	$\phi \wedge \psi$
T	**T**	**T**
T	**F**	**F**
F	**T**	**F**
F	**F**	**F**

ϕ	ψ	$\phi \vee \psi$
T	**T**	**T**
T	**F**	**T**
F	**T**	**T**
F	**F**	**F**

ϕ	ψ	$\phi \to \psi$
T	**T**	**T**
T	**F**	**F**
F	**T**	**T**
F	**F**	**T**

ϕ	$\neg\phi$
T	**F**
F	**T**

$\top$
T

$\bot$
F

- 问题: 怎样利用 atomic propositions 和 logical operators 来计算 ϕ?
  - 基本方法: 使用 Bool 真值表

定义: T and F

- The set of truth values contains two elements **T** and **F**, where **T** represents 'true' and **F** represents 'false'.
- A *valuation* or *model* of a formula ϕ is an assignment of each propositional atom in ϕ to a truth value.

ϕ	ψ	$\phi \land \psi$
T	**T**	**T**
T	**F**	**F**
F	**T**	**F**
F	**F**	**F**

ϕ	ψ	$\phi \lor \psi$
T	**T**	**T**
T	**F**	**T**
F	**T**	**T**
F	**F**	**F**

ϕ	ψ	$\phi \to \psi$
T	**T**	**T**
T	**F**	**F**
F	**T**	**T**
F	**F**	**T**

ϕ	$\neg\phi$
T	**F**
F	**T**

$\top$
T

$\bot$
F

- 问题: 怎样利用 atomic propositions 和 logical operators 来计算 ϕ?
  - 基本方法: 使用 Bool 真值表

定义: **T** and **F**

- The set of truth values contains two elements **T** and **F**, where **T** represents 'true' and **F** represents 'false'.
- A *valuation* or *model* of a formula ϕ is an assignment of each propositional atom in ϕ to a truth value.

ϕ	ψ	$\phi \wedge \psi$
T	**T**	**T**
T	**F**	**F**
F	**T**	**F**
F	**F**	**F**

ϕ	ψ	$\phi \vee \psi$
T	**T**	**T**
T	**F**	**T**
F	**T**	**T**
F	**F**	**F**

ϕ	ψ	$\phi \rightarrow \psi$
T	**T**	**T**
T	**F**	**F**
F	**T**	**T**
F	**F**	**T**

ϕ	$\neg\phi$
T	**F**
F	**T**

$\top$
T

$\bot$
F

例：真值表

p	q	$\neg p$	$\neg q$	$p \to \neg q$	$q \vee \neg p$	$\phi = (p \to \neg q) \to (q \vee \neg p)$
T	T	F	F	F	T	T
T	F	F	T	T	F	F
F	T	T	F	T	T	T
F	F	T	T	T	T	T

回到主题: 对于如下 $\mathcal{M}$ 和 ϕ, 是否满足 $\mathcal{M} \models \phi$?

- $\mathcal{M} = \{p, q\}$, $\phi = (p \to \neg q) \to (q \vee \neg p)$
  - Yes, 即 $\mathcal{M} \models \phi$
- $\mathcal{M} = \{p, \neg q\}$, $\phi = (p \to \neg q) \to (q \vee \neg p)$
  - No, 即 $\mathcal{M} \not\models \phi$
- $\mathcal{M} = \{p\}$, $\phi = (p \to \neg q) \to (q \vee \neg p)$
  - 无法确定，原因：公理不完备 (建模出现问题)
- $\mathcal{M} = \{p, \neg p\}$, $\phi = (p \to \neg q) \to (q \vee \neg p)$
  - 公理存在矛盾 (建模出现问题)

例：真值表

p	q	$\neg p$	$\neg q$	$p \to \neg q$	$q \vee \neg p$	$\phi = (p \to \neg q) \to (q \vee \neg p)$
T	**T**	**F**	**F**	**F**	**T**	**T**
T	**F**	**F**	**T**	**T**	**F**	**F**
F	**T**	**T**	**F**	**T**	**T**	**T**
F	**F**	**T**	**T**	**T**	**T**	**T**

回到主题: 对于如下 $\mathcal{M}$ 和 ϕ, 是否满足 $\mathcal{M} \vDash \phi$?

- $\mathcal{M} = \{p, q\}$, $\phi = (p \to \neg q) \to (q \vee \neg p)$
  - Yes, 即 $\mathcal{M} \vDash \phi$
- $\mathcal{M} = \{p, \neg q\}$, $\phi = (p \to \neg q) \to (q \vee \neg p)$
  - No, 即 $\mathcal{M} \nvDash \phi$
- $\mathcal{M} = \{p\}$, $\phi = (p \to \neg q) \to (q \vee \neg p)$
  - 无法确定，原因：公理不完备 (建模出现问题)
- $\mathcal{M} = \{p, \neg p\}$, $\phi = (p \to \neg q) \to (q \vee \neg p)$
  - 公理存在矛盾 (建模出现问题)

例：真值表

p	q	$\neg p$	$\neg q$	$p \to \neg q$	$q \vee \neg p$	$\phi = (p \to \neg q) \to (q \vee \neg p)$
T	**T**	**F**	**F**	**F**	**T**	**T**
T	**F**	**F**	**T**	**T**	**F**	**F**
F	**T**	**T**	**F**	**T**	**T**	**T**
F	**F**	**T**	**T**	**T**	**T**	**T**

回到主题: 对于如下 $\mathcal{M}$ 和 ϕ, 是否满足 $\mathcal{M} \vDash \phi$?

- $\mathcal{M} = \{p, q\}$, $\phi = (p \to \neg q) \to (q \vee \neg p)$
  - Yes, 即 $\mathcal{M} \vDash \phi$
- $\mathcal{M} = \{p, \neg q\}$, $\phi = (p \to \neg q) \to (q \vee \neg p)$
  - No, 即 $\mathcal{M} \nvDash \phi$
- $\mathcal{M} = \{p\}$, $\phi = (p \to \neg q) \to (q \vee \neg p)$
  - 无法确定，原因：公理不完备 (建模出现问题)
- $\mathcal{M} = \{p, \neg p\}$, $\phi = (p \to \neg q) \to (q \vee \neg p)$
  - 公理存在矛盾 (建模出现问题)

例：真值表

p	q	$\neg p$	$\neg q$	$p \to \neg q$	$q \vee \neg p$	$\phi = (p \to \neg q) \to (q \vee \neg p)$
T	**T**	**F**	**F**	**F**	**T**	**T**
T	**F**	**F**	**T**	**T**	**F**	**F**
F	**T**	**T**	**F**	**T**	**T**	**T**
F	**F**	**T**	**T**	**T**	**T**	**T**

回到主题: 对于如下 $\mathcal{M}$ 和 ϕ, 是否满足 $\mathcal{M} \models \phi$?

- $\mathcal{M} = \{p, q\}$, $\phi = (p \to \neg q) \to (q \vee \neg p)$
  - Yes, 即 $\mathcal{M} \models \phi$
- $\mathcal{M} = \{p, \neg q\}$, $\phi = (p \to \neg q) \to (q \vee \neg p)$
  - No, 即 $\mathcal{M} \not\models \phi$
- $\mathcal{M} = \{p\}$, $\phi = (p \to \neg q) \to (q \vee \neg p)$
  - 无法确定，原因：公理不完备 (建模出现问题)
- $\mathcal{M} = \{p, \neg p\}$, $\phi = (p \to \neg q) \to (q \vee \neg p)$
  - 公理存在矛盾 (建模出现问题)

例：真值表

p	q	$\neg p$	$\neg q$	$p \to \neg q$	$q \vee \neg p$	$\phi = (p \to \neg q) \to (q \vee \neg p)$
T	T	F	F	F	T	T
T	F	F	T	T	F	F
F	T	T	F	T	T	T
F	F	T	T	T	T	T

回到主题: 对于如下 $\mathcal{M}$ 和 ϕ, 是否满足 $\mathcal{M} \models \phi$?

- $\mathcal{M} = \{p, q\}$, $\phi = (p \to \neg q) \to (q \vee \neg p)$
  - Yes, 即 $\mathcal{M} \models \phi$
- $\mathcal{M} = \{p, \neg q\}$, $\phi = (p \to \neg q) \to (q \vee \neg p)$
  - No, 即 $\mathcal{M} \not\models \phi$
- $\mathcal{M} = \{p\}$, $\phi = (p \to \neg q) \to (q \vee \neg p)$
  - 无法确定，原因：公理不完备 (建模出现问题)
- $\mathcal{M} = \{p, \neg p\}$, $\phi = (p \to \neg q) \to (q \vee \neg p)$
  - 公理存在矛盾 (建模出现问题)

例：真值表

p	q	$\neg p$	$\neg q$	$p \to \neg q$	$q \vee \neg p$	$\phi = (p \to \neg q) \to (q \vee \neg p)$
T	T	F	F	F	T	T
T	F	F	T	T	F	F
F	T	T	F	T	T	T
F	F	T	T	T	T	T

回到主题: 对于如下 $\mathcal{M}$ 和 ϕ, 是否满足 $\mathcal{M} \vDash \phi$?

- $\mathcal{M} = \{p, q\}$, $\phi = (p \to \neg q) \to (q \vee \neg p)$
  - Yes, 即 $\mathcal{M} \vDash \phi$
- $\mathcal{M} = \{p, \neg q\}$, $\phi = (p \to \neg q) \to (q \vee \neg p)$
  - No, 即 $\mathcal{M} \nvDash \phi$
- $\mathcal{M} = \{p\}$, $\phi = (p \to \neg q) \to (q \vee \neg p)$
  - 无法确定，原因：公理不完备 (建模出现问题)
- $\mathcal{M} = \{p, \neg p\}$, $\phi = (p \to \neg q) \to (q \vee \neg p)$
  - 公理存在矛盾 (建模出现问题)

回到主题: 对于如下 $\mathcal{M}$ 和 ϕ, 是否满足 $\mathcal{M} \vDash \phi$?

- $\mathcal{M} = \{p, q\}$, $\phi = (p \to \neg q) \to (q \vee \neg p)$
  - Yes, 即 $\mathcal{M} \vDash \phi$
- $\mathcal{M} = \{p, \neg q\}$, $\phi = (p \to \neg q) \to (q \vee \neg p)$
  - No, 即 $\mathcal{M} \vDash \neg\phi$
- $\mathcal{M} = \{p\}$, $\phi = (p \to \neg q) \to (q \vee \neg p)$
  - 无法确定，原因：公理不完备 (建模出现问题)
- $\mathcal{M} = \{p, \neg p\}$, $\phi = (p \to \neg q) \to (q \vee \neg p)$
  - 公理存在矛盾 (建模出现问题)

定义: Semantic entailment relation

If, for all valuations in which all $\phi_1, \phi_2, \ldots, \phi_n$ evaluate to **T**, ψ evaluates to **T** as well, we say that

$$\phi_1, \phi_2, \ldots, \phi_n \vDash \psi$$

holds and call $\vDash$ the *semantic* entailment relation.

回到主题: 对于如下 $\mathcal{M}$ 和 ϕ, 是否满足 $\mathcal{M} \models \phi$?

- $\mathcal{M} = \{p, q\}$, $\phi = (p \to \neg q) \to (q \vee \neg p)$
  - Yes, 即 $\mathcal{M} \models \phi$
- $\mathcal{M} = \{p, \neg q\}$, $\phi = (p \to \neg q) \to (q \vee \neg p)$
  - No, 即 $\mathcal{M} \models \neg\phi$
- $\mathcal{M} = \{p\}$, $\phi = (p \to \neg q) \to (q \vee \neg p)$
  - 无法确定，原因：公理不完备 (建模出现问题)
- $\mathcal{M} = \{p, \neg p\}$, $\phi = (p \to \neg q) \to (q \vee \neg p)$
  - 公理存在矛盾 (建模出现问题)

定义: Semantic entailment relation

If, for all valuations in which all $\phi_1, \phi_2, \ldots, \phi_n$ evaluate to **T**, ψ evaluates to **T** as well, we say that

$$\phi_1, \phi_2, \ldots, \phi_n \models \psi$$

holds and call $\models$ the *semantic* entailment relation.

例：真值表

p	q	$\neg p$	$\neg q$	$p \to \neg q$	$q \vee \neg p$	$\phi = (p \to \neg q) \to (q \vee \neg p)$
T	**T**	**F**	**F**	**F**	**T**	**T**
T	**F**	**F**	**T**	**T**	**F**	**F**
F	**T**	**T**	**F**	**T**	**T**	**T**
F	**F**	**T**	**T**	**T**	**T**	**T**

复杂度?

- 若 $\mathcal{M}$ 原子命题的个数为 n, 判定所需时间为 $O(2^n)$.
- 怎么办?

例：真值表

p	q	$\neg p$	$\neg q$	$p \to \neg q$	$q \vee \neg p$	$\phi = (p \to \neg q) \to (q \vee \neg p)$
T	**T**	**F**	**F**	**F**	**T**	**T**
T	**F**	**F**	**T**	**T**	**F**	**F**
F	**T**	**T**	**F**	**T**	**T**	**T**
F	**F**	**T**	**T**	**T**	**T**	**T**

复杂度?

- 若 $\mathcal{M}$ 原子命题的个数为 n, 判定所需时间为 $O(2^n)$.
- 怎么办?

问题: Consider the declarative sentence:

- *Every* student is younger than *some* instructor.
  - How to define when there are 1,000,000,000 students?
  - Moreover, how to specify an instructor for each student?

解决方法: Design a richer *language* (*logic*):

- Predicate Logic (谓词逻辑), a.k.a, *First-order Logic (一阶逻辑)*
  - *Inherit* Propositional Logic
  - Introduce *Predicate*
    - S(*andy*) to denote that Andy is a student.
    - I(*paul*) to say that Paul is an instructor.
    - $Y(andy, paul)$ could mean that Andy is younger than Paul.
    - The symbols S, I and Y are called *predicates*.
  - Introduce *quantifiers* $\forall$ and $\exists$
    - $\forall$: for all, $\exists$: there exists

答案: $\forall x\ (S(x) \to (\exists y\ (\ I(y) \wedge Y(x, y)\ )))$

问题: Consider the declarative sentence:

- *Every* student is younger than *some* instructor.
  - How to define when there are 1,000,000,000 students?
  - Moreover, how to specify an instructor for each student?

解决方法: Design a richer *language* (*logic*):

- Predicate Logic (谓词逻辑), a.k.a, *First-order Logic (一阶逻辑)*
  - *Inherit* Propositional Logic
  - Introduce *Predicate*
    - S(*andy*) to denote that Andy is a student.
    - I(*paul*) to say that Paul is an instructor.
    - $Y(andy, paul)$ could mean that Andy is younger than Paul.
    - The symbols S, I and Y are called *predicates*.
  - Introduce *quantifiers* $\forall$ and $\exists$
    - $\forall$: for all, $\exists$: there exists

答案: $\forall x\ (S(x) \to (\exists y\ (\ I(y) \wedge Y(x, y)\ )))$

问题: Consider the declarative sentence:

- *Every* student is younger than *some* instructor.
  - How to define when there are 1,000,000,000 students?
  - Moreover, how to specify an instructor for each student?

解决方法: Design a richer *language* (*logic*):

- Predicate Logic (谓词逻辑), a.k.a, *First-order Logic (一阶逻辑)*
  - *Inherit* Propositional Logic
  - Introduce *Predicate*
    - S(*andy*) to denote that Andy is a student.
    - I(*paul*) to say that Paul is an instructor.
    - $Y(andy, paul)$ could mean that Andy is younger than Paul.
    - The symbols S, I and Y are called *predicates*.
  - Introduce *quantifiers* $\forall$ and $\exists$
    - $\forall$: for all, $\exists$: there exists

答案: $\forall x\ (S(x) \rightarrow (\exists y\ (\ I(y) \wedge Y(x, y)\ )))$

问题: Consider the declarative sentence:

- *Every* student is younger than *some* instructor.
  - How to define when there are 1,000,000,000 students?
  - Moreover, how to specify an instructor for each student?

解决方法: Design a richer *language* (*logic*):

- Predicate Logic (谓词逻辑), a.k.a, *First-order Logic (一阶逻辑)*
  - *Inherit* Propositional Logic
  - Introduce *Predicate*
    - $S(andy)$ to denote that Andy is a student.
    - $I(paul)$ to say that Paul is an instructor.
    - $Y(andy, paul)$ could mean that Andy is younger than Paul.
    - The symbols S, I and Y are called *predicates*.
  - Introduce *quantifiers* $\forall$ and $\exists$
    - $\forall$: for all, $\exists$: there exists

答案: $\forall x\ (S(x) \to (\exists y\ (\ I(y) \wedge Y(x,y)\ )))$

问题: Consider the declarative sentence:

- *Every* student is younger than *some* instructor.
  - How to define when there are 1,000,000,000 students?
  - Moreover, how to specify an instructor for each student?

解决方法: Design a richer *language* (*logic*):

- Predicate Logic (谓词逻辑), a.k.a, *First-order Logic (一阶逻辑)*
  - *Inherit* Propositional Logic
  - Introduce *Predicate*
    - $S(andy)$ to denote that Andy is a student.
    - $I(paul)$ to say that Paul is an instructor.
    - $Y(andy, paul)$ could mean that Andy is younger than Paul.
    - The symbols S, I and Y are called *predicates*.
  - Introduce *quantifiers* $\forall$ and $\exists$
    - $\forall$: for all, $\exists$: there exists

答案: $\forall x\ (S(x) \rightarrow (\exists y\ (\ I(y) \wedge Y(x, y)\ )))$

问题: Consider the declarative sentence:

- *Every* student is younger than *some* instructor.
  - How to define when there are 1,000,000,000 students?
  - Moreover, how to specify an instructor for each student?

解决方法: Design a richer *language* (*logic*):

- Predicate Logic (谓词逻辑), a.k.a, *First-order Logic (一阶逻辑)*
  - *Inherit* Propositional Logic
  - Introduce *Predicate*
    - $S(\mathit{andy})$ to denote that Andy is a student.
    - $I(\mathit{paul})$ to say that Paul is an instructor.
    - $Y(andy, paul)$ could mean that Andy is younger than Paul.
    - The symbols S, I and Y are called *predicates*.
  - Introduce *quantifiers* $\forall$ and $\exists$
    - $\forall$: for all, $\exists$: there exists

答案: $\forall x\ (S(x) \to (\exists y\ (\ I(y) \wedge Y(x, y)\ )))$

问题: Consider the declarative sentence:

- *Every* student is younger than *some* instructor.
  - How to define when there are 1,000,000,000 students?
  - Moreover, how to specify an instructor for each student?

解决方法: Design a richer *language* (*logic*):

- Predicate Logic (谓词逻辑), a.k.a, *First-order Logic (一阶逻辑)*
  - *Inherit* Propositional Logic
  - Introduce *Predicate*
    - $S(andy)$ to denote that Andy is a student.
    - $I(paul)$ to say that Paul is an instructor.
    - $Y(andy, paul)$ could mean that Andy is younger than Paul.
    - The symbols S, I and Y are called *predicates*.
  - Introduce *quantifiers* $\forall$ and $\exists$
    - $\forall$: for all, $\exists$: there exists

答案: $\forall x\ (S(x) \rightarrow (\exists y\ (\ I(y) \wedge Y(x, y)\ )))$

问题: Consider the declarative sentence:

- *Every* student is younger than *some* instructor.
  - How to define when there are 1,000,000,000 students?
  - Moreover, how to specify an instructor for each student?

解决方法: Design a richer *language* (*logic*):

- Predicate Logic (谓词逻辑), a.k.a, *First-order Logic (一阶逻辑)*
  - *Inherit* Propositional Logic
  - Introduce *Predicate*
    - S(*andy*) to denote that Andy is a student.
    - I(*paul*) to say that Paul is an instructor.
    - $Y(andy, paul)$ could mean that Andy is younger than Paul.
    - The symbols S, I and Y are called *predicates*.
  - Introduce *quantifiers* $\forall$ and $\exists$
    - $\forall$: for all, $\exists$: there exists

答案: $\forall x\ (S(x) \rightarrow (\exists y\ (\ I(y) \wedge Y(x, y)\ )))$

问题: Consider the declarative sentence:

- *Every* student is younger than *some* instructor.
  - How to define when there are 1,000,000,000 students?
  - Moreover, how to specify an instructor for each student?

解决方法: Design a richer *language* (*logic*):

- Predicate Logic (谓词逻辑), a.k.a, *First-order Logic (一阶逻辑)*
  - *Inherit* Propositional Logic
  - Introduce *Predicate*
    - $S(\textit{andy})$ to denote that Andy is a student.
    - $I(\textit{paul})$ to say that Paul is an instructor.
    - $Y(andy, paul)$ could mean that Andy is younger than Paul.
    - The symbols S, I and Y are called *predicates*.
  - Introduce *quantifiers* $\forall$ and $\exists$
    - $\forall$: for all, $\exists$: there exists

答案: $\forall x\ (S(x) \rightarrow (\exists y\ (\ I(y) \wedge Y(x,y)\ )))$

问题: Consider the declarative sentence:

- *Every* student is younger than *some* instructor.
  - How to define when there are 1,000,000,000 students?
  - Moreover, how to specify an instructor for each student?

解决方法: Design a richer *language* (*logic*):

- Predicate Logic (谓词逻辑), a.k.a, *First-order Logic (一阶逻辑)*
  - *Inherit* Propositional Logic
  - Introduce *Predicate*
    - S(*andy*) to denote that Andy is a student.
    - I(*paul*) to say that Paul is an instructor.
    - $Y(andy, paul)$ could mean that Andy is younger than Paul.
    - The symbols S, I and Y are called *predicates*.
  - Introduce *quantifiers* $\forall$ and $\exists$
    - $\forall$: for all, $\exists$: there exists

答案: $\forall x\ (S(x) \to (\exists y\ (\ I(y) \wedge Y(x,y)\ )))$

定义: Term

- Any variable is a term.
- If $c \in \mathcal{F}$ is a nullary function, then c is a term.
- If $t_1, t_2, ..., t_n$ are terms and $f \in \mathcal{F}$ has arity $n > 0$, then $f(t_1, t_2, ..., t_n)$ is a term.
- Nothing else is a term.

定义: Term in *BNF*

$$t ::= x \mid c \mid f(t, ..., t)$$

where x ranges over a set of variables var, c over nullary function symbols in $\mathcal{F}$, and f over those elements of $\mathcal{F}$ with arity $n > 0$.

定义: Term in *BNF*

$$t ::= x \mid c \mid f(t, ..., t)$$

where x ranges over a set of variables var, c over nullary function symbols in $\mathcal{F}$, and f over those elements of $\mathcal{F}$ with arity $n > 0$.

定义: First-order Logic in *BNF*

$$\phi ::= P(t_1, t_2, \ldots, t_n) \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \rightarrow \phi) \mid (\forall x\ \phi) \mid (\exists x\ \phi)$$

where $P \in \mathcal{P}$ is a predicate symbol of arity $n \geq 1$, t_i are terms over $\mathcal{F}$ and x is a variable.

定义: Term in *BNF*

$$t ::= x \mid c \mid f(t, ..., t)$$

where x ranges over a set of variables var, c over nullary function symbols in $\mathcal{F}$, and f over those elements of $\mathcal{F}$ with arity $n > 0$.

定义: First-order Logic in *BNF*

$$\phi ::= P(t_1, t_2, \ldots, t_n) \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \rightarrow \phi) \mid (\forall x\ \phi) \mid (\exists x\ \phi)$$

where $P \in \mathcal{P}$ is a predicate symbol of arity $n \geq 1$, t_i are terms over $\mathcal{F}$ and x is a variable.

回到主题: 如何定义 $\mathcal{M}$?

定义: $\mathcal{M}$

Let $\mathcal{F}$ be a set of function symbols and $\mathcal{P}$ a set of predicate symbols, each symbol with a fixed number of required arguments. A *model* $\mathcal{M}$ of the pair $(\mathcal{F}, \mathcal{P})$ consists of the following set of data:

1. A non-empty set A, the universe of *concrete* values
2. for each nullary function symbol $f \in \mathcal{F}$, a *concrete* element $f^{\mathcal{M}}$ of A
3. for each $f \in \mathcal{F}$ with arity $n > 0$, a *concrete* function $f^{\mathcal{M}} : A^n \to A$ from A^n, the set of n-tuples over A, to A
4. for each $P \in \mathcal{P}$ with arity $n > 0$, a subset $P^{\mathcal{M}} \subseteq A^n$ of n-tuples over A.

回到主题: 如何定义 $\mathcal{M}$?

例 (自动机):

Let $\mathcal{F} \overset{\text{def}}{=} \{i\}$ and $\mathcal{P} = \{R, F\}$;

- i is a constant
- F a predicate symbol with one argument
- R a predicate symbol with two arguments

A model $\mathcal{M}$ may contain:

- A: a set of states of a computer program.
- $i^{\mathcal{M}}$: a designated initial state.
- $R^{\mathcal{M}}$: a state transition relation.
- $F^{\mathcal{M}}$: a set of final (accepting) states.

$\mathcal{M}$ 的实例:

- $A \overset{\text{def}}{=} \{a, b, c\}$
- $i^{\mathcal{M}} = a$
- $R^{\mathcal{M}} \overset{\text{def}}{=} \{(a, a), (a, b), (a, c), (b, c), (c, c)\}$
- $F^{\mathcal{M}} = \{b, c\}$.

ϕ 的实例:

- $\exists y\ R(i, y)$
- $\neg F(i)$
- $\forall x \forall y \forall z\ (R(x, y) \wedge R(x, z) \rightarrow y = z)$
- $\forall x \exists y\ R(x, y)$

回到主题: 如何定义 $\mathcal{M}$?

例 (自动机):

Let $\mathcal{F} \stackrel{\text{def}}{=} \{i\}$ and $\mathcal{P} = \{R, F\}$;

- i is a constant
- F a predicate symbol with one argument
- R a predicate symbol with two arguments

A model $\mathcal{M}$ may contain:

- A: a set of states of a computer program.
- $i^{\mathcal{M}}$: a designated initial state.
- $R^{\mathcal{M}}$: a state transition relation.
- $F^{\mathcal{M}}$: a set of final (accepting) states.

$\mathcal{M}$ 的实例:

- $A \stackrel{\text{def}}{=} \{a, b, c\}$
- $i^{\mathcal{M}} = a$
- $R^{\mathcal{M}} \stackrel{\text{def}}{=} \{(a, a), (a, b), (a, c), (b, c), (c, c)\}$
- $F^{\mathcal{M}} = \{b, c\}$.

ϕ 的实例:

- $\exists y\ R(i, y)$
- $\neg F(i)$
- $\forall x \forall y \forall z\ (R(x, y) \wedge R(x, z) \rightarrow y = z)$
- $\forall x \exists y\ R(x, y)$

回到主题: 如何定义 $\mathcal{M}$?

例 (自动机):

Let $\mathcal{F} \stackrel{\text{def}}{=} \{i\}$ and $\mathcal{P} = \{R, F\}$;

- i is a constant
- F a predicate symbol with one argument
- R a predicate symbol with two arguments

A model $\mathcal{M}$ may contain:

- A: a set of states of a computer program.
- $i^{\mathcal{M}}$: a designated initial state.
- $R^{\mathcal{M}}$: a state transition relation.
- $F^{\mathcal{M}}$: a set of final (accepting) states.

$\mathcal{M}$ 的实例:

- $A \stackrel{\text{def}}{=} \{a, b, c\}$
- $i^{\mathcal{M}} = a$
- $R^{\mathcal{M}} \stackrel{\text{def}}{=} \{(a,a), (a,b), (a,c), (b,c), (c,c)\}$
- $F^{\mathcal{M}} = \{b, c\}$.

ϕ 的实例:

- $\exists y\ R(i, y)$
- $\neg F(i)$
- $\forall x \forall y \forall z\ (R(x,y) \wedge R(x,z) \rightarrow y = z)$
- $\forall x \exists y\ R(x, y)$

回到主题: 如何定义 $\mathcal{M}$?

例 (自动机):

Let $\mathcal{F} \stackrel{\text{def}}{=} \{i\}$ and $\mathcal{P} = \{R, F\}$;

- i is a constant
- F a predicate symbol with one argument
- R a predicate symbol with two arguments

A model $\mathcal{M}$ may contain:

- A: a set of states of a computer program.
- $i^{\mathcal{M}}$: a designated initial state.
- $R^{\mathcal{M}}$: a state transition relation.
- $F^{\mathcal{M}}$: a set of final (accepting) states.

$\mathcal{M}$ 的实例:

- $A \stackrel{\text{def}}{=} \{a, b, c\}$
- $i^{\mathcal{M}} = a$
- $R^{\mathcal{M}} \stackrel{\text{def}}{=} \{(a, a), (a, b), (a, c), (b, c), (c, c)\}$
- $F^{\mathcal{M}} = \{b, c\}$.

ϕ 的实例:

- $\exists y\ R(i, y)$
- $\neg F(i)$
- $\forall x \forall y \forall z\ (R(x, y) \wedge R(x, z) \rightarrow y = z)$
- $\forall x \exists y\ R(x, y)$

回到主题: 如何定义 $\mathcal{M}$?

例 (自动机):

Let $\mathcal{F} \overset{\text{def}}{=} \{i\}$ and $\mathcal{P} = \{R, F\}$;

- i is a constant
- F a predicate symbol with one argument
- R a predicate symbol with two arguments

A model $\mathcal{M}$ may contain:

- A: a set of states of a computer program.
- $i^{\mathcal{M}}$: a designated initial state.
- $R^{\mathcal{M}}$: a state transition relation.
- $F^{\mathcal{M}}$: a set of final (accepting) states.

$\mathcal{M}$ 的实例:

- $A \overset{\text{def}}{=} \{a, b, c\}$
- $i^{\mathcal{M}} = a$
- $R^{\mathcal{M}} \overset{\text{def}}{=} \{(a, a), (a, b), (a, c), (b, c), (c, c)\}$
- $F^{\mathcal{M}} = \{b, c\}$.

ϕ 的实例:

- $\exists y\ R(i, y)$
- $\neg F(i)$
- $\forall x \forall y \forall z\ (R(x, y) \wedge R(x, z) \rightarrow y = z)$
- $\forall x \exists y\ R(x, y)$

回到主题: 如何定义 $\mathcal{M}$?

例 (自动机):

Let $\mathcal{F} \stackrel{\text{def}}{=} \{i\}$ and $\mathcal{P} = \{R, F\}$;

- i is a constant
- F a predicate symbol with one argument
- R a predicate symbol with two arguments

A model $\mathcal{M}$ may contain:

- A: a set of states of a computer program.
- $i^{\mathcal{M}}$: a designated initial state.
- $R^{\mathcal{M}}$: a state transition relation.
- $F^{\mathcal{M}}$: a set of final (accepting) states.

$\mathcal{M}$ 的实例:

- $A \stackrel{\text{def}}{=} \{a, b, c\}$
- $i^{\mathcal{M}} = a$
- $R^{\mathcal{M}} \stackrel{\text{def}}{=} \{(a, a), (a, b), (a, c), (b, c), (c, c)\}$
- $F^{\mathcal{M}} = \{b, c\}$.

ϕ 的实例:

- $\exists y\ R(i, y)$
- $\neg F(i)$
- $\forall x \forall y \forall z\ (R(x, y) \wedge R(x, z) \rightarrow y = z)$
- $\forall x \exists y\ R(x, y)$

回到主题: 如何定义 $\mathcal{M}$?

例 (自动机):

Let $\mathcal{F} \stackrel{\text{def}}{=} \{i\}$ and $\mathcal{P} = \{R, F\}$;

- i is a constant
- F a predicate symbol with one argument
- R a predicate symbol with two arguments

A model $\mathcal{M}$ may contain:

- A: a set of states of a computer program.
- $i^{\mathcal{M}}$: a designated initial state.
- $R^{\mathcal{M}}$: a state transition relation.
- $F^{\mathcal{M}}$: a set of final (accepting) states.

$\mathcal{M}$ 的实例:

- $A \stackrel{\text{def}}{=} \{a, b, c\}$
- $i^{\mathcal{M}} = a$
- $R^{\mathcal{M}} \stackrel{\text{def}}{=} \{(a, a), (a, b), (a, c), (b, c), (c, c)\}$
- $F^{\mathcal{M}} = \{b, c\}$.

ϕ 的实例:

- $\exists y\ R(i, y)$
- $\neg F(i)$
- $\forall x \forall y \forall z\ (R(x, y) \wedge R(x, z) \rightarrow y = z)$
- $\forall x \exists y\ R(x, y)$

回到主题: 如何定义 $\mathcal{M}$?

例 (自动机):

Let $\mathcal{F} \stackrel{\text{def}}{=} \{i\}$ and $\mathcal{P} = \{R, F\}$;

- i is a constant
- F a predicate symbol with one argument
- R a predicate symbol with two arguments

A model $\mathcal{M}$ may contain:

- A: a set of states of a computer program.
- $i^{\mathcal{M}}$: a designated initial state.
- $R^{\mathcal{M}}$: a state transition relation.
- $F^{\mathcal{M}}$: a set of final (accepting) states.

$\mathcal{M}$ 的实例:

- $A \stackrel{\text{def}}{=} \{a, b, c\}$
- $i^{\mathcal{M}} = a$
- $R^{\mathcal{M}} \stackrel{\text{def}}{=} \{(a, a), (a, b), (a, c), (b, c), (c, c)\}$
- $F^{\mathcal{M}} = \{b, c\}$.

ϕ 的实例:

- $\exists y \; R(i, y)$
- $\neg F(i)$
- $\forall x \forall y \forall z \; (R(x, y) \land R(x, z) \to y = z)$
- $\forall x \exists y \; R(x, y)$

回到主题: 对于给定 $\mathcal{M}$ 和 ϕ, 是否满足 $\mathcal{M} \vDash \phi$?

基本方法：类似 Propositional Logic, 枚举所有情况

1. 定义 Environment l
2. 定义 $\vDash_l$
3. 枚举 $\vDash_l$ 求解 $\vDash$

定义：Environment l

- l : var $\rightarrow A$
  - Type: from the set of variables var to A
  - A look-up table or environment for a universe A of concrete values
- $l[x \mapsto a]$
  - the look-up table
  - maps x to a and any other variable y to $l(y)$

回到主题: 对于给定 $\mathcal{M}$ 和 ϕ, 是否满足 $\mathcal{M} \vDash \phi$?

基本方法：类似 Propositional Logic, 枚举所有情况

1. 定义 Environment l
2. 定义 $\vDash_l$
3. 枚举 $\vDash_l$ 求解 $\vDash$

定义：Environment l

- l : var $\to A$
  - Type: from the set of variables var to A
  - A look-up table or environment for a universe A of concrete values
- $l[x \mapsto a]$
  - the look-up table
  - maps x to a and any other variable y to $l(y)$

回到主题: 对于给定 $\mathcal{M}$ 和 ϕ, 是否满足 $\mathcal{M} \vDash \phi$?
基本方法: 类似 Propositional Logic, 枚举所有情况

1. 定义 Environment l
2. 定义 $\vDash_l$
3. 枚举 $\vDash_l$ 求解 $\vDash$

定义: Environment l

- l : var $\to A$
  - Type: from the set of variables var to A
  - A look-up table or environment for a universe A of concrete values
- $l[x \mapsto a]$
  - the look-up table
  - maps x to a and any other variable y to $l(y)$

回到主题: 对于给定 $\mathcal{M}$ 和 ϕ, 是否满足 $\mathcal{M} \models \phi$?
基本方法: 类似 Propositional Logic, 枚举所有情况

1. 定义 Environment l
2. 定义 $\models_l$
3. 枚举 $\models_l$ 求解 $\models$

定义: Environment l

- l : var $\to A$
  - Type: from the set of variables var to A
  - A look-up table or environment for a universe A of concrete values
- $l[x \mapsto a]$
  - the look-up table
  - maps x to a and any other variable y to $l(y)$

回到主题: 对于给定 $\mathcal{M}$ 和 ϕ, 是否满足 $\mathcal{M} \vDash \phi$?
基本方法: 类似 Propositional Logic, 枚举所有情况

1. 定义 Environment l
2. 定义 $\vDash_l$
3. 枚举 $\vDash_l$ 求解 $\vDash$

定义: Environment l

- l : var $\to A$
  - Type: from the set of variables var to A
  - A look-up table or environment for a universe A of concrete values
- $l[x \mapsto a]$
  - the look-up table
  - maps x to a and any other variable y to $l(y)$

回到主题: 对于给定 $\mathcal{M}$ 和 ϕ, 是否满足 $\mathcal{M} \vDash \phi$?

基本方法: 类似 Propositional Logic, 枚举所有情况

1. 定义 Environment l
2. 定义 $\vDash_l$
3. 枚举 $\vDash_l$ 求解 $\vDash$

定义: Environment l

- l : var $\rightarrow A$
  - Type: from the set of variables var to A
  - A look-up table or environment for a universe A of concrete values
- $l[x \mapsto a]$
  - the look-up table
  - maps x to a and any other variable y to $l(y)$

回到主题: 对于给定 $\mathcal{M}$ 和 ϕ, 是否满足 $\mathcal{M} \vDash \phi$?
基本方法: 类似 Propositional Logic, 枚举所有情况

1. 定义 Environment l
2. 定义 $\vDash_l$
3. 枚举 $\vDash_l$ 求解 $\vDash$

定义: Environment l

- l : var $\to A$
    - Type: from the set of variables var to A
    - A look-up table or environment for a universe A of concrete values
- $l[x \mapsto a]$
    - the look-up table
    - maps x to a and any other variable y to $l(y)$

定义: Environment l

- l : var $\to A$
  - A look-up table or environment for a universe A of concrete values

定义 $\models_l$

Given a model $\mathcal{M}$ for a pair $(\mathcal{F}, \mathcal{P})$ and given an environment l, we define the satisfaction relation $\mathcal{M} \models_l \phi$ for each logical formula ϕ over the pair $(\mathcal{F}, \mathcal{P})$ and look-up table l by structural induction on ϕ.

If $M \models_l \phi$ holds, we say that ϕ computes to T in the model M with respect to the environment l.

定义 $\mathcal{M} \models \phi$

$\mathcal{M} \models \phi$ holds, iff for all choices of l, $\mathcal{M} \models_l \phi$

定义: Environment l

- l : var $\to A$
  - A look-up table or environment for a universe A of concrete values

定义 $\vDash_l$

Given a model $\mathcal{M}$ for a pair $(\mathcal{F}, \mathcal{P})$ and given an environment l, we define the satisfaction relation $\mathcal{M} \vDash_l \phi$ for each logical formula ϕ over the pair $(\mathcal{F}, \mathcal{P})$ and look-up table l by structural induction on ϕ.

If $M \vDash_l \phi$ holds, we say that ϕ computes to T in the model M with respect to the environment l.

定义 $\mathcal{M} \vDash \phi$

$\mathcal{M} \vDash \phi$ holds, iff for all choices of l, $\mathcal{M} \vDash_l \phi$

定义: Environment l

- l : var $\to A$
  - A look-up table or environment for a universe A of concrete values

定义 $\models_l$

Given a model $\mathcal{M}$ for a pair $(\mathcal{F}, \mathcal{P})$ and given an environment l, we define the satisfaction relation $\mathcal{M} \models_l \phi$ for each logical formula ϕ over the pair $(\mathcal{F}, \mathcal{P})$ and look-up table l by structural induction on ϕ.

If $M \models_l \phi$ holds, we say that ϕ computes to T in the model M with respect to the environment l.

定义 $\mathcal{M} \models \phi$

$\mathcal{M} \models \phi$ holds, iff for all choices of l, $\mathcal{M} \models_l \phi$

头疼的问题: 计算复杂度相比于命题逻辑的复杂度似乎更大

更头疼的问题: 先考虑可计算性?

回顾: 问题可以解么? – 问题 4

Given a set $A \subseteq S$, and $x \in S$, *whether there is a machine* that can compute whether $x \in A$.

- Define a new machine, named *Turing machine*, 图灵机.
- If yes, i.e., there is a Turing machine M for A, language A is *decidable*.
- If no, but there is a Turing machine M that can only accept s, if $s \in A$, language A is *still* *Turing-recognizable*.

定理 (*Undecidability* in First-order logic)

The decision problem of validity in predicate logic is *undecidable*: no program exists which, given any ϕ, decides whether $\vDash \phi$.

头疼的问题: 计算复杂度相比于命题逻辑的复杂度似乎更大
更头疼的问题: 先考虑可计算性?

回顾: 问题可以解么? – 问题 4

Given a set $A \subseteq S$, and $x \in S$, *whether there is a machine* that can compute whether $x \in A$.

- Define a new machine, named *Turing machine, 图灵机*.
- If yes, i.e., there is a Turing machine M for A, language A is *decidable*.
- If no, but there is a Turing machine M that can only accept s, if $s \in A$, language A is *still Turing-recognizable*.

定理（*Undecidability* in First-order logic）

The decision problem of validity in predicate logic is *undecidable*: no program exists which, given any ϕ, decides whether $\vDash \phi$.

头疼的问题: 计算复杂度相比于命题逻辑的复杂度似乎更大
更头疼的问题: 先考虑可计算性?

回顾: 问题可以解么? – 问题 4

Given a set $A \subseteq S$, and $x \in S$, *whether there is a machine* that can compute whether $x \in A$.

- Define a new machine, named *Turing machine*, 图灵机.
- If yes, i.e., there is a Turing machine M for A, language A is *decidable*.
- If no, but there is a Turing machine M that can only accept s, if $s \in A$, language A is *still* *Turing-recognizable*.

定理 (*Undecidability* in First-order logic)

The decision problem of validity in predicate logic is *undecidable*: no program exists which, given any ϕ, decides whether $\models \phi$.

头疼的问题: 计算复杂度相比于命题逻辑的复杂度似乎更大
更头疼的问题: 先考虑可计算性?

回顾: 问题可以解么? – 问题 4

Given a set $A \subseteq S$, and $x \in S$, *whether there is a machine* that can compute whether $x \in A$.

- Define a new machine, named *Turing machine,* 图灵机.
- If yes, i.e., there is a Turing machine M for A, language A is *decidable*.
- If no, but there is a Turing machine M that can only accept s, if $s \in A$, language A is *still* *Turing-recognizable*.

定理（*Undecidability* in First-order logic）

The decision problem of validity in predicate logic is *undecidable*: no program exists which, given any ϕ, decides whether $\vDash \phi$.

定义 $\mathcal{M}$ 和 ϕ? Logics

另一个问题: First-order Logic 的表达能力?

- 能表达所有问题么?

解答: 考虑一个反例——有向图 (directed graph) 的建模

- Software models, design standards, and execution models of hardware or programs often are described in terms of directed graphs.

反例:

Given a set of states $A = \{s_0, s_1, s_2, s_3\}$, let $R^{\mathcal{M}}$ be the set $\{(s_0, s_1), (s_1, s_0), (s_1, s_1), (s_1, s_2), (s_2, s_0), (s_3, s_0), (s_3, s_2)\}$. We may depict this model as *a directed graph* in a figure, where an edge (a transition) leads from a node s to a node s' iff $(s, s') \in R^{\mathcal{M}}$.

另一个问题: First-order Logic 的表达能力?

- 能表达所有问题么?

解答: 考虑一个反例——有向图 (directed graph) 的建模

- Software models, design standards, and execution models of hardware or programs often are described in terms of directed graphs.

反例:

Given a set of states $A = \{s_0, s_1, s_2, s_3\}$, let $R^{\mathcal{M}}$ be the set $\{(s_0, s_1), (s_1, s_0), (s_1, s_1), (s_1, s_2), (s_2, s_0), (s_3, s_0), (s_3, s_2)\}$. We may depict this model as *a directed graph* in a figure, where an edge (a transition) leads from a node s to a node s' iff $(s, s') \in R^{\mathcal{M}}$.

另一个问题: First-order Logic 的表达能力?

- 能表达所有问题么?

解答: 考虑一个反例——有向图 (directed graph) 的建模

- Software models, design standards, and execution models of hardware or programs often are described in terms of directed graphs.

反例:

Given a set of states $A = \{s_0, s_1, s_2, s_3\}$, let $R^{\mathcal{M}}$ be the set $\{(s_0, s_1), (s_1, s_0), (s_1, s_1), (s_1, s_2), (s_2, s_0), (s_3, s_0), (s_3, s_2)\}$. We may depict this model as *a directed graph* in a figure, where an edge (a transition) leads from a node s to a node s' iff $(s, s') \in R^{\mathcal{M}}$.

反例:

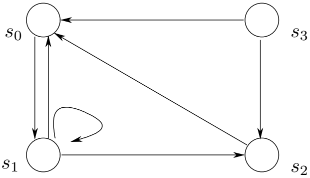
Given a set of states $A = \{s_0, s_1, s_2, s_3\}$, let $R^{\mathcal{M}}$ be the set $\{(s_0, s_1), (s_1, s_0), (s_1, s_1), (s_1, s_2), (s_2, s_0), (s_3, s_0), (s_3, s_2)\}$. We may depict this model as *a directed graph* in a figure, where an edge (a transition) leads from a node s to a node s' iff $(s, s') \in R^{\mathcal{M}}$.

反例：How to define Reachability as ϕ

Given nodes n and n' in a directed graph, is there a finite path of transitions from n to n'?

反例:

Given a set of states $A = \{s_0, s_1, s_2, s_3\}$, let $R^{\mathcal{M}}$ be the set $\{(s_0, s_1), (s_1, s_0), (s_1, s_1), (s_1, s_2), (s_2, s_0), (s_3, s_0), (s_3, s_2)\}$. We may depict this model as *a directed graph* in a figure, where an edge (a transition) leads from a node s to a node s' iff $(s, s') \in R^{\mathcal{M}}$.



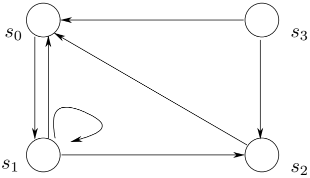
反例：How to define Reachability as ϕ

Given nodes n and n' in a directed graph, is there a finite path of transitions from n to n'?

定义 $\mathcal{M}$ 和 ϕ? Logics

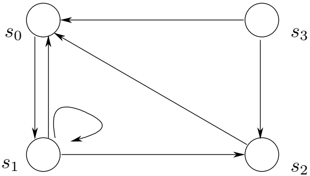
反例:

Given a set of states $A = \{s_0, s_1, s_2, s_3\}$, let $R^{\mathcal{M}}$ be the set $\{(s_0, s_1), (s_1, s_0), (s_1, s_1), (s_1, s_2), (s_2, s_0), (s_3, s_0), (s_3, s_2)\}$. We may depict this model as *a directed graph* in a figure, where an edge (a transition) leads from a node s to a node s' iff $(s, s') \in R^{\mathcal{M}}$.



反例：How to define Reachability as ϕ

Given nodes n and n' in a directed graph, is there a finite path of transitions from n to n'?



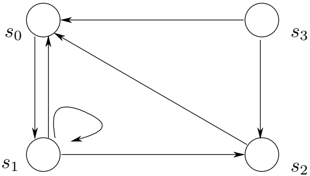
反例：How to define Reachability as ϕ

Given nodes n and n' in a directed graph, is there a finite path of transitions from n to n'?

反例：一种答案

$$(u = v) \vee \exists x (R(u, x) \wedge R(x, v)) \vee \exists x_1 \exists x_2 (R(u, x_1) \wedge R(x_1, x_2) \wedge R(x_2, v)) \vee \ldots$$

- This is infinite, so it' s *not* a well-formed formula.
- Can we find a well-formed formula with the same meaning? *No!*



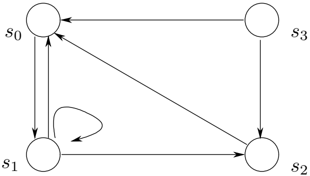
反例：How to define Reachability as ϕ

Given nodes n and n' in a directed graph, is there a finite path of transitions from n to n'?

反例：一种答案

$$(u = v) \vee \exists x (R(u,x) \wedge R(x,v)) \vee \exists x_1 \exists x_2 (R(u,x_1) \wedge R(x_1,x_2) \wedge R(x_2,v)) \vee \ldots$$

- This is infinite, so it' s *not* a well-formed formula.
- Can we find a well-formed formula with the same meaning? *No!*



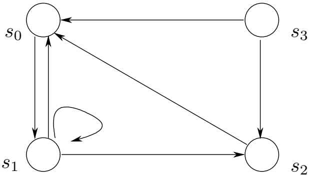
反例：How to define Reachability as ϕ

Given nodes n and n' in a directed graph, is there a finite path of transitions from n to n'?

反例：一种答案

$$(u = v) \vee \exists x (R(u,x) \wedge R(x,v)) \vee \exists x_1 \exists x_2 (R(u,x_1) \wedge R(x_1,x_2) \wedge R(x_2,v)) \vee \ldots$$

- This is infinite, so it' s *not* a well-formed formula.
- Can we find a well-formed formula with the same meaning? *No!*



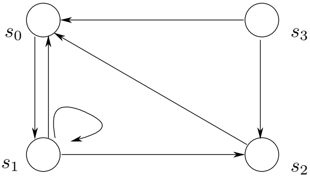
反例：How to define Reachability as ϕ

Given nodes n and n' in a directed graph, is there a finite path of transitions from n to n'?

反例：一种答案

$$(u = v) \vee \exists x(R(u, x) \wedge R(x, v)) \vee \exists x_1 \exists x_2(R(u, x_1) \wedge R(x_1, x_2) \wedge R(x_2, v)) \vee \ldots$$

- This is infinite, so it' s *not* a well-formed formula.
- Can we find a well-formed formula with the same meaning? *No!*



反例：How to define Reachability as ϕ

Given nodes n and n' in a directed graph, is there a finite path of transitions from n to n'?

反例：一种答案

$$(u = v) \vee \exists x(R(u, x) \wedge R(x, v)) \vee \exists x_1 \exists x_2(R(u, x_1) \wedge R(x_1, x_2) \wedge R(x_2, v)) \vee \ldots$$

- This is infinite, so it' s *not* a well-formed formula.
- Can we find a well-formed formula with the same meaning? *No!*

进一步问题: 既然 First-order Logic 不能表达 ϕ, 那怎么表达

- 使用 Second-order Logic
- 怎么用?
  - This can be realized by applying *quantifiers* not only to variables, but also to *predicate symbols*.

回顾: 定义: First-order Logic in *BNF*

$$\phi ::= P(t_1, t_2, \ldots, t_n) \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \rightarrow \phi) \mid (\forall x\ \phi) \mid (\exists x\ \phi)$$

where $P \in \mathcal{P}$ is a predicate symbol of arity $n \geq 1$, t_i are terms over $\mathcal{F}$ and x is a variable.

进一步问题: 既然 First-order Logic 不能表达 ϕ，那怎么表达

- 使用 Second-order Logic
- 怎么用?
  - This can be realized by applying *quantifiers* not only to variables, but also to *predicate symbols*.

回顾: 定义: First-order Logic in *BNF*

$$\phi ::= P(t_1, t_2, \ldots, t_n) \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \rightarrow \phi) \mid (\forall x\ \phi) \mid (\exists x\ \phi)$$

where $P \in \mathcal{P}$ is a predicate symbol of arity $n \geq 1$, t_i are terms over $\mathcal{F}$ and x is a variable.

进一步问题: 既然 First-order Logic 不能表达 ϕ，那怎么表达

- 使用 Second-order Logic
- 怎么用?
  - This can be realized by applying *quantifiers* not only to variables, but also to *predicate symbols*.

回顾: 定义: First-order Logic in *BNF*

$$\phi ::= P(t_1, t_2, \ldots, t_n) \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \rightarrow \phi) \mid (\forall x\ \phi) \mid (\exists x\ \phi)$$

where $P \in \mathcal{P}$ is a predicate symbol of arity $n \geq 1$, t_i are terms over $\mathcal{F}$ and x is a variable.

进一步问题: 既然 First-order Logic 不能表达 ϕ，那怎么表达

- 使用 Second-order Logic
- 怎么用?
  - This can be realized by applying *quantifiers* not only to variables, but also to *predicate symbols*.

回顾: 定义: First-order Logic in *BNF*

$$\phi ::= P(t_1, t_2, \ldots, t_n) \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \to \phi) \mid (\forall x\ \phi) \mid (\exists x\ \phi)$$

where $P \in \mathcal{P}$ is a predicate symbol of arity $n \geq 1$, t_i are terms over $\mathcal{F}$ and x is a variable.

回顾: 定义: First-order Logic in *BNF*

$$\phi ::= P(t_1, t_2, \ldots, t_n) \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \to \phi) \mid (\forall x\ \phi) \mid (\exists x\ \phi)$$

where $P \in \mathcal{P}$ is a predicate symbol of arity $n \geq 1$, t_i are terms over $\mathcal{F}$ and x is a variable.

解决思路

For a predicate symbol P with $n \geq 1$ arguments, consider formulas of the form:

$$\exists P\ \phi$$

where ϕ is a formula of predicate logic in which P *occurs*.

回顾: 定义: First-order Logic in *BNF*

$$\phi ::= P(t_1, t_2, \ldots, t_n) \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \rightarrow \phi) \mid (\forall x\ \phi) \mid (\exists x\ \phi)$$

where $P \in \mathcal{P}$ is a predicate symbol of arity $n \geq 1$, t_i are terms over $\mathcal{F}$ and x is a variable.

解决思路

For a predicate symbol P with $n \geq 1$ arguments, consider formulas of the form:

$$\exists P\ \phi$$

where ϕ is a formula of predicate logic in which P *occurs*.

反例：一种答案 First-order Logic (Not well-formed)

$$(u = v) \vee \exists x (R(u, x) \wedge R(x, v)) \vee \exists x_1 \exists x_2 (R(u, x_1) \wedge R(x_1, x_2) \wedge R(x_2, v)) \vee \ldots$$

具体答案: Second-order Logic

$$\neg \exists P \forall x \forall y \forall z \ (C_1 \wedge C_2 \wedge C_3 \wedge C_4)$$

where

$$C_1 \stackrel{\text{def}}{=} P(x, x)$$

$$C_2 \stackrel{\text{def}}{=} P(x, y) \wedge P(y, z) \to P(x, z)$$

$$C_3 \stackrel{\text{def}}{=} P(u, v) \to \bot$$

$$C_4 \stackrel{\text{def}}{=} R(x, y) \to P(x, y)$$

反例：一种答案 First-order Logic (Not well-formed)

$(u = v) \vee \exists x (R(u,x) \wedge R(x,v)) \vee \exists x_1 \exists x_2 (R(u,x_1) \wedge R(x_1,x_2) \wedge R(x_2,v)) \vee \ldots$

具体答案: Second-order Logic

$$\neg \exists P \forall x \forall y \forall z \ (C_1 \wedge C_2 \wedge C_3 \wedge C_4)$$

where

$$C_1 \overset{\text{def}}{=} P(x,x)$$

$$C_2 \overset{\text{def}}{=} P(x,y) \wedge P(y,z) \to P(x,z)$$

$$C_3 \overset{\text{def}}{=} P(u,v) \to \bot$$

$$C_4 \overset{\text{def}}{=} R(x,y) \to P(x,y)$$

下一个问题: 有没有 Third-order Logic, Fourth-order Logic,...?

答案: 有

- First-order logic quantifies only variables that range over individuals
- Second-order logic, in addition, also quantifies *over sets*
  - e.g., we can define $P(x, y) \stackrel{\text{def}}{=} (x, y) \in \mathbf{P}$, where $\mathbf{P}$ is a set.
- Third-order logic also quantifies over sets of sets, and so on.

Higher-order logic is the union of first-, second-, third-, ..., nth-order logic

- i.e., higher-order logic admits quantification over sets that are nested arbitrarily deeply.

下一个问题: 有没有 Third-order Logic, Fourth-order Logic,...?
答案: 有

- First-order logic quantifies only variables that range over individuals
- Second-order logic, in addition, also quantifies *over sets*
  - e.g., we can define $P(x, y) \stackrel{\text{def}}{=} (x, y) \in \mathbf{P}$, where $\mathbf{P}$ is a set.
- Third-order logic also quantifies over sets of sets, and so on.

Higher-order logic is the union of first-, second-, third-, ..., nth-order logic

- i.e., higher-order logic admits quantification over sets that are nested arbitrarily deeply.

下一个问题: 有没有 Third-order Logic, Fourth-order Logic,...?
答案: 有

- First-order logic quantifies only variables that range over individuals
- Second-order logic, in addition, also quantifies *over sets*
  - e.g., we can define $P(x, y) \stackrel{\text{def}}{=} (x, y) \in \mathbf{P}$, where $\mathbf{P}$ is a set.
- Third-order logic also quantifies over sets of sets, and so on.

Higher-order logic is the union of first-, second-, third-, ..., nth-order logic

- i.e., higher-order logic admits quantification over sets that are nested arbitrarily deeply.

下一个问题: 有没有 Third-order Logic, Fourth-order Logic,...?
答案: 有

- First-order logic quantifies only variables that range over individuals
- Second-order logic, in addition, also quantifies *over sets*
    - e.g., we can define $P(x, y) \stackrel{\text{def}}{=} (x, y) \in \mathbf{P}$, where $\mathbf{P}$ is a set.
- Third-order logic also quantifies over sets of sets, and so on.

Higher-order logic is the union of first-, second-, third-, ..., nth-order logic

- i.e., higher-order logic admits quantification over sets that are nested arbitrarily deeply.

下一个问题: 有没有 Third-order Logic, Fourth-order Logic,...?
答案: 有

- First-order logic quantifies only variables that range over individuals
- Second-order logic, in addition, also quantifies *over sets*
  - e.g., we can define $P(x, y) \stackrel{\text{def}}{=} (x, y) \in \mathbf{P}$, where $\mathbf{P}$ is a set.
- Third-order logic also quantifies over sets of sets, and so on.

Higher-order logic is the union of first-, second-, third-, ..., nth-order logic

- i.e., higher-order logic admits quantification over sets that are nested arbitrarily deeply.

下一个问题: 有没有 Third-order Logic, Fourth-order Logic,...?
答案: 有

- First-order logic quantifies only variables that range over individuals
- Second-order logic, in addition, also quantifies *over sets*
  - e.g., we can define $P(x, y) \stackrel{\text{def}}{=} (x, y) \in \mathbf{P}$, where $\mathbf{P}$ is a set.
- Third-order logic also quantifies over sets of sets, and so on.

Higher-order logic is the union of first-, second-, third-, ..., nth-order logic

- i.e., higher-order logic admits quantification over sets that are nested arbitrarily deeply.

下一个问题: 有没有 Third-order Logic, Fourth-order Logic,...?
答案: 有

- First-order logic quantifies only variables that range over individuals
- Second-order logic, in addition, also quantifies *over sets*
    - e.g., we can define $P(x, y) \stackrel{\text{def}}{=} (x, y) \in \mathbf{P}$, where $\mathbf{P}$ is a set.
- Third-order logic also quantifies over sets of sets, and so on.

Higher-order logic is the union of first-, second-, third-, ..., nth-order logic

- i.e., higher-order logic admits quantification over sets that are nested arbitrarily deeply.

作业

a. Compute the complete truth table of the formula:

1. $((p \to q) \to p) \to p$
2. $(p \wedge q) \to (p \vee q)$
3. $(p \to q) \vee (p \to \neg q)$
4. $((p \vee q) \to r) \to ((p \to r) \vee (q \to r))$

作业

1. Use the predicates

$$\begin{aligned} A(x,y): &\quad x \text{ admires } y \\ B(x,y): &\quad x \text{ attended } y \\ P(x): &\quad x \text{ is a professor} \\ S(x): &\quad x \text{ is a student} \\ L(x): &\quad x \text{ is a lecture} \end{aligned}$$

and the nullary function symbol (constant)

$$m: \quad \text{Mary}$$

to translate the following into predicate logic:

(a) Mary admires every professor.
(The answer is not $\forall x\, A(m, P(x))$.)

(b) Some professor admires Mary.

(c) Mary admires herself.

(d) No student attended every lecture.

(e) No lecture was attended by every student.

(f) No lecture was attended by any student.

作业

2. Consider the sentence $\phi \stackrel{\text{def}}{=} \forall x\, \exists y\, \exists z\, (P(x,y) \wedge P(z,y) \wedge (P(x,z) \rightarrow P(z,x)))$. Which of the following models satisfies ϕ?
   (a) The model $\mathcal{M}$ consists of the set of natural numbers with $P^{\mathcal{M}} \stackrel{\text{def}}{=} \{(m,n) \mid m < n\}$.
   (b) The model $\mathcal{M}'$ consists of the set of natural numbers with $P^{\mathcal{M}'} \stackrel{\text{def}}{=} \{(m, 2*m) \mid m \text{ natural number}\}$.
   (c) The model $\mathcal{M}''$ consists of the set of natural numbers with $P^{\mathcal{M}''} \stackrel{\text{def}}{=} \{(m,n) \mid m < n+1\}$.