Chapter 8 First-order Logic

8.1 Representation revisited

8.2 Syntax and Semantics of FOL

8.3 Using FOL

8.4 Knowledge Engineering in FOL

一阶逻辑

对象，函数，关系
变元，常元，函词，谓词，量词，等词，连接词；
解释，模型；项，原子语句，复合语句

1. 对象、函数、关系
2. 模型、解释
3. 项、语句、量词、等词
4. 使用一阶逻辑

对象、函数、关系

review: Chapter 7

- Logical agents apply inference to a knowledge base to derive new information and make decisions

- Basic concepts of logic
  - syntax: formal structure of sentences
  - semantics: truth of sentences wrt(关于) models

review: Chapter 7

- Propositional logic lacks expressive power
  - can’t say "pits cause breezes in adjacent squares”
  - except by writing one sentence for each square

Wumpus world using propositional logic{8.1}

$\neg P_{1,1}$、 $\neg W_{1,1}$ 、 $\neg B_{1,1}$、 $\neg S_{1,1}$

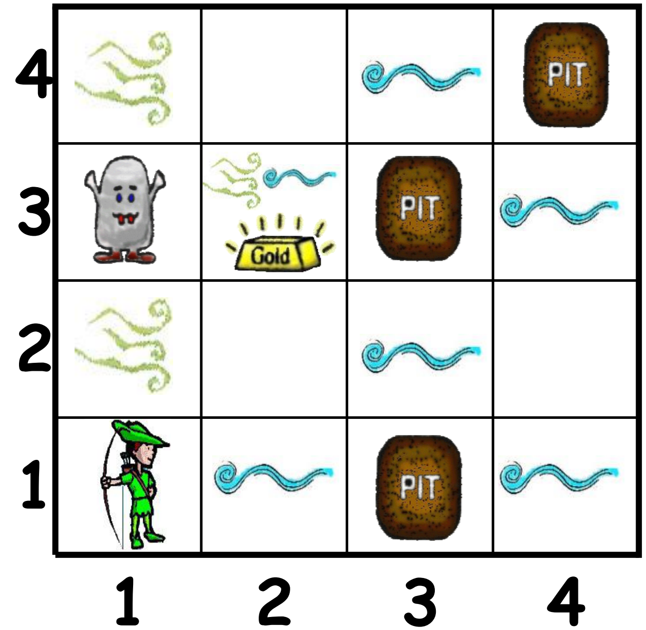
$B_{2,1}$ 、 $\neg S_{2,1}$

$B_{2,1} \Leftrightarrow P_{1,1} \vee P_{2,2} \vee P_{3,1}$

$S_{2,1} \Leftrightarrow W_{1,1} \vee W_{2,2} \vee W_{3,1}$

$P_{2,2} \vee P_{3,1}$ 、 $\neg W_{2,2}$、 $\neg W_{3,1}$

…



"pits cause breezes in adjacent squares" needs 16 PL sentences, such as $B_{2,1} \Leftrightarrow P_{1,1} \vee P_{2,2} \vee P_{3,1}$.

many distinct proposition symbols, many sentences

How does PL say these? {8.1}

- In PL (Propositional Logic), in order to say
  - The adjacent squares of pit are breezy.
  - All students are smart.
  - Some students work hard.
  - Hardworking students have good marks.
  - If someone is someone else's grandfather, then someone else has a grandson. …
- we have to _______________
- With FOL (First-order Logic), we can say each of these situations with one sentence.

How does PL say these? {8.1}

- In PL (Propositional Logic), in order to say
  - The adjacent squares of pit are breezy.
  - All students are smart.
  - Some students work hard.
  - Hardworking students have good marks.
  - If someone is someone else's grandfather, then someone else has a grandson. …
- we have to enumerate …

- With FOL (First-order Logic), we can say each of these situations with one sentence.

FOL: objects, relations, functions {8.1}

- First-order logic assumes the world contains
  - Objects: people, numbers, games, wars, …
  - Relations: red, prime, bigger than, part of, …
  - Functions: father of, one's best friend, plus, …
- "One plus two equals three"

  Objects: ____________________

  Relations: ____________________

  Functions: ____________________
- "Squares neighboring the Wumpus are smelly"

  Objects: ____________________

  Relations: ____________________

  Functions: ____________________

FOL: objects, relations, functions {8.1}

- First-order logic assumes the world contains
  - Objects: people, numbers, games, wars, …
  - Relations: red, prime, bigger than, part of, …
  - Functions: father of, one's best friend, plus, …
- "One plus two equals three"

  Objects: one, two, three, one plus two

  Relations: equals

  Functions: plus
- "Squares neighboring the Wumpus are smelly"

  Objects: Wumpus, square

  Relations: neighboring, smelly (properties)

  Functions: --

模型、解释

review: models for PL {8.2.1}

- Each model links the vocabulary of the logical sentences to elements of the possible world, so that the truth of any sentence can be determined.
- What are the vocabularies for PL sentences?
  - ______________________
- What are the elements of the possible world for PL?
  - ______________________

review: models for PL {8.2.1}

- Each model links the vocabulary of the logical sentences to elements of the possible world, so that the truth of any sentence can be determined.
- What are the vocabularies for PL sentences?
  - Proposition symbols, such as P, Q, …
- What are the elements of the possible world for PL?
  - truth values: T, F

models for FOL{8.2.1}

- models for PL link proposition symbols to predefined truth values.

vocabularies of PL: P,Q,… $\xrightarrow{\text{model for PL}}$ elements of the possible world: T,F.

models for FOL{8.2.1}

- What are models for FOL?
  - what are the elements of the possible world?
  - what are the vocabularies of FOL sentences?
  - how to establish link between them?

models for FOL{8.2.1}

- models for FOL
  - have objects
  - The domain of a model is the nonempty set of objects
  - it doesn't matter *what* these objects are — all that matters is *how many* there are in each particular model
  - The objects in the model may be *related* in various ways. a relation is the set of tuples of objects that are related

symbols {8.2.2}

- Constants(个体常元)
  - a, b, 2, NUDT,...
- Variables(个体变元)
  - x, y,...
- Functions(函词)
  - Sqrt, LeftLegOf, f, g, h,...
- Predicates (谓词)
  - Brother, P, Q, R, ...

- Connectives(连接词)
  - $\neg$, $\Rightarrow$, $\wedge$, $\vee$, $\Leftrightarrow$
- Quantifiers(量词)
  - $\forall$, $\exists$

symbols {8.2.2}

- there is a correspondence between
  - functions, which return values
  - predicates, which are true or false

Function: father_of(Mary) = Bill

Predicate: father_of(Mary, Bill)

interpretations {8.2.2}

- Model in FOL consists of a set of objects and an interpretation that maps
  - constant symbols → objects
  - predicate symbols → relations on objects
  - function symbols → functions on objects
  - An atomic sentence $predicate(term_1,...,term_n)$ is true iff the objects referred to by $term_1,...,term_n$ are in the relation referred to by *predicate*

interpretations {8.2.2}

- Suppose there are two objects, a and b, in the domain. There is a relation mother={<a,b>}. How many possible models are there for FOL sentence mother(sucie, bob)?

interpretations {8.2.2}

- 考虑某知识库只包括两条语句$P(a)$和$P(b)$。此知识库是否蕴涵$\forall x\ P(x)$？

interpretations {8.2.2}

- 模型检验在一阶谓词逻辑中是否可行？为什么？

interpretations {8.2.2}

- 如命题逻辑一样，蕴涵、有效性等都根据*所有可能模型*来定义。由于可能模型的数量是无限的，通过枚举所有可能模型以检验蕴涵在一阶逻辑中是不可行的。即使对象数量有限，各种组合的数量仍然可能非常大。图8.4，如果有6个或更少对象，语句有2个常量、1个二元关系，会有137,506,194,466个模型。

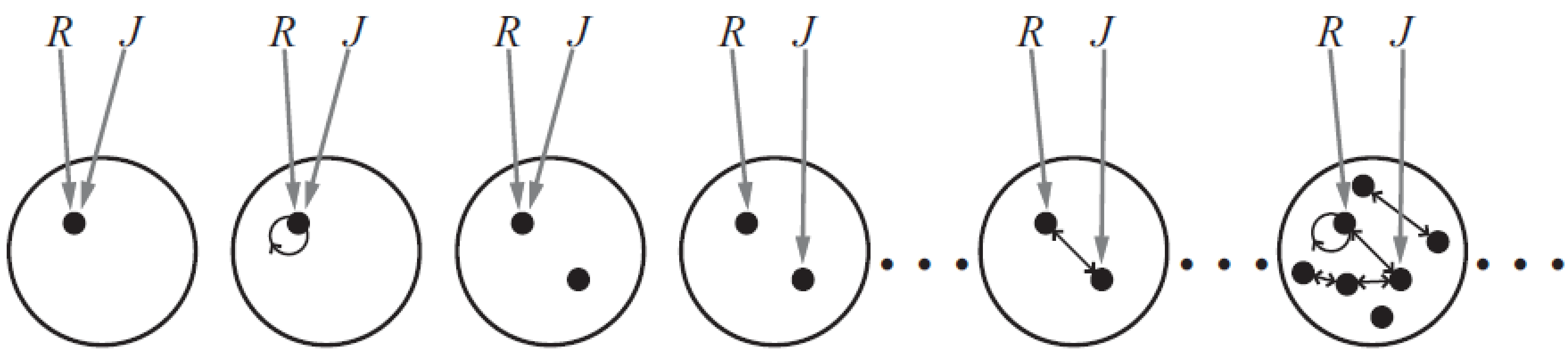


Figure 8.4 Some members of the set of all models for a language with two constant symbols, R and J, and one binary relation symbol. The interpretation of each constant symbol is shown by a gray arrow. Within each model, the related objects are connected by arrows.

项、语句、量词、等词

term, Sentences{8.2.3-5}

- Term (项)
  - Constant symbol, function symbol, or variable
  - At(x,CS)： x is a variable, CS is a constant.

- atomic sentence (原子语句)
  - Predicate symbol with value true or false
  - Represents a relation between terms
  - At(x,CS) is an atom.

- complex sentence (复合语句)
  - Atom(s) joined together using logical connectives and/or quantifiers

Quantifiers {8.2.6}

- Expressing sentences about *collections* of objects without enumeration (naming individuals)
  - All Computer Science (CS) students are clever.
  - Someone in the class is sleeping.

- Universal quantification (for all): ∀
  - ∀<*variables*> <*sentence*>
- Existential quantification (there exists): ∃
  - ∃ <*variables*> <*sentence*>

Quantifiers {8.2.6}

- examples:
  - All Computer Science (CS) students are clever.
  - ______________________________
  - Someone in the class is sleeping.
  - ______________________________

Quantifiers {8.2.6}

- examples :
  - All Computer Science (CS) students are clever.
  - $\forall x\ (At(x,CS) \Rightarrow Smart(x))$
  - Someone in the class is sleeping.
  - $\exists\ x\ (Inclass(x) \land\ Sleeping(x))$

Universal quantification {8.2.6}

- $\forall$<*variables*> <*sentence*>
  - Everyone at School of Computer Science is smart.
  - $\forall x\ (At(x, CS) \Rightarrow Smart(x)))$
- $\forall x$ *P* is true in a model *m* iff *P* is true with *x* being each possible object in the model
- Suppose the object set is {John, Micheal},

  $$\forall x\ (At(x, CS) \Rightarrow Smart(x)))$$

  is equivalent to ________________________________

Universal quantification {8.2.6}

- $\forall$*<variables> <sentence>*
  - Everyone at School of Computer Science is smart.
  - $\forall x\ (At(x, CS) \Rightarrow Smart(x)))$

- $\forall x$ *P* is true in a model *m* iff *P* is true with *x* being each possible object in the model

- Suppose the object set is {John, Micheal},

  $$\forall x\ (At(x, CS) \Rightarrow Smart(x)))$$

  is equivalent to the conjunction of instantiations of P

  $$(At(John, CS) \Rightarrow Smart(John))$$
  $$\wedge\ (At(Micheal, CS) \Rightarrow Smart(Micheal))$$

Quantifiers {8.2.6}

- Everyone at School of Computer Science is smart.
  - $\forall x\ (At(x,CS) \land Smart(x))$ correct?
- Someone at School of Physics is not smart.
  - $\exists\ x\ (At(x,Phy) \Rightarrow \neg\ Smart(x))$ correct?

Quantifiers {8.2.6}

- Everyone at School of Computer Science is smart.
  - $\forall x\ (At(x,CS) \wedge Smart(x))$ incorrect

- Someone at School of Physics is not smart.
  - $\exists\ x\ (At(x,Phy) \Rightarrow \neg\ Smart(x))$ incorrect

A common mistake to avoid {8.2.6}

- Typically, $\Rightarrow$ is the main connective with $\forall$
- Common mistake: using $\wedge$ as the main connective with $\forall$:
  - $\forall x\ (At(x,CS) \wedge Smart(x))$
  - means "Everyone is at CS and everyone is smart"

- Typically, $\wedge$ is the main connective with $\exists$
- Common mistake: using $\Rightarrow$ as the main connective with $\exists$.

Properties of quantifiers {8.2.6}

- $\forall x\ \forall y$ is the same as $\forall y\ \forall x$ correct?
- $\exists x\ \exists y$ is the same as $\exists y\ \exists x$ correct?
- $\exists x\ \forall y$ is the same as $\forall y\ \exists x$ correct?
  - $\forall y\ \exists x$ Mother(x,y)
  - $\exists x\ \forall y$ Mother(x,y)

- Quantifier duality: each can be expressed using the other
  - $(\forall x)$Likes(x, IceCream) ______________________
  - $(\exists x)$Likes(x, Cheese) ______________________

Properties of quantifiers {8.2.6}

- $\forall x\ \forall y$ is the same as $\forall y\ \forall x$
- $\exists x\ \exists y$ is the same as $\exists y\ \exists x$
- $\exists x\ \forall y$ is *not* the same as $\forall y\ \exists x$
  - $\forall y\ \exists x$ Mother(x,y) Everyone has mother.
  - $\exists x\ \forall y$ Mother(x,y) There is a person who is everyone's mother.
- Quantifier duality: each can be expressed using the other
  - $(\forall x)$Likes(x, IceCream) $\neg\exists x(\neg$Likes(x, IceCream))
  - $(\exists x)$Likes(x, Cheese) $\neg\forall x(\neg$Likes(x, Cheese))

equality {8.2.7}

- use the equality symbol to signify that two terms refer to the same object. For example,
  Father (John)=Henry
- To say that Richard has at least two brothers, we would write

- The notation $x \neq y$ is sometimes used as an abbreviation for $\neg(x=y)$.

equality {8.2.7}

- use the equality symbol to signify that two terms refer to the same object. For example,
  Father (John)=Henry
- To say that Richard has at least two brothers, we would write
  $\exists$ x, y Brother (x,Richard) $\wedge$ Brother (y,Richard) $\wedge \neg$(x=y)
- The notation x $\neq$ y is sometimes used as an abbreviation for $\neg$(x=y).

equality {8.2.7}

- *Brother*(*John*, *Richard*) ∧*Brother*(*Geoffrey*, *Richard*)
- 能否表达“Richard有两个兄弟John和Geoffrey”？

equality {8.2.7}

- $Brother(John, Richard) \land Brother(Geoffrey, Richard)$
- 能否表达“Richard有两个兄弟John和Geoffrey”？
- 正确翻译：
- $Brother(John, Richard) \land Brother(Geoffrey, Richard) \land John \neq Geoffrey \land \forall x\ Brother(x, Richard) \Rightarrow (x = John \lor x = Geoffrey)$.

- 这（标准的一阶逻辑语义）比自然语言表述累赘得多。在将知识翻译成一阶逻辑的时候直观上也很容易出错。

- 能否设计一种语义使得逻辑表达更直接呢？

an alternative semantics {8.2.8}

- 数据库语义
  - **关键字假设**：坚持每个常量符号指代一个确定对象。
  - **封闭世界假设**：假设我们不知道的所有原子语句事实上都为假。
  - **论域闭包**：每个模型只包括常量符号指代的对象。
- **数据库语义**区分于标准的一阶逻辑语义
- *Brother*(*John*, *Richard*) ∧*Brother*(*Geoffrey*, *Richard*) 表达了 “Richard有两个兄弟John和Geoffrey”

使用一阶逻辑

Assertions and Queries in FOL

- TELL将语句添加到知识库
  - 例如，TELL(*KB*, *King*(*John*)).
- ASK向知识库询问问题。例如
  - 例如，ASK(*KB*, *King*(*John*))
- ASKVARS询问什么样的*x*使得语句为真
  - 例如，ASKVARS(*KB*, *Person*(*x*))

the kinship domain

the kinship(亲属关系) domain

- 一元谓词：Male, Female
- 二元谓词：*Parent、Sibling、Brother、Sister、Child、Daughter、Son、Spouse、Wife、Husband、Grandparent、Grandchild、Cousin、Aunt、Uncle*
- *函数：Mother、Father*

the kinship(亲属关系) domain

有些公理是定义：

- 母亲(mother)是指女性(female)家长(parent)：
  -

- 丈夫(husband)则是指某人的男性(male)配偶(spouse)：
  -

- 女性(female)和男性(male)是两个不相交的集合
  -

the kinship(亲属关系) domain

有些公理是定义：

- 母亲(mother)是指女性(female)家长(parent)：
  - $\forall m, c\ Mother(c)=m \Leftrightarrow Female(m) \wedge Parent(m, c)$
- 丈夫(husband)则是指某人的男性(male)配偶(spouse)：
  - $\forall w, h\ Husband(h, w) \Leftrightarrow Male(h) \wedge Spouse(h, w)$
- 女性(female)和男性(male)是两个不相交的集合
  - $\forall x\ Male(x) \Leftrightarrow \neg Female(x)$

the kinship(亲属关系) domain

- 家长和孩子是反关系：
  -
- 祖父母(grandparent)是家长的家长：
  -
- 同胞(sibling)是某人家长的另一个孩子：
  -

the kinship(亲属关系) domain

- 家长和孩子是反关系：
  - $\forall p, c\ Parent(p, c) \Leftrightarrow Child(c, p)$
- 祖父母(grandparent)是家长的家长：
  - $\forall g, c\ Grandparent(g, c) \Leftrightarrow \exists p\ Parent(g, p) \wedge Parent(p, c)$
- 同胞(sibling)是某人家长的另一个孩子：
  - $\forall x, y\ Sibling(x, y) \Leftrightarrow x \neq y \wedge \exists p\ Parent(p, x) \wedge Parent(p, y)$
- …

the kinship(亲属关系) domain

并非所有公理都是定义

- 有些谓词没有完整定义，是因为我们具备的知识还不足以完全刻划它们。例如，没有显而易见的方法来完成以下语句：
  - $\forall x\ Person(x) \Leftrightarrow \ldots$
- 一阶逻辑允许我们无需完整定义*Person*。反而支持我们写出每个人都具有的属性或哪些属性使其成为一个人：
  - $\forall x\ Person(x) \Rightarrow \ldots$
  - $\forall x\ \ldots \Rightarrow Person(x)$

the kinship(亲属关系) domain

公理还可以是“普通事实”

- 如*Male*(*Jim*)和*Spouse*(*Jim, Laura*)。

- 经常，人们发现期望的答案是得不到的——例如，从*Male*(*George*)和*Spouse*(*George, Laura*)，希望能够推导出*Female*(*Laura*)；但是这无法由先前已知的公理推导得到。这表明______________________________。

the kinship(亲属关系) domain

公理还可以是“普通事实”

- 如*Male*(*Jim*)和*Spouse*(*Jim, Laura*)。

- 经常，人们发现期望的答案是得不到的——例如，从*Male*(*George*)和*Spouse*(*George, Laura*)，希望能够推导出*Female*(*Laura*)；但是这无法由先前已知的公理推导得到。这表明公理不充分。

the kinship(亲属关系) domain

不是所有关于论域的逻辑语句都是公理

- 有些是**定理**——它们通过公理推导而来。

- 例如，由公理$\forall x, y\ Sibling(x, y) \Leftrightarrow x \neq y \land \exists p\ Parent(p, x) \land Parent(p, y)$可得到定理：$\forall x, y\ Sibling(x, y) \Leftrightarrow Sibling(y, x)$。

- 从纯逻辑的观点来看，知识库只需包括公理。从实用观点来看，定理可以降低生成新语句的计算成本。

{8.3}

I married a widow who had a grown-up daughter. My father, who visited us quite often, fell in love with my step-daughter and married her. Hence, my father became my son-in-law, and my step-daughter became my mother. Some months later, my wife gave birth to a son, who became the brother-in-law of my father as well as my uncle. The wife of my father, that is my stepdaughter, also had a son. Thereby, I got a brother and at the same time a grandson. My wife is my grandmother, since she is my mother's mother. Hence, I am my wife's husband and at the same time her step-grandson; in other words, I am my own grandfather.

- objects? relations?

Example model {8.3}

- Objects: I, my father, widow, daughter
- Relation: sets of tuples of objects
  - marriage relation
  - parent relation
  - grandparent relation
- marriage relation:
  - {<I, widow>, <my father, daughter>}
  - then marriage(I, widow) is true
  - marriage(my father, widow) is false

natural numbers

natural numbers {8.3.3}

- *natural numbers are recursively defined as*
  - $NatNum(0)$
  - $\forall n\ NatNum(n) \Rightarrow NatNum(S(n))$
- we need axioms to constrain the successor function
  1. $\forall n\ 0 \neq S(n)$
  2. $\forall n, m\ m \neq n \Rightarrow S(m) \neq S(n)$
- define addition in terms of the successor function
  3. $\forall m\ NatNum(m) \Rightarrow +(0, m) = m$
  4. $\forall m, n\ NatNum(m) \wedge NatNum(n) \Rightarrow +(S(m), n) = S(+(m, n))$

natural numbers {8.3.3}

- 使用中缀表示法，且$S(n)$写成$n + 1$，则
- $\forall m, n\ NatNum(m) \wedge NatNum(n) \Rightarrow +(S(m), n) = S(+(m, n))$ 变成：
- $\forall m, n\ NatNum(m) \wedge NatNum(n) \Rightarrow (m + 1) + n = (m + n) + 1$

- 中缀表示法的使用是**含糖语法**的实例。含糖语法是标准语法的扩展或缩写，它不改变语句的语义。

natural numbers {8.3.3}

- 一旦有了加法，就可以直接定义乘法为重复做加法、定义求幂为重复做乘法、定义整数除法和余数、质数等等。因此，整个数论（包括密码学）可以从一个常数(   )、一个函数(   )、一个谓词(   )和四条公理(   )开始建立。

sets

sets{8.3.3}

- 我们使用集合论的常用词汇形成**含糖语法**。
- 空集是常量，用{}表示。
- 一元谓词*Set*判断对象是否为集合。
- 二元谓词为$x \in s$（x是集合s中的一个元素）和$s_1 \subseteq s_2$（集合s_1是集合s_2的子集，不一定是真子集）。
- 二元函词为$s_1 \cap s_2$（两个集合的交）、$s_1 \cup s_2$（两个集合的并）和$\{x \mid s\}$（把元素x添加到集合s而产生的集合）。

sets{8.3.3}

可能的公理：

- 集合是空集或通过将一些元素添加到集合中而构成。

sets{8.3.3}

可能的公理：

- 集合是空集或通过将一些元素添加到集合中而构成。
  - $\forall s\ Set(s) \Leftrightarrow (s = \{\ \}) \lor (\exists x, s_2\ Set(s_2) \land s = \{x \mid s_2\})$
- 空集中没有任何元素，即，*空集*无法再分解为更小的集合和元素。

sets{8.3.3}

可能的公理：

- 集合是空集或通过将一些元素添加到集合中而构成。
  - $\forall s\ Set(s) \Leftrightarrow (s = \{\ \}) \vee (\exists x, s_2\ Set(s_2) \wedge s = \{x \mid s_2\})$
- 空集中没有任何元素，即，*空集*无法再分解为更小的集合和元素。
  - $\neg \exists x, s\ \{x \mid s\} = \{\ \}$
- 将已经存在于集合中的元素添加到该集合中，该集合无任何变化。

sets{8.3.3}

可能的公理：

- 集合是空集或通过将一些元素添加到集合中而构成。
  - $\forall s\ Set(s) \Leftrightarrow (s = \{\ \}) \vee (\exists x, s_2\ Set(s_2) \wedge s = \{x \mid s_2\})$
- 空集中没有任何元素，即，*空集*无法再分解为更小的集合和元素。
  - $\neg \exists x, s\ \{x \mid s\} = \{\ \}$
- 将已经存在于集合中的元素添加到该集合中，该集合无任何变化。
  - $\forall x, s\ x \in s \Leftrightarrow s = \{x \mid s\}$

sets{8.3.3}

- 一个集合的元素是被添加到该集合中去的。我们采用递归的方式来表示：x是集合s的元素，当且仅当s等于某个集合s_2添加了元素y，其中x与y相同或者x是s_2的元素。

sets{8.3.3}

- 一个集合的元素是被添加到该集合中去的。我们采用递归的方式来表示：x是集合s的元素，当且仅当s等于某个集合s_2添加了元素y，其中x与y相同或者x是s_2的元素。
  - $\forall x, s\ x \in s \Leftrightarrow \exists y, s_2\ (s = \{y \mid s_2\} \wedge (x = y \vee x \in s_2))$
- 一个集合是另一个集合的子集，当且仅当第一个集合的所有元素都是第二个集合的元素。

sets{8.3.3}

- 一个集合的元素是被添加到该集合中去的。我们采用递归的方式来表示：x是集合s的元素，当且仅当s等于某个集合s_2添加了元素y，其中x与y相同或者x是s_2的元素。
  - $\forall x,s\ x \in s \Leftrightarrow \exists y,s_2\ (s = \{y \mid s_2\} \wedge (x = y \vee x \in s_2))$
- 一个集合是另一个集合的子集，当且仅当第一个集合的所有元素都是第二个集合的元素。

  $\forall s_1,s_2\ s_1 \subseteq s_2 \Leftrightarrow (\forall x\ x \in s_1 \Rightarrow x \in s_2)$

sets{8.3.3}

- 两个集合相等当且仅当它们互为子集。

sets{8.3.3}

- 两个集合相等当且仅当它们互为子集。

  $\forall s_1, s_2\ s_1 = s_2 \Leftrightarrow (s_1 \subseteq s_2 \wedge s_2 \subseteq s_1)$

- 一个对象属于两个集合的交集，当且仅当它同时是这两个集合中的元素。

sets{8.3.3}

- 两个集合相等当且仅当它们互为子集。

  $\forall s_1, s_2 \; s_1 = s_2 \Leftrightarrow (s_1 \subseteq s_2 \wedge s_2 \subseteq s_1)$

- 一个对象属于两个集合的交集，当且仅当它同时是这两个集合中的元素。

  $\forall x, s_1, s_2 \;\; x \in (s_1 \cap s_2) \Leftrightarrow (x \in s_1 \wedge x \in s_2)$

- 一个对象属于两个集合的并集，当且仅当它是其中任一集合的元素。

sets{8.3.3}

- 两个集合相等当且仅当它们互为子集。

  $\forall s_1, s_2 \; s_1 = s_2 \Leftrightarrow (s_1 \subseteq s_2 \wedge s_2 \subseteq s_1)$

- 一个对象属于两个集合的交集，当且仅当它同时是这两个集合中的元素。

  $\forall x, s_1, s_2 \;\; x \in (s_1 \cap s_2) \Leftrightarrow (x \in s_1 \wedge x \in s_2)$

- 一个对象属于两个集合的并集，当且仅当它是其中任一集合的元素。

  $\forall x, s_1, s_2 \;\; x \in (s_1 \cup s_2) \Leftrightarrow (x \in s_1 \vee x \in s_2)$

lists

lists {8.3.3}

- **表**与集合相似，差别在于表中元素是有序的，同一个元素可在表中出现不止一次。
- 可以采用Lisp语言的词汇：
  - *Nil*是没有元素的表常量；
  - *Cons*、*Append*、*First*和*Rest*都是函词；
  - *Find*是谓词，在表中的功能与*Member*在集合中的类似。*List*？为谓词，判断对象是否为表。
  - 和集合一样，在涉及表的逻辑语句中也经常使用含糖语法。空表用[ ]表示。项*Cons*(*x*, *y*)写成[*x* | *y*]，其中，*y*为非空表。项*Cons*(*x*, *Nil*)（即只包含元素*x*的表）用[*x*]表示。有多个元素的列表，诸如[*A*, *B*, *C*]相当于嵌套项*Cons*(*A*, *Cons*(*B*, *Cons*(*C*, *Nil*)))。

lists {8.3.3}

- 以集合公理为例，写出表的公理，包括所提及的所有常量、函数和谓词

Wumpus world {8.3.4}

- Percept([Stench,Breeze,Glitter,None,None],5)
- $\forall$t,s,g,m,c Percept([s,Breeze,g,m,c],t) $\Rightarrow$Breeze(t)
- $\forall$x,y,a,b Adjacent([x,y],[a,b]) $\Leftrightarrow$[a,b] $\in$ {[x+1,y],[x-1,y][x,y+1],[x,y-1]}
- $\forall$s,t At(Agent, s, t) $\wedge$Breeze(t) $\Rightarrow$Breezy(s)
- $\forall$s Breezy(s) $\Leftrightarrow$ $\exists$r Adjacent(r,s) $\wedge$Pit(r)
- $\forall$r Pit(r) $\Rightarrow$[$\forall$s Adjacent(r,s) $\Rightarrow$Breezy(s)]

Knowledge engineering in FOL{8.4}

- 确定任务
- 搜集相关知识
- 确定词汇表，包括谓词、函词和常量
- 对领域通用知识编码
- 对特定问题实例描述编码
- 把查询提交给推理过程并获取答案
- 知识库调试

Higher-order logic

- First-order logic allows us to quantify over objects.
- Higher-order logic also allows quantification over relations and functions.
  - e.g., "two objects are equal iff all properties applied to them are equivalent":
  - $(\forall x,y)\ ((x=y) \Leftrightarrow (\forall p,\ p(x) \Leftrightarrow p(y)))$
- Higher-order logics are more expressive than first-order; we have little understanding on how to effectively reason with sentences in higher-order logic.

Summary

- First-order logic:
  - objects and relations are semantic primitives
  - syntax: constants, functions, predicates, equality, quantifiers
  - models, interpretation, quantifiers, equality

- Increased expressive power: sufficient to express real-world problems

map of concept

对象 函数 关系

变元 常元 函词 谓词 量词 等词 连接词

解释 模型 项 原子语句 复合语句

Thanks!

next:
Chapter 9 Inference in FOL