Overview

School of Electronic and Computer Engineering
Peking University

Wang Wenmin

AI

Contents

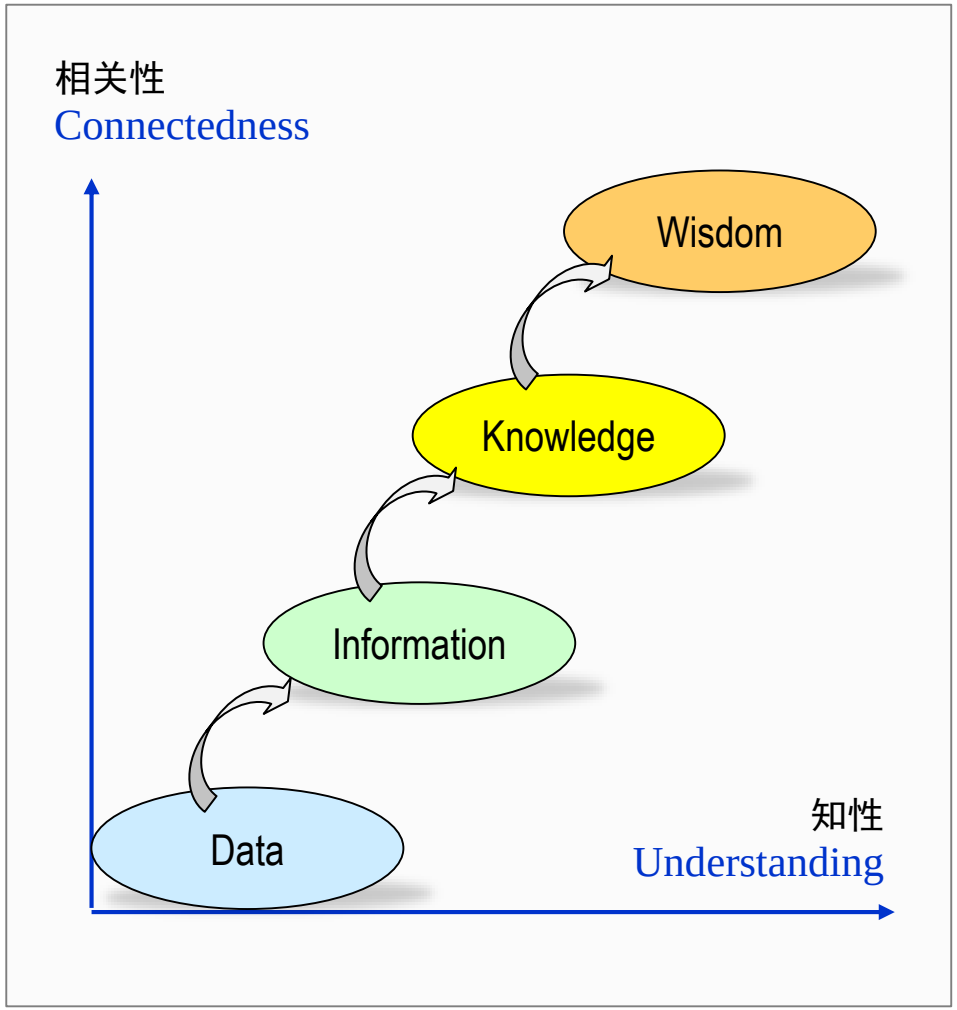
- 7.1.1 Data, Information, Knowledge and Wisdom
- 7.1.2 Explicit and Tacit Knowledge
- 7.1.3 Knowledge Types
- 7.1.4 Knowledge Base and Knowledge Base System
- 7.1.5 Knowledge Engineering (KE)
- 7.1.6 Knowledge-based Engineering (KBE)

Data, Information, Knowledge and Wisdom 数据、信息、知识与智慧

Data 数据	➢ The measures and representations of the world. 世界的计量和表征。 ➢ As fact, signal, or symbol. 表现为事实、信号、或者符号。
Information 信息	➢ Produced by assigning meaning to data. 对数据赋予含义而生成。 ➢ Structural vs. functional, subjective vs. objective. 结构与功能的，主观与客观的。
Knowledge 知识	➢ Defined with reference to information. 对信息进行加工而确立。 ➢ As processed, procedural, or propositional. 表现为加工的、过程的或者命题的。
Wisdom 智慧	➢ The experience to make decisions and judgments. 作出决定和判断的经验。 ➢ As “know-why”, “know-how”, or “why do”. 表现为“知因”、“知然”、或“因何”。

Example: in Bank 在银行

- Data: 数据
  - The numbers 100 or 5 (out of context)
    数字100或者5（无上下文）
- Information: 信息
  - Principal amount: $100, interest rate: 5%
    本金：100美元；利率：5%
- Knowledge: 知识
  - At the end of year I get $105 back
    年底拿回105美元
- Wisdom: 智慧
  - Investment?
    投资？



Explicit and Tacit Knowledge 显性与隐性知识

- Explicit knowledge 显性知识
  - Can be articulated into formal language, including grammatical statements, etc.
    可以表示为形式语言，包括语法陈述、数学表达式、等等。
  - Can be readily transmitted to others.
    可以快捷地转化成其它形式。
  - Can be easily represented using computer languages, decision trees and rules.
    可以容易地用计算机语言、决策树和规则等表示。
- Tacit knowledge 隐性知识
  - Individual experience and intangible factors, such as perspective, and etc.
    个人的经验和无形的因素、如观点、等等。
  - Hard to articulate with formal language.
    难以用形式化语言来表示。
  - Neural network offers the method to represent tacit knowledge.
    神经网络提供了表示隐形知识的方法。

Knowledge Types 知识的类型

Types 类型	Features 特点
Static knowledge 静态知识	Unlikely to change 不太可能改变
Dynamic knowledge 动态知识	Records in a database 记录在数据库中
Surface knowledge 表层知识	Accumulated through experience 通过经验积累
Deep knowledge 深层知识	Theories/Proofs/Problem Specifics 理论 / 证明 / 问题细节
Procedural knowledge 过程性知识	Describes how a problem is solved 描述如何解决问题
Declarative knowledge 陈述性知识	Describes what is known about a problem 描述已知的问题是什么
Meta-knowledge 元知识	Describes knowledge about knowledge 描述知识的知识
Heuristic knowledge 启发式知识	A rule of thumb that guide the reasoning process 引导推理过程的经验法则

Knowledge Base and Knowledge Base System 知识库和知识库系统

- The term ‘knowledge base (KB)’ was to distinguish from the more common widely used term ‘data base (DB)’.

  “知识库（KB）”这个术语是用于区分更广泛使用的术语“数据库（DB）”。

- KB is used to store complex structured and unstructured knowledge. It consists of a set of sentences, each one is expressed in a language called a knowledge representation language and represents some assertion about the world.

  知识库被用于存储复杂的结构和非结构化知识。它由一套语句组成，每个语句都是由一种被称为知识表示语言来表示的，从而表示关于世界的某些断言。

- A KB system (KBS) consists of a KB and an inference engine, where, KB represents facts about the world, inference engine can reason about those facts.

  一个知识库系统（KBS）由知识库和推理引擎组成，其中，知识库表示关于世界的事实，推理引擎则可以基于这些事实进行推理。

Knowledge Engineering (KE) 知识工程(KE)

- KE refers to all technical, scientific and social aspects involved in building, maintaining and using KB systems.

  KE指的是构建、维护和使用知识库系统中所关联的所有技术、科学和社会的方方面面。

Knowledge-based Engineering (KBE) 基于知识的工程(KBE)

- KBE is the application of knowledge-based systems technology to the domain of manufacturing design and production.

  KBE是将基于知识的系统技术用于制造设计和生产领域。

KE or KBE is essentially engineering on the basis of knowledge models, that use knowledge representation to represent the artifacts of the design process. The initial use of the KE or KBE was expert systems.

KB或KBE本质上是在知识模型基础之上的工程，它采用知识表示来表征设计过程的产品。
KB或KBE最初的应用是专家系统。

Thank you for your attention!

AI

Knowledge Representation

School of Electronic and Computer Engineering
Peking University

Wang Wenmin

Contents

- 7.2.1 Overview of Knowledge Representation
- 7.1.2 Core Issues of Knowledge Representation
- 7.1.3 Typical Methods of Knowledge Representation
- 7.1.4 What is Semantic Network
- 7.1.5 Basics of Semantic Network

Overview of Knowledge Representation 知识表示概述

What is knowledge representation
什么是知识表示

- Focus on designing computer representations that capture knowledge about the world that can be used to solve complex problems.
  关注于设计计算机表示来采集关于世界的知识，可用于解决复杂的问题。
- Make complex software easier to define and maintain than procedural code and can be used in expert systems.
  与过程性代码相比，使复杂的软件容易定义和维护，可用于专家系统。

Why use knowledge representation
为什么采用知识表示

- Conventional procedural code is not best formalism to solve complex problems.
  传统的过程性代码并非是解决复杂问题的最好形式。

Core Issues of Knowledge Representation 知识表示的核心问题

Primitive 原语

- The underlying framework used to represent knowledge, e.g., semantic network, first-order logic, and etc.

  用于表示知识的基础框架，例如：语义网络、一阶逻辑、等等。

Meta-representation 元表示

- The knowledge representation language is itself expressed in that language, e.g., in Frame based environments, all frames would be instances of a frame.

  知识表示语言用该语言本身表示，例如：在Frame环境中，所有的frames都是某个frame的实例。

Incompleteness 不完备性

- To associate certainty factors with rules and conclusions, e.g., "Socrates is human with confidence 50%".

  将确定性因子与规则和结论相关联，例如：苏格拉底是人，具有50%的置信度。

Core Issues of Knowledge Representation 知识表示的核心问题

Universals vs. Facts 共性与事实

- Universals: general statements about the world, e.g., “All humans are mortal”.
  共性：关于世界的一般性描述。例如：所有的人都会死。
- Facts: specific examples of universals, e.g., “Socrates is a human and therefore mortal”.
  事实：共性中的具体事例。例如：苏格拉底是人，因此他会死。

Expressive adequacy 表现的充分性

- How expressive they intend their representation to be.
  它们想要的表示是如何表现的。

Reasoning efficiency 推理的有效性

- Refer to the run time efficiency of the system.
  指的是该系统运行时的有效性。

Typical Methods of Knowledge Representation 典型的知识表示方法

- Bayesian Network 贝叶斯网络
- First Order Logic 一阶逻辑
- Frame-based System 基于Frame的系统
- Ontology 本体
- Production System 产生式系统
- Script 脚本
- Semantic Network 语义网络

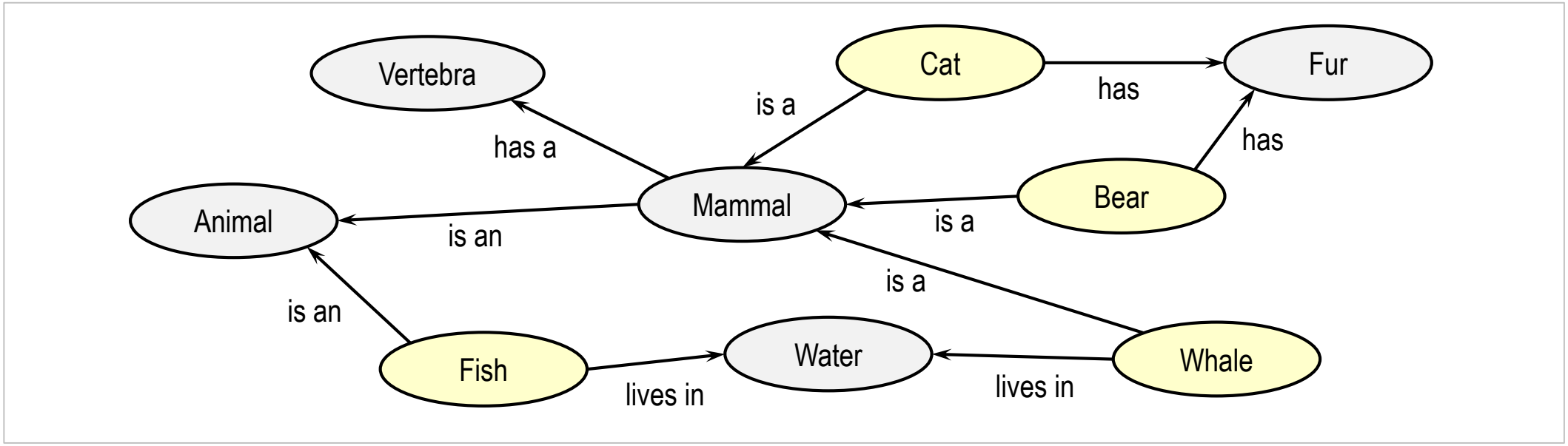
What is Semantic Network 什么是语义网络

- It is network which represents semantic relations between concepts.

  它是一种表示概念间语义关系的网络。

- It is a directed or undirected graph consisting of nodes and arcs, where nodes: represent concepts, arcs: semantic relations between the concepts.

  它是一种由节点和弧组成的有向或无向图，其中，节点：表示概念，弧：概念间的语义关系。



Example: A Semantic Network in Lisp 用Lisp语言表示的语义网络

- Using an association list. 采用关联表。

```
(defun a-knowledge-base ()
    ((canary  (is-a bird)
              (color yellow)
              (size small))
     (penguin (is-a bird)
              (movement swim))
     (bird    (is-a vertebrate)
              (has-part wings)
              (reproduction egg-laying))))
```

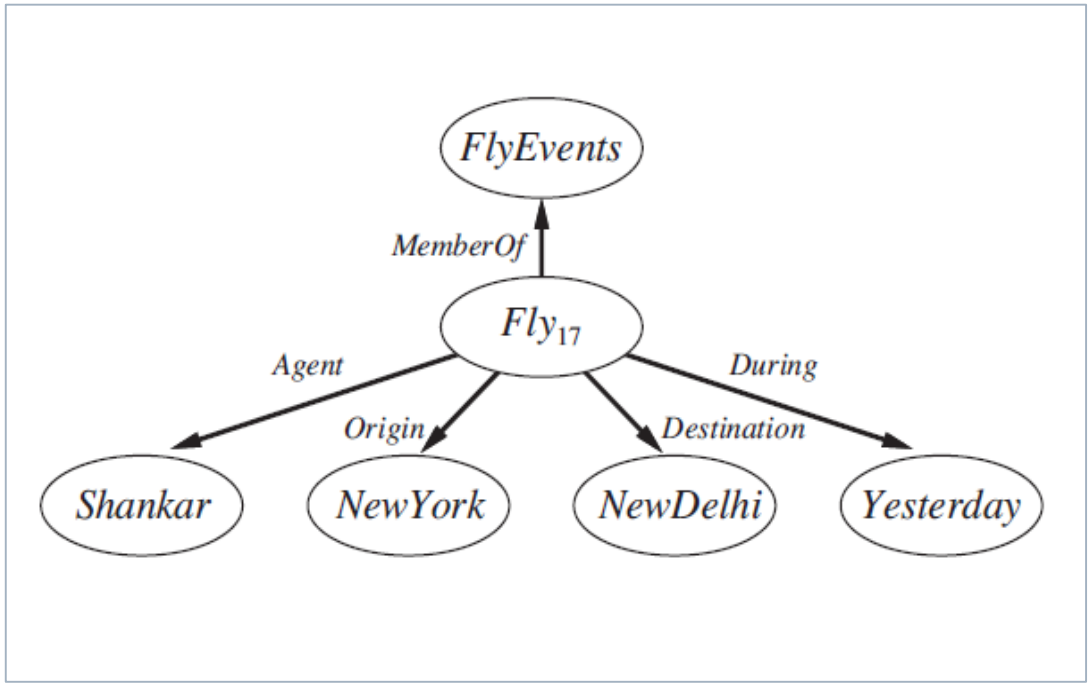
Use “assoc” function with a key of “canary” to extract all information about “canary” type.

使用“assoc”函数与关键字“canary”来提取关于“canary”类型的所有信息。

Basics of Semantic Network 语义网络的基本概念

- Semantic networks are cognitively based, organized into a taxonomic hierarchy.
  语义网络是基于认知的，被组织成为一个分类层次结构。
- A semantic network is used when one has knowledge that is best understood as a set of concepts that are related to one another.
  语义网络被采用的情形是当某种知识可以很好地化解为一组彼此相关的概念时。
- But, it is intractable for large domains, and can not represent performance or meta-knowledge very well.
  但是，它难以驾驭大型领域，并且不能很好地表现性能或者元知识。
- Some properties are not easily expressed, e.g.,
  某些特性也不易表达，例如：
  - negation, disjunction, or general non-taxonomic knowledge.
    否定、析取、或者一般的非分类知识。

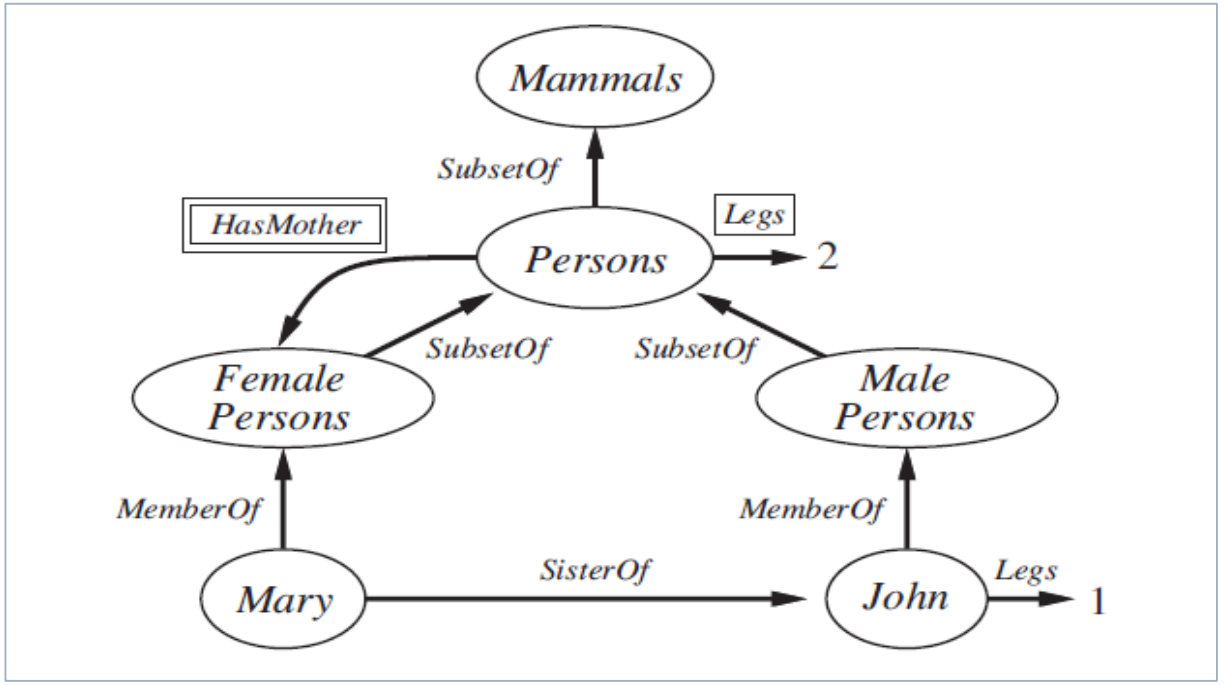
Examples



A semantic net of the logical assertion

一个逻辑断言的语义网络

Fly(*Shankar, NewYork, NewDelhi , Yesterday*)



A semantic net with categories and objects

一个具有类别和对象的语义网络

Mary ∈ *FemalePersons*, *John* ∈ *MalePersons*,

SisterOf (*Mary*, *John*)

∀*x* *x* ∈ *Persons* ⇒ [∀*y* *HasMother* (*x*, *y*) ⇒ *y* ∈ *FemalePersons*]

Thank you for your attention!

AI

Representation Using Logic

School of Electronic and Computer Engineering
Peking University

Wang Wenmin

Contents

- 7.3.1 Procedural vs. Declarative Approaches
- 7.3.2 Five Different Logics
- 7.3.3 Logical Symbols
- 7.3.4 Propositional Logic vs. First-order Logic
- 7.3.5 Formation Rules in First Order Logic
- 7.3.6 Prolog Language

Procedural vs. Declarative Approaches 过程性与陈述性方法

Procedural approaches 过程性方法

- use procedural languages, such as 采用过程性语言，例如
  - C/C++/C#/Java,
  - Lisp,
  - Python.

Declarative approaches 陈述性方法

- use declarative languages, such as 采用陈述性语言，例如
  - Propositional logic, 命题逻辑,
  - First-order logic, 一阶逻辑,
  - Temporal logic. 时序逻辑。

Five Different Logics 五种不同的逻辑

Formal Language 形式语言	Ontological Commitment 本体论约定	Epistemological Commitment 认识论约定
Propositional logic 命题逻辑	facts 事实	true/false/unknown 真/假/未知
First-order logic 一阶逻辑	facts, objects, relations 事实、对象、关系	true/false/unknown 真/假/未知
Temporal logic 时序逻辑	facts, objects, relations, times 事实、对象、关系、时间	true/false/unknown 真/假/未知
Probability theory 概率论	Facts 事实	degree of belief $\in$ [0, 1] 可信度
Fuzzy logic 模糊逻辑	facts with degree of truth $\in$ [0, 1] 事实具有真实度	known interval value 已知区间值

Logical Symbols 逻辑符号

Category 类别	Symbol 符号	Mean 含义	
Connectives 连接词	¬	not	非
	∧	and	与
	∨	or	或
	⇒	implies	蕴含
	⇔	if and only if (≡)	当且仅当
	⊨	entailment	导出
	⊭		
Quantifiers 限量词	∀	for all	所有
	∃	there exist	存在
Equality 等量词	=	equal	等于

Propositional Logic vs. First-order Logic 命题逻辑与一阶逻辑

- Propositional logic: 命题逻辑：

  also known as propositional calculus,

  亦被称为命题演算

  - use of logical connectives, deal with simple declarative propositions (if they are true or false).

    使用逻辑连接词，用于处理简单的陈述性命题。

- First-order logic: 一阶逻辑：

  also known as first-order predicate calculus,

  亦被称为一阶谓词演算，

  - additionally, use quantifiers, equality, and use predicates (often associated with sets).

    此外，还使用限量词、等量词、以及谓词（通常与集合相关联）。

Propositional Logic Syntax with BNF 用BNF表述的命题逻辑语法

$$
\begin{aligned}
\textit{Sentence} &\rightarrow \textit{AtomicSentence} \mid \textit{ComplexSentence} \\
\textit{AtomicSentence} &\rightarrow \textit{True} \mid \textit{False} \mid P \mid Q \mid R \mid \ldots \\
\textit{ComplexSentence} &\rightarrow (\textit{Sentence}) \mid [\textit{Sentence}] \\
&\mid \neg \textit{Sentence} \\
&\mid \textit{Sentence} \wedge \textit{Sentence} \\
&\mid \textit{Sentence} \vee \textit{Sentence} \\
&\mid \textit{Sentence} \Rightarrow \textit{Sentence} \\
&\mid \textit{Sentence} \Leftrightarrow \textit{Sentence} \\
\textit{OPERATOR PRECEDENCE} &: \neg, \wedge, \vee, \Rightarrow, \Leftrightarrow
\end{aligned}
$$

BNF: Backus–Naur Form

巴科斯–诺尔范式

First-Order Logic Syntax with BNF 用BNF表述的一阶逻辑的语法

$$
\begin{aligned}
\textit{Sentence} &\rightarrow \textit{AtomicSentence} \mid \textit{ComplexSentence} \\
\textit{AtomicSentence} &\rightarrow \textit{Predicate} \mid \textit{Predicate}(\textit{Term}, \ldots) \mid \textit{Term} = \textit{Term} \\
\textit{ComplexSentence} &\rightarrow < \textit{Sentence} > \mid [\, \textit{Sentence} \,] \mid \neg\, \textit{Sentence} \mid \textit{Sentence} \wedge \textit{Sentence} \\
&\mid \textit{Sentence} \vee \textit{Sentence} \mid \textit{Sentence} \Rightarrow \textit{Sentence} \mid \textit{Sentence} \Leftrightarrow \textit{Sentence} \\
&\mid \textit{Quantifier Variable}, \ldots \textit{Sentence} \\
\textit{Term} &\rightarrow \textit{Function}(\textit{Term}, \ldots) \mid \textit{Constant} \mid \textit{Variable} \\
\textit{Quantifier} &\rightarrow \forall \mid \exists \\
\textit{Constant} &\rightarrow A \mid X_1 \mid \textit{John} \mid \ldots \\
\textit{Variable} &\rightarrow a \mid x \mid s \mid \ldots \\
\textit{Predicate} &\rightarrow \textit{True} \mid \textit{False} \mid \textit{After} \mid \textit{Loves} \mid \textit{Raining} \mid \ldots \\
\textit{Function} &\rightarrow \textit{Mother} \mid \textit{LeftLeg} \mid \ldots \\
\textit{OPERATOR PRECEDENCE} &: \neg, =, \wedge, \vee, \Rightarrow, \Leftrightarrow
\end{aligned}
$$

Formation Rules in First Order Logic 一阶逻辑的形式规则

- The formation rules define
  该形式规则定义
  - terms, and
    项，以及
  - formulas.
    公式
- The formation rules can be used to write a formal grammar for terms and formulas.
  该形式规则可以用于书写项和公式的形式文法。
- Formation rules are generally context-free, i.e.,
  形式规则通常是上下文无关的，即
  - each production has a single symbol on the left side.
    每个产生式左侧有一个单一的符号。

Formation Rules of First Order Logic: Terms 一阶逻辑的形式规则：项

Rule1: Variables 规则1：变量

Any variable is a term.

任何变量都是一个项。

Rule2: Constants 规则2：常数

Any constant is also a term.

任何常数也都是一个项。

Rule3: Functions 规则3：函数

Any expression $f(t_1,...,t_n)$ of n arguments is a term, where each argument t_i is a term, and f is a function symbol of valence n. In particular, symbols denoting individual constants are 0-ary function symbols, and are thus terms.

任何n个参数的表达式f(t1,...,tn)都是一个项，其中每个参数ti是一个项，并且f是一个价n的函数符号。尤其是，表示个体常量的符号是0元函数符号，因此也是一个项。

Formation Rules of First Order Logic: Formulas 一阶逻辑的形式规则：公式

- Predicate symboles. If P is an *n-ary* predicate symbol and $t_1,...t_n$ are terms, then $P(t_1,...t_n)$ is a formula.

  谓词符号：若P是一个n元谓词符号并且$t_1,...t_n$是项，则 $P(t_1,...t_n)$ 是一个公式。

- Equality. If the equality symbol is considered part of logic, and t_1 and t_2 are terms, then $t_1=t_2$ is a formula.

  等量：若等量符号被认为是逻辑的一部分，并且 t_1和 t_2是项，则 $t_1=t_2$ 是一个公式。

- Negation. If φ is a formula, then $\neg\varphi$ is a formula.

  否定：若φ 是一个公式，则 $\neg\varphi$ 是一个公式。

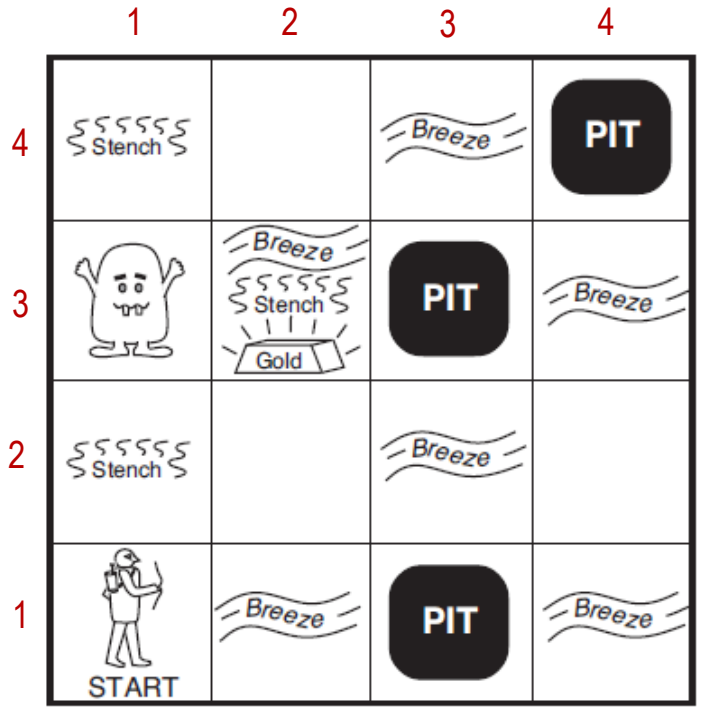
- Binary connectives. If φ and ψ are formulas, then $(\varphi \Rightarrow \psi)$ is a formula. Similar rules apply to other binary logical connectives.

  二元连接：若φ 和 ψ 是公式，则 $(\varphi \Rightarrow \psi)$是一个公式。类似的规则可用于其他二元逻辑连接。

- Quantifiers. If φ is a formula and x is a variable, then $\forall x\varphi$ and $\exists x\varphi$ are formulas.

  限量：若φ 是一个公式并且x是一个变量，则 $\forall x\varphi$和 $\exists x\varphi$ 是公式。

Example: Wumpus world 魔兽世界

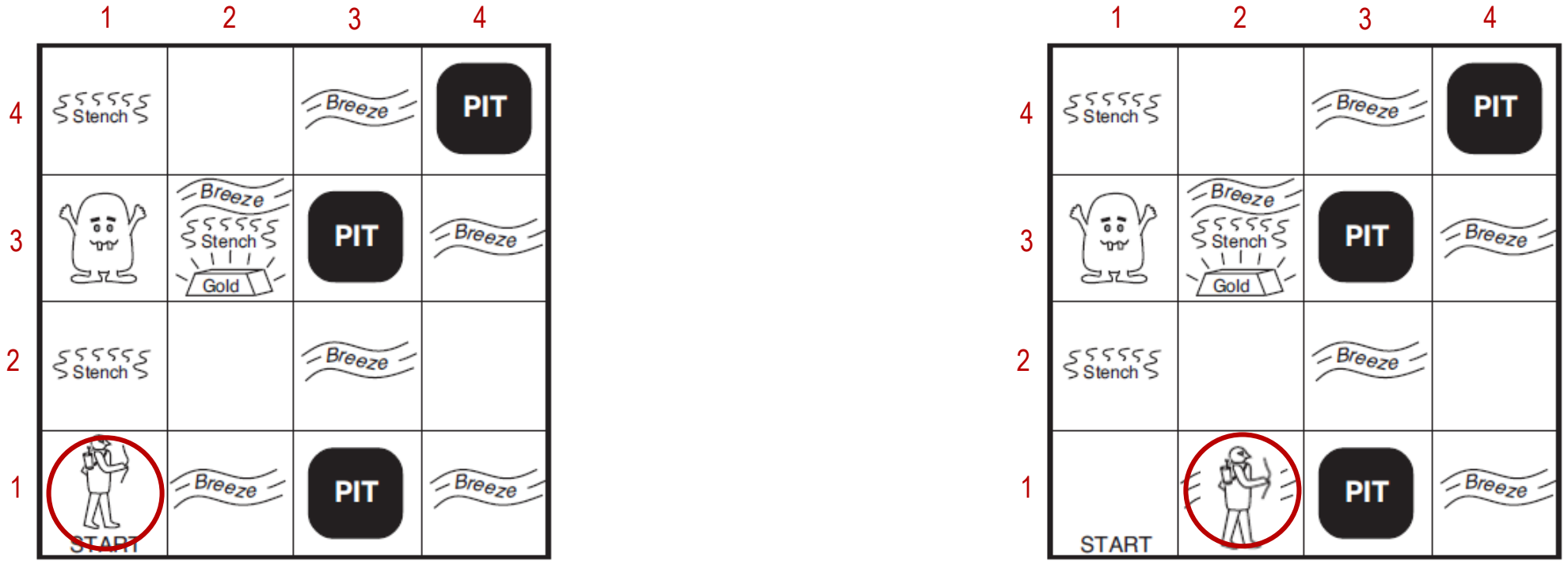


- Environment: 环境：
  - Agent 智能体,
  - Wumpus 魔兽,
  - Gold 黄金,
  - Pit (probability 0.2) 陷阱（概率0. 2）

- Performance measure: 性能指标：
  - +1000: gold, 黄金
  - -1000: death 死亡 (enters a PIT or a wumpus),
  - -1: per step, 每一步
  - -10: using the arrow. 用箭
- Actuators: 执行器：
  - *Turn Left*, *Turn Right*, *Forward*, 向左、向右、前进
  - *Shoot*: to fire an arrow, 射击：发射一只箭
  - *Grab*: to pick up gold, 抓住：拾起黄金
  - *Climb*: to climb out of cave. 攀爬：攀越陷阱
- Sensors: 感受器：
  - *Stench* 臭气,
  - *Breeze* 微风,
  - *Glitter* 闪光,
  - *Bump* 碰撞,
  - *Scream* 尖叫.

Percept [*Stench*, *Breeze*, *Glitter*, *Bump*, *Scream*]

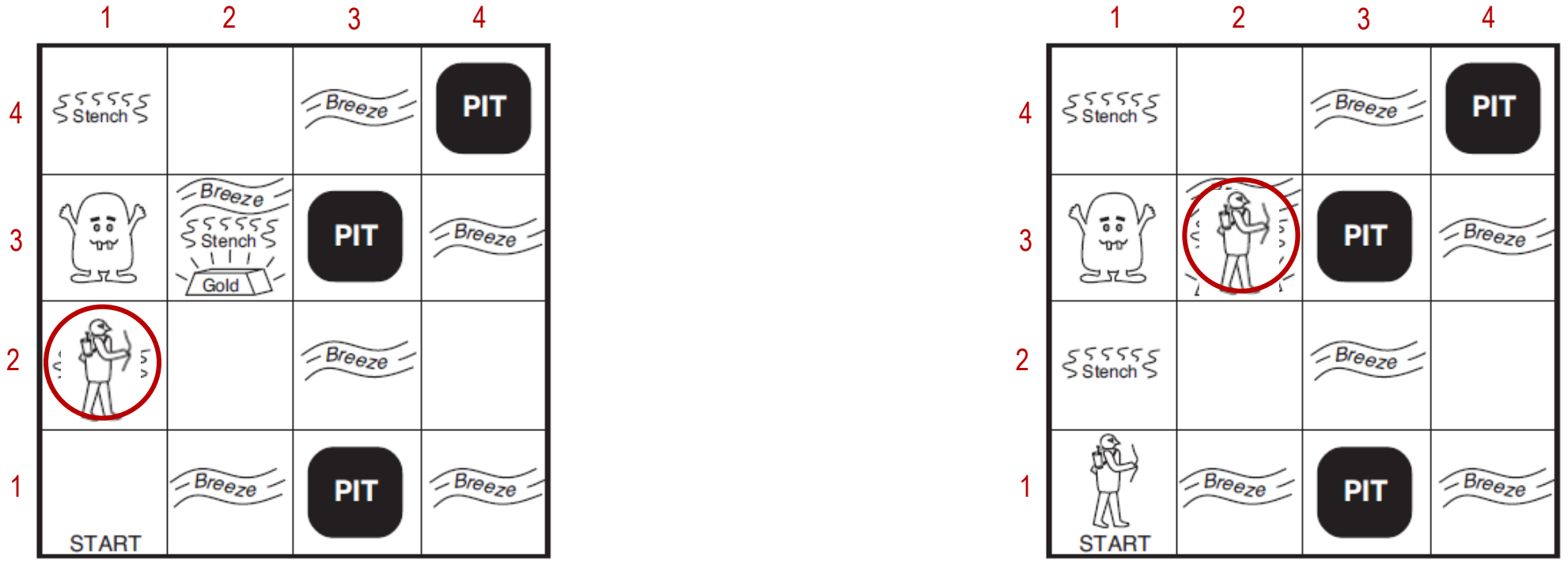
Example: Wumpus world 魔兽世界



The first step taken by the agent. (a) Initial situation, after *Percept* [*None*, *None*, *None*, *None*, *None*].
(b) After one move, with *Percept* [*None*, *Breeze*, *None*, *None*, *None*].

智能体所采取的第一步。(a) 初始状态，在*Percept* [*None*, *None*, *None*, *None*, *None*]之后。
(b) 移动一步后，具有*Percept* [*None*, *Breeze*, *None*, *None*, *None*]。

Example: Wumpus world 魔兽世界



(c) (d)

Two later stages in the progress of the agent. (c) After third move, with *Percept* [*Stench*, *None*, *None*, *None*, *None*].
(d) After fifth move, with *Percept* [*Stench*, *Breeze*, *Glitter*, *None*, *None*].

智能体进展的两个后期阶段。(c) 移动第三步后，具有*Percept* [*Stench*, *None*, *None*, *None*, *None*]。
(d) 移动第四步后，具有*Percept* [*Stench*, *Breeze*, *Glitter*, *None*, *None*]。

Using First-Order Logic for Wumpus World 用一阶逻辑描述魔兽世界

- Percept sentence 感知语句

$$Percept([Stench, Breeze, Glitter, None, None], 5)$$
$$\forall t, s, g, m, c \;\; Percept([s, Breeze, g, m, c], t) \Rightarrow Breeze(t)$$
$$\forall t, s, b, m, c \;\; Percept([s, b, Glitter, m, c], t) \Rightarrow Glitter(t)$$

- Action sentence 动作语句

$$Turn(Right), Turn(Left), Forward\ , Shoot\ , Grab, Climb.$$

- Query sentence 查询语句 (To determine which is best, 确定那个是最好的)

$$\text{ASKVARS}(\exists a, BestAction(a, 5))$$

Prolog Language Prolog语言

- Prolog language has its roots in first-order logic.
  Prolog语言起源于一阶逻辑。
- Prolog is a general purpose logic programming language, has been used for theorem proving, expert systems, natural language processing, and so on.
  Prolog是一种通用的逻辑编程语言，已经被用于定理证明、专家系统、自然语言处理，等等。
- Unlike other programming languages, Prolog is declarative: the program logic is expressed in terms of relations, represented as facts and rules.
  不同于其它编程语言， Prolog是陈述性的：程序逻辑由关系来表达，表示为事实与规则。

```
likes(bill, car).
animal(X) :- cat(X).
bird(X) :- animal(X), has(X, feather).
```

Thank you for your attention!

AI

Ontological Engineering

School of Electronic and Computer Engineering
Peking University

Wang Wenmin

Contents

- 7.4.1 What is Ontology
- 7.4.2 What is Ontological Engineering
- 7.4.3 Two Classes of Ontology Languages
- 7.4.4 Typical Ontology Languages
- 7.4.5 Applications of Ontologies

What is Ontology 什么是本体

- Ontology is the philosophical *study of the nature of being, becoming, existence, or reality, as well as the basic categories of being and their relations*.

  本体论是关于生物、生成、存在或现实的本质、以及生物及其关系的基本类别的哲学研究。

- An *ontology* is a formal naming and definition of the types, properties, and interrelationships of the entities, that really or fundamentally exist for a particular domain of discourse.

  一个本体是一种对若干实体的类型、特性和相互关系的形式化命名和定义，它真实的、或根本地存在于一个特定范围的论域。

- An *ontology* provides a common vocabulary of an area and define, with different levels of formality, the meaning of the terms and the relationships between them.

  一个本体提供了一个领域的公共词汇，并且用不同层次的形式定义一些术语的含义和它们之间的关系。

What is Ontology 什么是本体

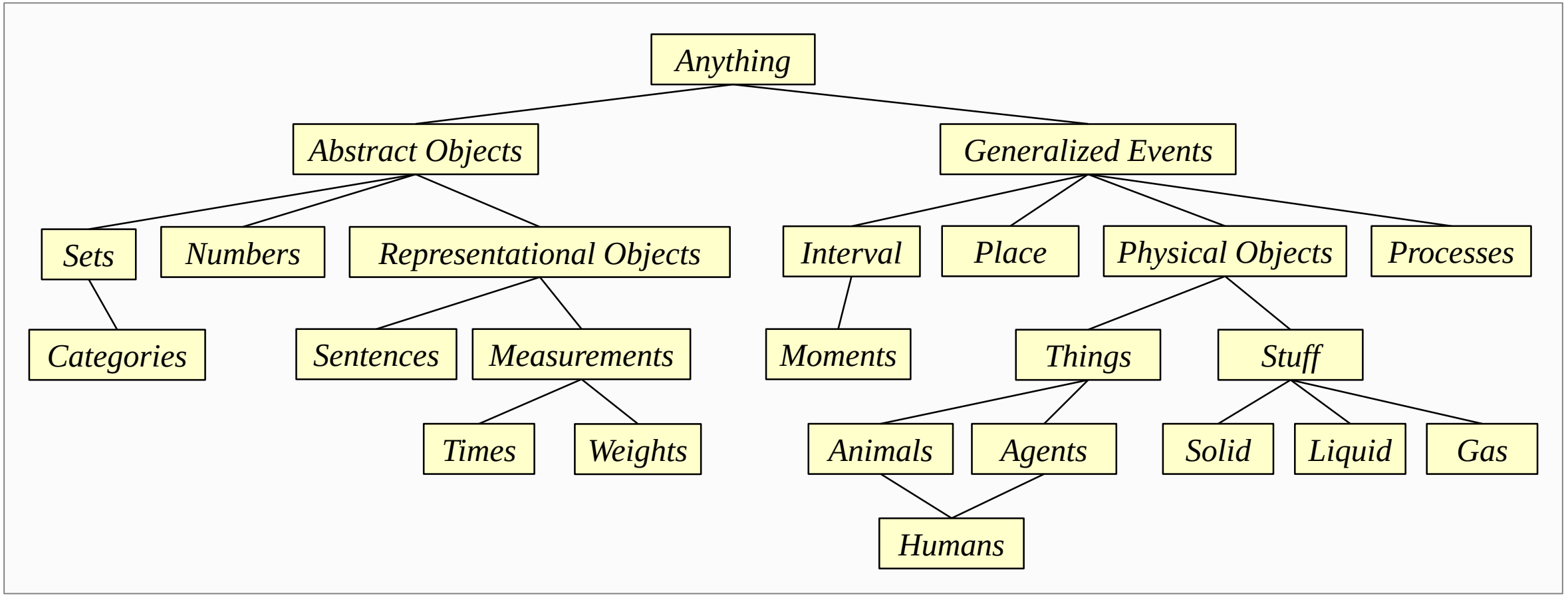
- Some fields create ontologies to organize information, then the ontologies can be applied to *problem solving*, include:
  某些领域创建本体来组织信息，然后再将这些本体用于问题求解，包括：

Artificial intelligence	人工智能
Semantic Web	语义Web
Systems engineering	系统工程
Software engineering	软件工程
Biomedical informatics	生物医学信息学
Library science	图书馆学
Information architecture	信息架构

Type of Ontology 本体的类型

Types 类型	Concepts 概念
Upper ontology 上层本体	A model of the common objects that are generally applicable across a wide range of domain ontologies. 一种公共对象的模型，通常可应用于广泛的领域本体。
Domain ontology 领域本体	Relevant to a particular topic or area of interest, e.g., information technology, or particular branches of science. 与一个特定的主题或兴趣领域有关，如：信息技术或科学的某个分支。
Hybrid ontology 混合本体	A combination of an upper ontology and a domain ontology. 一个上层本体与一个领域本体的结合。

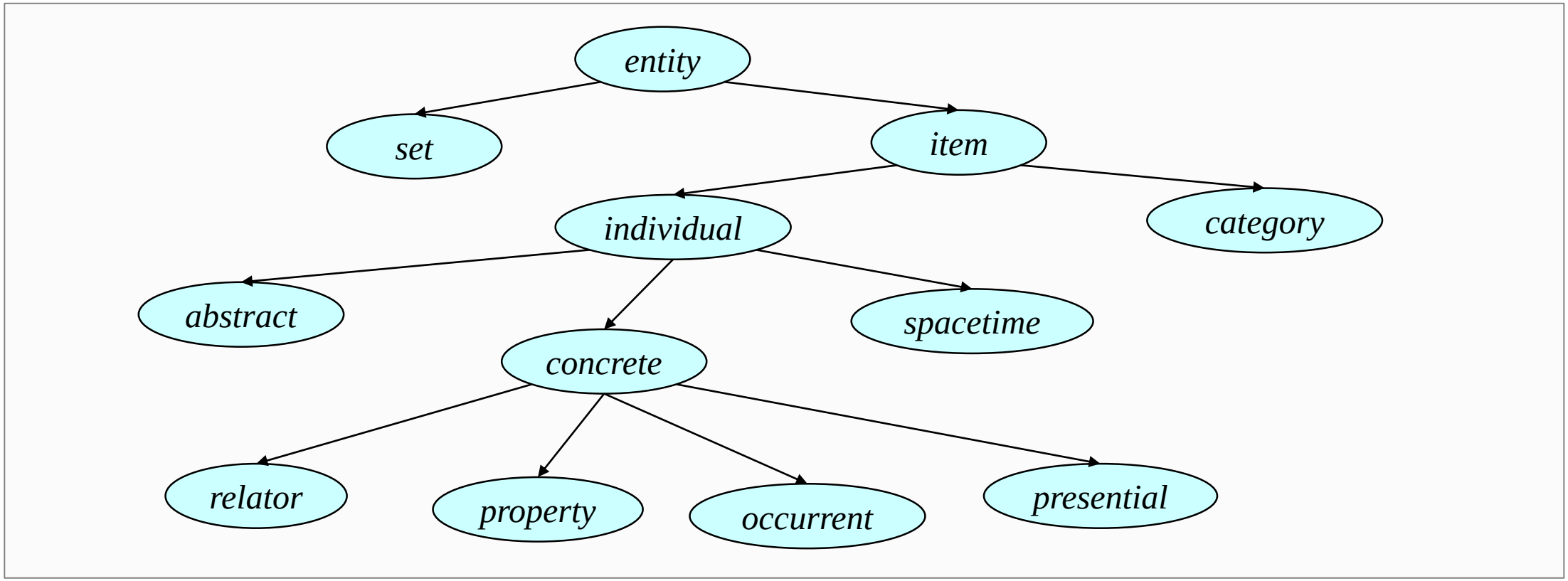
Example: The Upper Ontology of the World



The lower concept is a specialization of the upper one.

其底层的概念是上层概念的一个特化。

Example: A Domain Ontology 一个领域本体



An ontology represents a set of concepts within a domain, and the relationships between those concepts.

一个本体用来表示一个领域中概念的集合，以及这些概念之间的关系。

Components of Ontology 本体的成分

Components 成分	Instances 实例
Individuals 个体	instances or objects 实例或对象
Classes 类	sets, collections, concepts, classes in programming, types of objects, or kinds of things 集、集合、概念、编程中的类、对象的类型、或者事物的种类
Attributes 属性	aspects, properties, features, characteristics, or parameters that objects (and classes) can have 对象（以及类）所能够具有的方面、属性、特性、特征、或者参数
Relations 关系	ways in which classes and individuals can be related to one another 类和个体可以相互关联的方式
Function terms 功能项	complex structures formed from certain relations that can be used in place of an individual term in a statement 从某些关系所形成的复杂结构，可以用来替代某个陈述中的独立项。

Components of Ontology 本体的成分

Components 成分	Instances 实例
Restrictions 限定	formally stated descriptions of what must be true in order for some assertion to be accepted as input 正式规定的必须为真的描述，为了使某些断言被接受为输入
Rules 规则	statements in the form of an if-then (antecedent-consequent) sentence that describe the logical inferences that can be drawn from an assertion in a particular form 采用if-then（前因–后果）语句形式的陈述，描述从一个特定形式的断言得到的逻辑推理
Axioms 公理	assertions (including rules) in a logical form that together comprise the overall theory that the ontology describes in its domain of application. 逻辑形式的断言（包括规则），共同构成本体在它的应用领域描述的全部理论
Events 事件	the changing of attributes or relations 属性或关系的改变

What is Ontological Engineering 什么是本体工程

- A field which studies the methods and methodologies for building ontologies:
  一个研究构建本体的方法和方法学的领域：
  - The formal representations of a set of concepts within a domain, and
    一组某个领域概念的形式化表示，以及
  - The relationships between those concepts.
    这些概念之间的关系。
- The general concepts to represent, i.e.:
  要表示的一般概念，即：
  - Event, 事件、
  - Time, 时间、
  - Physical objects, and 物理对象、以及
  - Beliefs. 置信度。

What is Ontological Engineering 什么是本体工程

- Ontological engineering is to study

  本体工程研究：

  - the ontology development process,

    本体开发过程，
  - the ontology life cycle,

    本体生命周期，
  - the methods and methodologies for building ontologies,

    构建本体的方法和方法学，
  - the tool suites and languages to support ontologies.

    支持本体的工具套件和语言的新领域。

Two Classes of Ontology Languages 两类本体语言

- 1) Traditional syntax ontology languages
  传统语法的本体语言
  - Common Logic
  - DOGMA (Developing Ontology-Grounded Methods and Applications)
  - F-Logic (Frame Logic)
  - KIF (Knowledge Interchange Format)
  - KM programming language
  - LOOM (ontology)
  - OCML (Operational Conceptual Modelling Language)
  - OKBC (Open Knowledge Base Connectivity)

Two Classes of Ontology Languages 两类本体语言

- 2) Markup ontology languages

  标记式本体语言

  These languages use a markup scheme to encode knowledge, most commonly with XML.

  这些语言使用标记方案对知识进行编码，最常用的是XML。

  - DAML+OIL
  - OIL (Ontology Inference Layer)
  - OWL (Web Ontology Language)
  - RDF (Resource Description Framework)
  - RDFS (RDF Schema)
  - SHOE

Typical Ontology Languages 典型的本体语言

Common logic

- ISO standard 24707, a specification for a family of ontology languages that can be accurately translated into each other.
  已成为ISO 24707标准，是一套本体语言的规范，这些语言彼此之间可以被精确地转换。

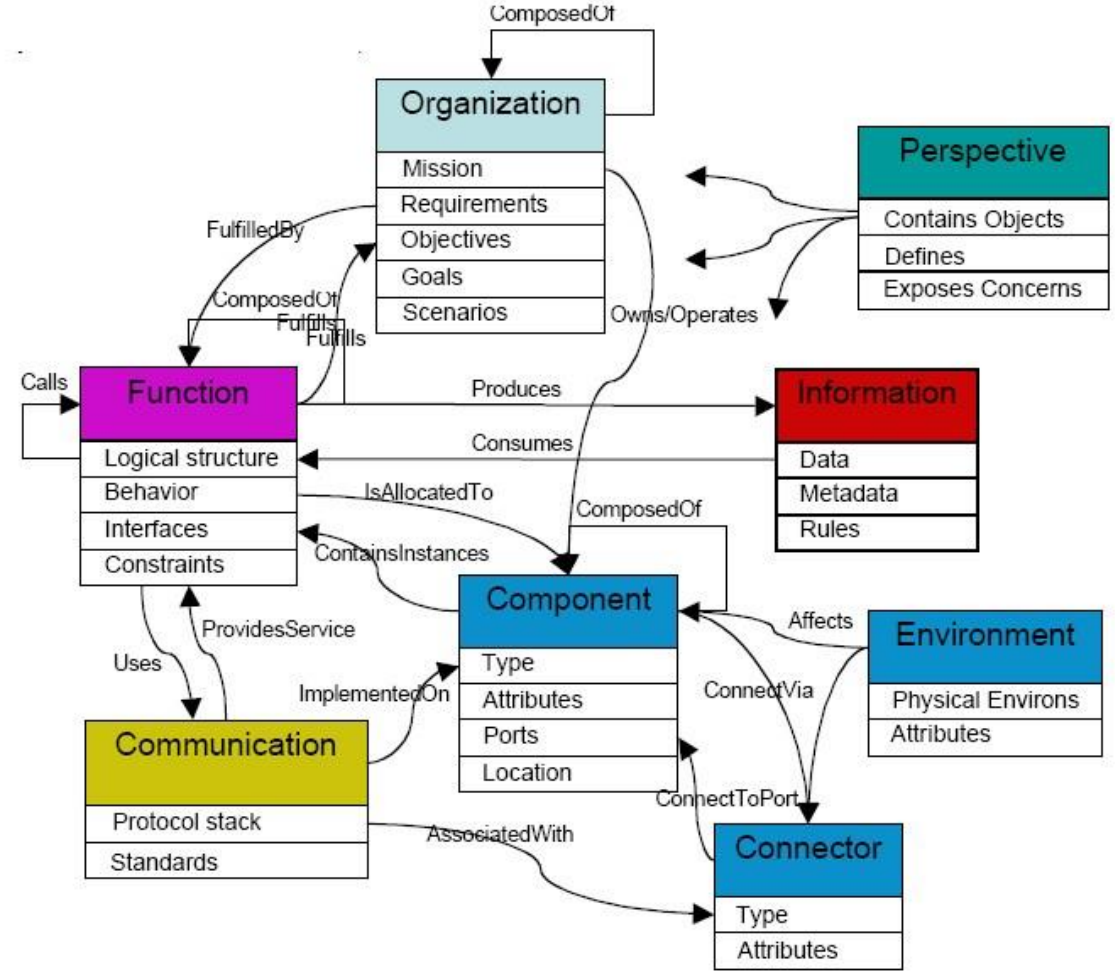
OWL (Web Ontology Language)

- A language for making ontological statements, intended to be used over Web.
  是一种用于本体陈述的语言，目的是在Web上使用。
- All its elements (classes, properties and individuals) are defined as RDF (Resource Description Framework) resources, and identified by URIs (Uniform Resource Identifiers).
  所有的元素（类、特性和个体）都被定义为RDF（资源描述框架）资源，并且通过URIs（统一资源标识）加以识别。

Applications of Ontologies 本体的应用

Knowledge management	知识管理
Natural language processing	自然语言处理
E-commerce	电子商务
Intelligent information integration	智能信息集成
Bio-informatics	生物信息学
Education	教育
Semantic Web	语义Web

Example: A top level ontology 一个顶层本体



A top level ontology based on the nominal set of views.

一个基于views公称集的顶层本体

Thank you for your attention!

AI

Bayesian Networks

School of Electronic and Computer Engineering
Peking University

Wang Wenmin

Contents

- 7.5.1 About Uncertain Knowledge
- 7.5.2 Rational Decisions
- 7.5.3 Algorithm of a Decision-theoretic Agent
- 7.5.4 Bayes' Rule
- 7.5.5 Representing Full Joint Distribution
- 7.5.6 Constructing Bayesian Networks

Contents

- 7.5.7 Compactness
- 7.5.8 Node Ordering
- 7.5.9 Conditional Independence Relations

About Uncertain Knowledge 关于不确定性知识

- Evolution of an intelligent agent: problem solving, reasoning, planning and learning.
  智能体的进化：问题求解、推理、规划以及学习。
- Agents may need to handle uncertainty, due to partial observability and non-determinism.
  智能体可能需要处理不确定性，由于部分可观察性和不确定性问题。
- To make decision with uncertainty, we need
  在不确定性的情况下做出决策，我们需要
  - Probability theory,
    概率论，
  - Utility theory,
    效用论，
  - Decision theory.
    决策论。

Example: An uncertainty problem 一个不确定性问题

A_{90} = home to airport 90 minutes by taxi before flight departs.

从家里打车在航班起飞前90分钟到机场

- Question: 问题

  "Will A_{90} get me to the airport on time?"

  A_{90} 能使我准时到达机场吗？

- Answer: 答案

  The taxi agent concludes either:

  出租汽车公司给出两个结论中的一个：

  - the risks falsehood: "A_{90} will get us there in time".

    有风险的谎言：A_{90}将使我们及时到达机场。

  - the weaker conclusion: "A_{90} will get us there in time, if there is no traffic jam, I don't get into an accident, the car doesn't break down, and ..."

    A_{90}将使我们及时到达机场，如果没有交通堵塞、不出交通事故、汽车不出故障的话，……

Rational Decisions 理性决策

- **Probability theory** 概率论

  for dealing with degrees of belief.

  是用于处理置信度的理论

- **Utility theory** 效用论

  the quality of being useful.

  是有效性的质量

  - to represent and reason with preferences, every state has a degree of usefulness/utility.

    用偏好来表现和推理，每个状态都具有“有效性/效用”的度量值。

- **Decision theory** 决策论

  the general theory of rational decisions.

  是理性决策的通论

  - Decision theory = probability theory + utility theory

    决策论 = 概率论 + 效用论

Algorithm of a Decision-theoretic Agent 一种决策论智能体的算法

function DECISION-THEORETIC-AGENT(*percept*) **returns** an *action*
  persistent: *belief_state*, probabilistic beliefs about the current state of the world
    action, the agent's action

  update *belief_state* based on *action* and *percept*
  calculate outcome probabilities for actions,
    given action descriptions and current *belief_state*
  select *action* with highest expected utility,
    given probabilities of outcomes and utility information
return *action*

A decision-theoretic agent that selects rational actions.
一个选择理性动作的决策论智能体

Bayes' Rule 贝叶斯规则

Product rule 乘积规则

- Two ways to factor a joint distribution over two variables:

  两个变量联合分布的两种计算方法：

$$P(a \wedge b) = P(a \mid b)\, P(b) \quad \text{and} \quad P(a \wedge b) = P(b \mid a)\, P(a)$$

Bayes' rule 贝叶斯规则

$$P(b \mid a) = \frac{P(a \mid b)\, P(b)}{P(a)}$$

- This rule underlies most modern AI for probabilistic inference.

  这个规则成为大多数现代人工智能概率推理的基础。

- Why is Bayes' rule useful 为什么贝叶斯定理有用
  - Often we have good probability estimates for three terms to compute the fourth.

    我们常常需要根据三个项的概率估计值去计算第四个。

Example: Inference with Bayes' Rule 贝叶斯规则进行推理

- Often we perceive as evidence the *effect* of some unknown cause, and would like to determine that *cause*. In that case, Bayes' rule becomes

  我们往往想根据一些未知原因的证据，来查明其原因。这样，贝叶斯定理就变成了

$$P(cause \mid effect) = \frac{P(effect \mid cause)P(cause)}{P(effect)}$$

- Knows $P(symptoms \mid disease)$ and want to derive a diagnosis, $P(disease \mid symptoms)$.

  已知 $P(symptoms$ 症状 $\mid disease$ 疾病$)$，想要得出一个诊断 P（疾病 ｜ 症状）。

$P(s \mid m) = 0.7$ // *conditional probability that meningitis causes a stiff neck* 脑膜炎导致颈部僵硬的条件概率

$P(m) = 1/50000$ // *prior probability that a patient has meningitis* 病人患脑膜炎的先验概率

$P(s) = 0.01$ // *prior probability that any patient has a stiff neck* 任何病人患有颈部僵硬的先验概率

$$P(m \mid s) = \frac{P(s \mid m)P(m)}{P(s)} = \frac{0.7 \times 1/5000}{0.01} = 0.0014$$ // *a stiff neck to have meningitis* 颈部僵硬患有脑膜炎的概率

About Bayesian Networks 贝叶斯网络

- A probabilistic graphical model (a type of statistical model)

  一种概率图模型（一种统计模型的类型）

- With a directed acyclic graph (DAG), it represents:
  - a set of random variables, and conditional dependencies between the variables.

  采用一种有向无环图 (DAG)，它表示：一组随机变量，以及变量之间的条件相关性。

- Its specification: 它的规范如下：
  - 1) a set of nodes, each corresponds to a random variable, 2) a set of directed links to those nodes, and 3) a conditional probability distribution for each node given its parents:

    1）一组节点，每个节点对应于一个随机变量，2）一组这对这些节点的有向连接，以及 3）每个节点在给定双亲下的条件概率分布：

$$\mathbf{P}(X_i \mid Parents(X_i))$$

About Bayesian Networks 贝叶斯网络

- The name of Bayesian networks is the most common one, but there are many synonyms, including:

  贝叶斯网络这个名称是最常用的，但还有许多同义词，包括：

  - belief network, 信念网络
  - probabilistic network, 概率网络
  - causal network. 因果网络

- A Bayesian network represents a set of random variables and their conditional dependencies.

  一个贝叶斯网络表示一组随机变量和他们的条件依赖关系。

  - E.g., it could represent the probabilistic relationships between diseases and symptoms.

    例如：它可以表示疾病与症状之间的概率关系。

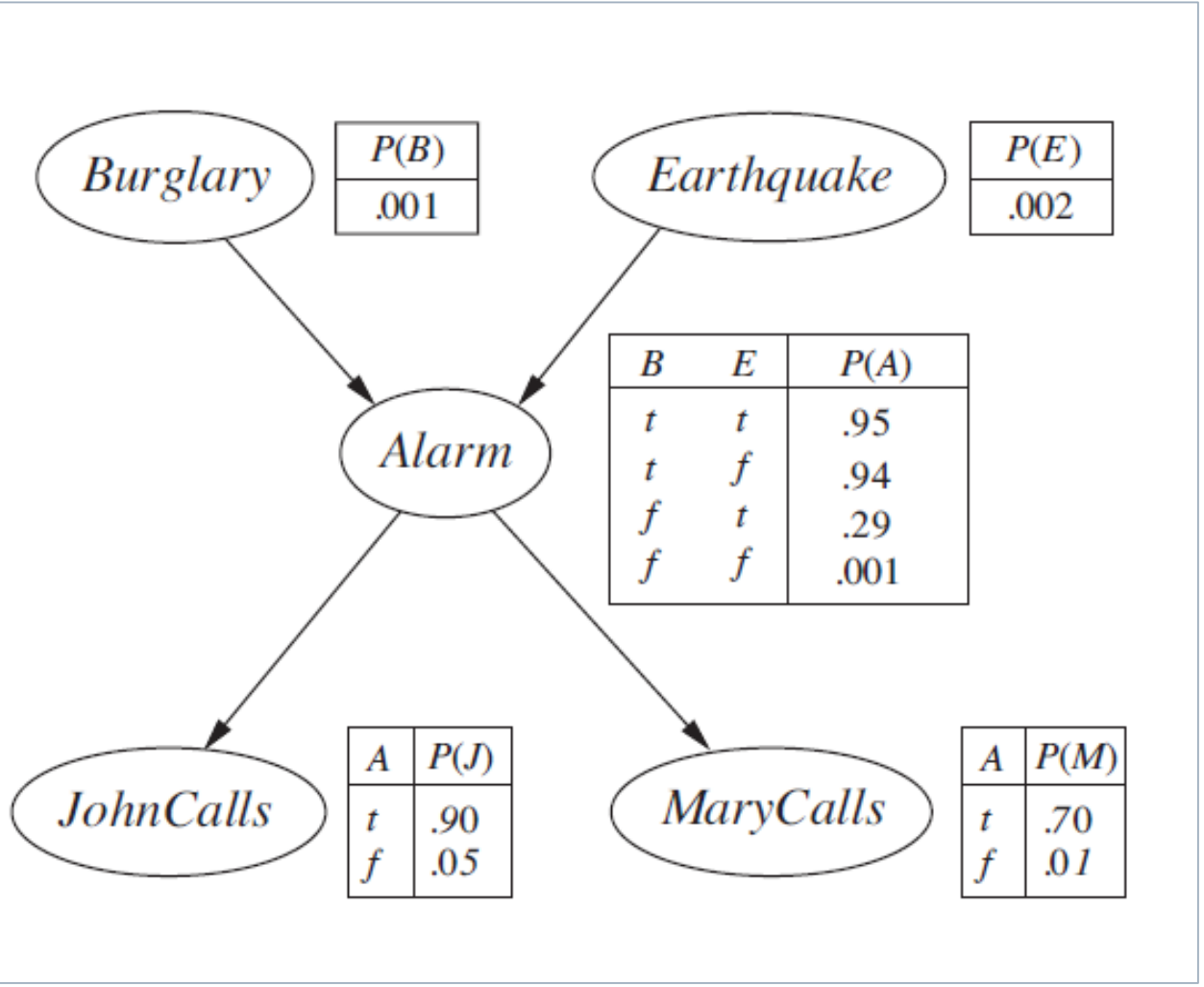
Two Views 两个观点

- The two views to understand semantics of Bayesian networks:
  理解贝叶斯网络语义的两个观点：
  - 1st: to view the network as a representation of the joint probability distribution.
    第一、将该网络视为一种联合概率分布的表示。
  - 2nd: to view it as an encoding of a collection of conditional independence statements.
    第二、将其视为一组条件独立语句的一种编码。
- The two views are equivalent, but: 这两个观点是等价的，但是：
  - 1st view: helpful in understanding how to construct networks,
    第一种观点：有助于理解如何构建网络，
  - 2nd view: helpful in designing inference procedures.
    第二种观点：有助于设计推理过程。

Example: A typical Bayesian network 一个典型的贝叶斯网络

- A burglar alarm installed at home, used to detect a burglary or minor earthquakes.
  房子里安装了一个防盗报警器，用于检测被盗或地震。
- Two neighbors, John and Mary, who have promised to call you at work when they hear the alarm.
  有两个邻居，John和Mary，他们已答应当听到报警时，就给你的办公室打电话。
- Variables 变量： *Burglar*, *Earthquake*, *Alarm*, *JohnCalls*, *MaryCalls*.
- Network topology reflects the knowledge:
  网络的拓扑结构要反应如下知识：
  - A *Burglar* or an *Earthquake* can set the *Alarm*.
    盗窃或者地震会导致报警。
  - The *Alarm* can cause *Mary* or *John* to call.
    报警会引起Mary或John打电话。

Example: A typical Bayesian network 一个典型的贝叶斯网络



A Bayesian network with the conditional probability tables (CPTs).

一个具有条件概率表 (CPTS) 的贝叶斯网络。

Where 其中

- *B*, *E*, *A*, *J*, *M* stand for *Burglary*, *Earthquake*, *Alarm*, *JohnCalls*, *MaryCalls*;

  B, *E*, *A*, *J*, *M* 代表 *Burglary*, *Earthquake*, *Alarm*, *JohnCalls*, *MaryCalls*

- *t* and *f* stand for *true* and *false*;

  t 与 *f* 代表 *true* 和 *false*

- Each row shows one number p for $X_i = t$, (the number for $X_i = f$ is just $1 - p$).

  每一行显示 $X_i = t$ 的概率值 p，$X_i = f$ 时概率值则为 $1\text{-}p$

Representing Full Joint Distribution 表征全联合分布

- By the product of the elements of conditional distributions:
  采用条件分布元素的乘积：

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid parents(X_i))$$

- To illustrate this, we can calculate the probability: Alarm has sounded, but neither a Burglary nor an Earthquake has occurred, and both John and Mary call.
  为了说明，我们可以计算概率：报警响起，但盗窃和地震都未发生，而John和Mary打电话。
- We multiply entries from the joint distribution:
  我们依据联合分布，将这些项相乘：

$$P(j, m, a, \neg b, \neg e) = P(j \mid a)\, P(m \mid a)\, P(a \mid \neg b \wedge \neg e)\, P(\neg b)\, P(\neg e)$$

$$= 0.90 \times 0.70 \times 0.001 \times 0.999 \times 0.998 = 0.000628$$

where j, m, a, b, e stand for *JohnCalls, MaryCalls, Alarm, Burglary, Earthquake*.

其中 j, m, a, b, e 代表 *JohnCalls, MaryCalls, Alarm, Burglary, Earthquake*.

Constructing Bayesian Networks 构建贝叶斯网络

- First, rewrite joint distribution in terms of conditional probability, using product rule:
  首先，用条件概率公式对联合概率进行改写，使用如下乘积规则：

$$P(a \wedge b) = P(a \mid b)\, P(b)$$

$$P(x_1, \ldots, x_n) = P(x_n \mid x_{n-1}, \ldots, x_1)\, P(x_{n-1}, \ldots, x_1)$$

- Then, repeat the process, reducing each conjunctive probability to a conditional probability and a smaller conjunction. We end up with one big product.
  然后，重复这个过程，将每个合取概率缩减为条件概率和较小的合取。最终得到一个大的乘积。

$$P(x_1, \ldots, x_n) = P(x_n \mid x_{n-1}, \ldots, x_1)\, P(x_{n-1} \mid x_{n-2}, \ldots, x_1) \ldots P(x_2 \mid x_1)\, P(x_1)$$

$$= \prod_{i=1}^{n} P(x_i \mid x_{i-1}, \ldots, x_1)$$

This is called chain rule, for any set of random variables.
这被称为对任意一组随机变量的链式法则。

Constructing Bayesian Networks 构建贝叶斯网络

- The specification of the joint distribution is equivalent to the general assertion that, for every variable X_i in the network,

  联合分布的规格等同于一般性断言，即：对于网络中的每个变量X_i，

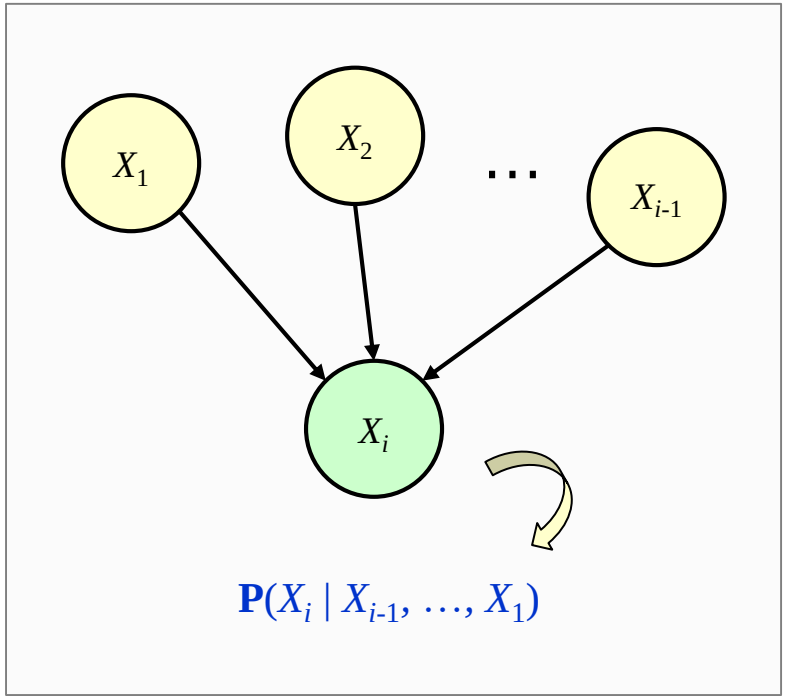
  $$\mathbf{P}(X_i \mid X_{i\text{-}1}, \ldots, X_1) = \mathbf{P}(X_i \mid Parents(X_i))$$

  provided that 假设

  $$Parents(X_i) \subseteq \{X_{i\text{-}1}, \ldots, X_1\}$$

- The equation says: Bayesian network is a correct representation of domain, only if each node is conditionally independent of its other predecessors in the node ordering, given its parents.

  该等式说明：贝叶斯网络是域的一个正确表示，仅当给定其父节点，每个节点条件独立于节点序中其它前趋节点时。



Constructing Bayesian Networks 构建贝叶斯网络

- We can satisfy this condition with this methodology:
  我们可以用如下方法来满足该条件：
  - 1. *Nodes*: First determine the set of variables to model the domain, $\{X_1, \ldots, X_n\}$.
    节点：先确定要对域建模的变量集，$\{X_1, \ldots, X_n\}$。
  - *2. Links*: For $i = 1$ to n do: 链接：从$i = 1$至n，做：
    - Choose, from $X_1, \ldots, X_{i-1}$, a minimal set of parents for X_i.
      从 $X_1, \ldots, X_{i-1}$ 中选择 X_i 的父节点的最小集。
    - For each parent insert a link from the parent to X_i.
      对每个父节点插入一个从该父节点至 X_i 的链接。
    - Write down the conditional probability table (CPT),
      记录 该条件概率表 (CPT)，

$$\mathbf{P}(X_i \mid Parents(X_i)).$$

The parents of node X_i should contain all nodes in $X_1, \ldots, X_{i-1}$ that directly influence X_i.
节点 X_i 的父辈应该包含直接影响 X_i 的 $X_1, \ldots, X_{i-1}$ 中的所有节点。

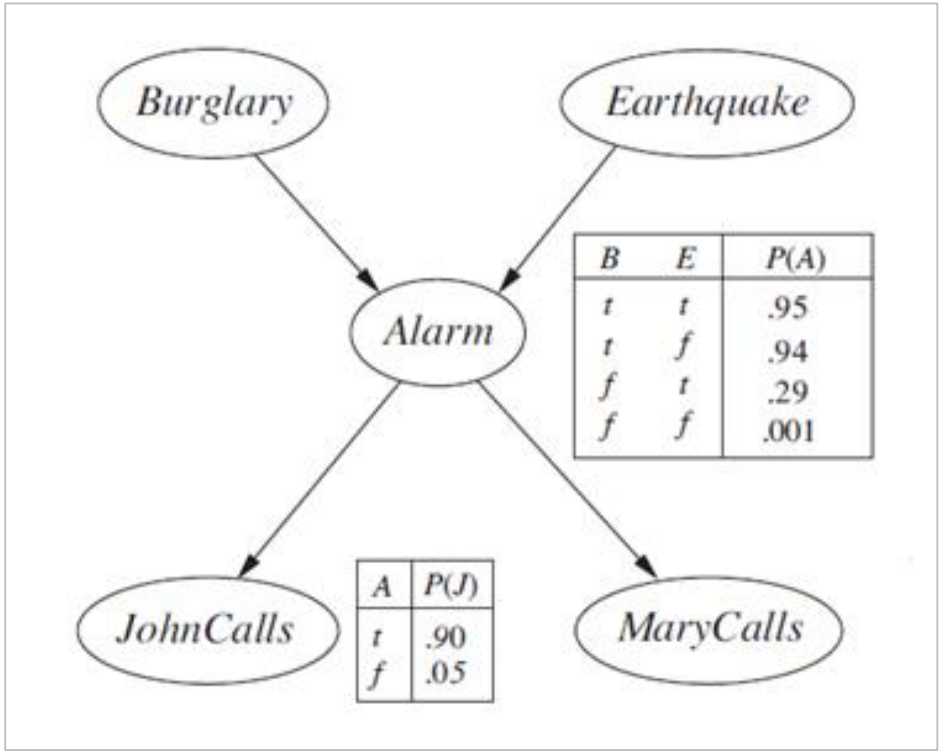
Example: A typical Bayesian network 一种典型的贝叶斯网络

- Suppose we have completed the network, except for the choice of parents for *MaryCalls*.
  假设除了*MaryCalls*父节点的选择之外，我们已经完成了该网络。
- *MaryCalls* is not directly influenced by a *Burglary* or an *Earthquake*. Her calling behavior only through their effect on the alarm.
  *MaryCalls*并非直接受盗窃或地震的支配，她打电话的行为只受它们对报警器的影响。
- Also, given the alarm state, whether John calls has no influence on Mary's calling.
  并且，给定报警状态，John打电话与否并不影响Mary的电话。
- Therefore, the following *conditional independence statement* holds:
  因此，如下条件独立语句成立：

$$\mathbf{P}(MaryCalls \mid JohnCalls, Alarm, Earthquake, Burglary) = \mathbf{P}(MaryCalls \mid Alarm)$$

Compactness 紧凑性

- A Bayesian network is far more *compact* than full joint distribution. Its compactness is a general property of locally structured (also called sparse) systems.
  贝叶斯网络远比全联合分布紧凑。它的紧凑性是局部结构化（也称为稀疏）系统的一般特性。
- In a locally structured system, each subcomponent interacts directly with only a bounded number of other components.
  在局部结构化系统中，每个子成分仅直接与其它成分的有限数量打交道。
- In Bayesian networks, a CPT (conditional probability table) for a Boolean variable with k parents has 2^k rows.
  贝叶斯网络中，具有k个父节点的布尔变量的CPT（条件概率表）有2^k行。

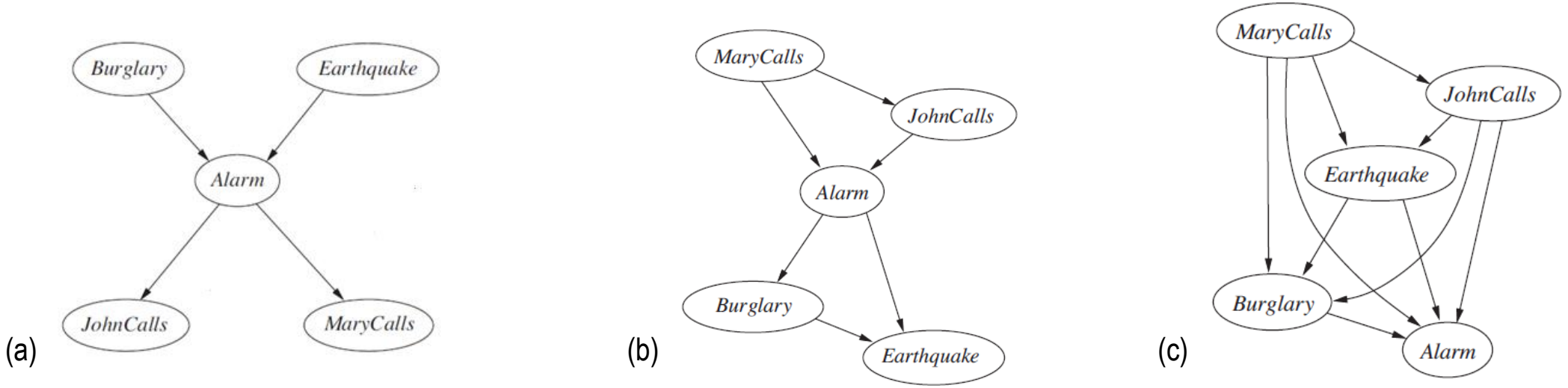


Compactness 紧凑性

- Assume there are n Boolean variables: 假设有n个布尔变量：
  - Bayesian networks: 贝叶斯网络
    if each variable has no more than k parents, then the complete network can be specified by $n2^k$ numbers.
    如果每个变量的父节点不超过k个，则全部网络可以用 $n2^k$ 个数指定。
  - Full joint distribution: 全联合分布
    it contains 2^n numbers.
    它包含 2^n 个数。
- E.g., suppose $n = 30$ nodes, each with $k = 5$ parents, then
  例如：设节点$n = 30$，每个具有 $k = 5$ 个父节点，则
  - Bayesian network: $n2^k = 30 \times 2^5 = 960$.
  - Full joint distribution: $2^n = 2^{30} = 1{,}073{,}741{,}824$.

Node Ordering 节点排序

- We will get a compact Bayesian network only if choose node ordering well (Fig. a).
  只有当选择一个好的节点排序（图a），我们才会得到一个紧凑的贝叶斯网络。
- What happens if we choose the wrong order (Fig. b and c).
  如果我们选择一个差的排序会发生什么（图b和c）。



(a) A typical Bayesian network. 一个典型的贝叶斯网络
(b) A bad node ordering 一个差的节点排序： *MaryCalls, JohnCalls, Alarm, Burglary, Earthquake.*
(c) A very bad node ordering 一个更差的节点排序： *MaryCalls, JohnCalls, Earthquake, Burglary, Alarm.*

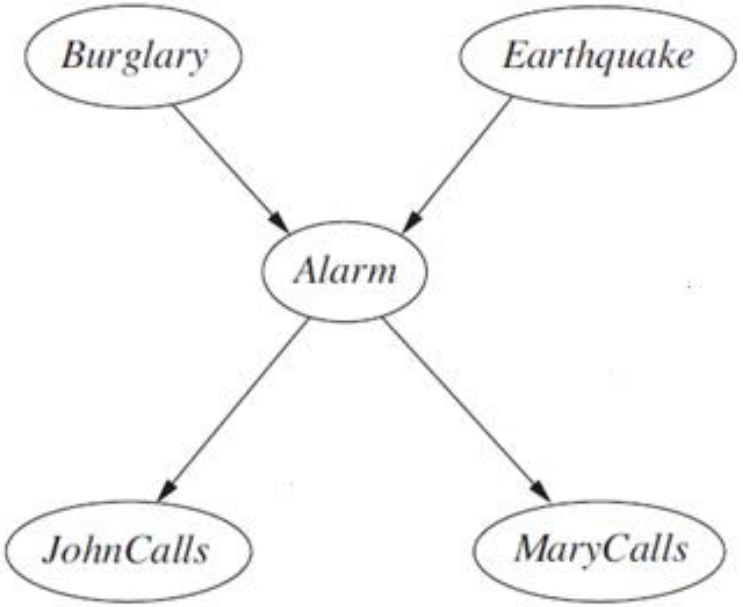
Conditional Independence Relations 条件独立关系

Numerical semantics: 数值语义

a node is conditionally independent of its other predecessors, given its parents.

一个节点，给定其父节点后，条件独立于其它前趋节点。

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid parents(X_i))$$



Example 举例

$$P(j, m, a, \neg b, \neg e)$$
$$= P(j \mid a)\, P(m \mid a)\, P(a \mid \neg b \wedge \neg e)\, P(\neg b)\, P(\neg e)$$
$$= 0.90 \times 0.70 \times 0.001 \times 0.999 \times 0.998$$
$$= 0.000628$$

where j, m, a, b, e stand for 其中j, m, a, b, e代表

JohnCalls, MaryCalls, Alarm, Burglary, Earthquake.

Conditional Independence Relations 条件独立关系

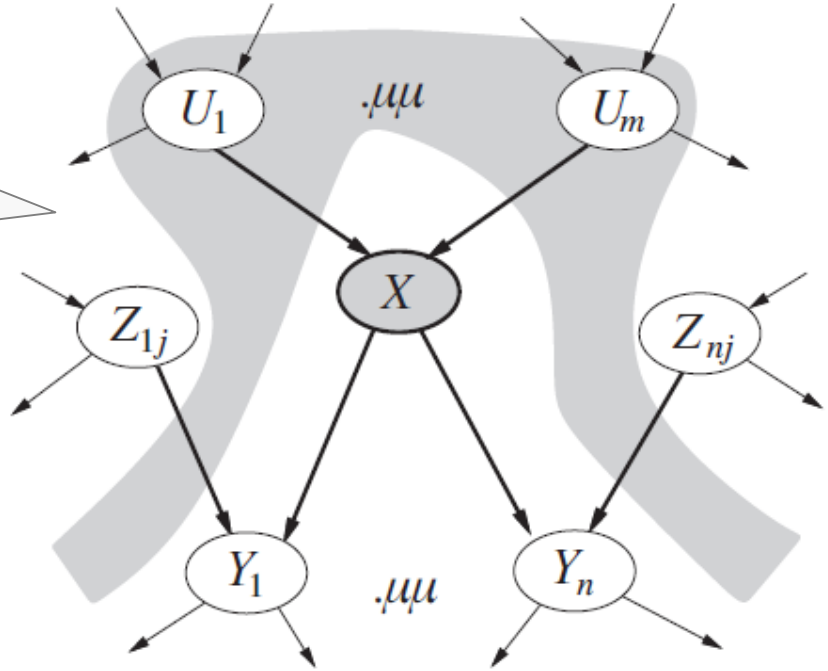
- Topological semantics: 拓扑语义

  each node is conditionally independent of its non-descendants, given its parents.

  每个节点，给定其父节点后，条件独立于它的非后继节点。

A node X is conditionally independent of its non-descendants Z_{ij}, given its parents U_i (shown in the gray area).

节点X，给定其父节点U_i（灰色区域所示）后，条件独立于它的非后继节点Z_{ij}。



Numerical semantics ⇔ Topological semantics

数值语义 ⇔ 拓扑语义

Conditional Independence Relations 条件独立关系

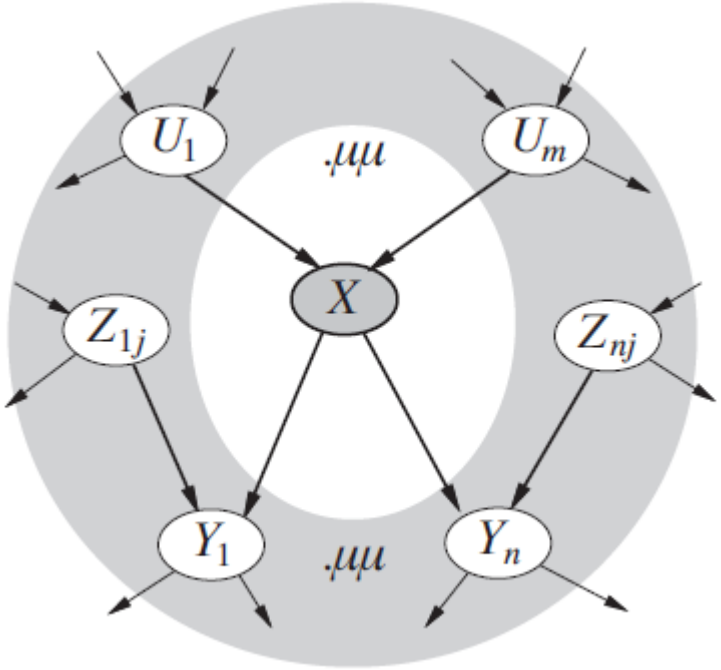
Markov blanket 马尔科夫覆盖

a node is conditionally independent of all other nodes in the network, given its Markov blanket (i.e. parents, children, and children's parents).

一个节点，给定其马尔科夫覆盖后，条件独立于网络中的所有其它节点（即：父节点、子节点、以及子节点的其它父节点）。

A node X is conditionally independent of all other nodes in the network given its Markov blanket (the gray area).

一个节点X，给定马尔科夫覆盖（灰色区域）后，条件独立于网络中的所有其它节点。



Thank you for your attention!

AI