# 物联网事件处理与融合感知

## 引言

随着物联网IoT（Internet of Things）与人工智能AI（Artificial Intelligence）领域的发展，人工智能逐渐赋能物联网，增强物联网的智能能力，面向未来万物互联时代的到来。AI带来的高数据处理能力，与物联网带来的海量数据，构成一种极强的AIoT技术应用场景，在工业生产、智慧家庭、智慧城市等领域得到了广泛应用。物联网体系中的感知层带来的大数据问题，需要物联网中间件与应用层具备数据与事件处理的能力，才能对数据与事件进行有效的管理与利用，体现万物互联的信息与价值。在物联网技术中，复杂事件处理与环境上下文融合感知是两项关键技术。复杂事件处理用于从分散的系统中提取信息并构建和管理信息，事件是该处理技术的核心。环境上下文融合感知是利用物联场景中的时间、空间与事件上下文信息，提供性能与效果更优的感知能力，增强对物联网所在环境、场景行为的语义理解。这两项技术十分契合物联网领域诸多场景的基本任务，属于当前物联网领域的研究热点。

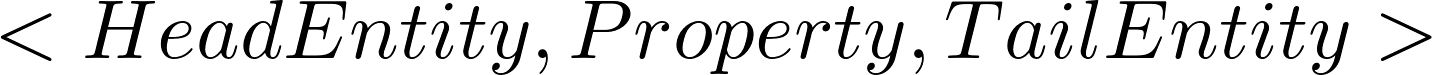
本章内容面向物联网事件处理与环境上下文融合感知技术。其中，物联网事件处理包括两个部分，分别是事件描述与数据处理、事件流检测与事件共享；环境上下文融合感知技术包括两个部分，分别是时空语义的数据融合、环境上下文感知技术。事件描述与事件处理部分，根据物联网的特征与需求，对物联网事件与事件流进行定义，并对物联网数据抽象的描述方法进行介绍；事件及事件信息的背后支撑者是数据，因此介绍和讲解其涉及到的关键数据预处理技术与事件处理框架。事件流检测与事件共享部分，核心在于复杂事件处理技术，以及事件共享机制。以上两部分提供了物联网主要的事件描述与处理方法基础，物联网事件包含丰富的时空语义信息与环境上下文信息，物联网技术要求对这两类信息进行有效利用，环境上下文融合感知提供了这类能力。时空语义的数据融合部分，解答何谓时空语义数据，对时空语义进行描述与约束，揭示关键的时空语义融合算法。环境上下文感知技术部分部分，介绍环境上下文感知技术的关键问题、生命周期、建模与决策推理模型关键技术。

## 事件描述与事件处理

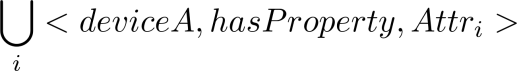
### 物联网数据抽象描述方法

物联网正在推动基础架构中具有唯一性、可识别性的嵌入式计算设备的互联。面对海量的可连接设备，一个实际的物联网系统可能从成千上万个互联设备中收集大量数据。这种数据爆炸需要高度灵活的数据模型，才能收集数据的过程与结果进行有效利用。数据模型可以让一个物联网系统具备监视、处理、优化和分析数据以获得对数据的洞悉，进而做出更好的决策或任何其他可行的结果。

在之前的章节中，我们对物联网的语义信息表达技术进行了介绍。对感知数据进行语义建模，是物联网数据抽象描述方法的核心。以Ontology技术为核心数据模型，将物联网系统与场景中的各个“物”与“物”的属性归纳为语义实体Entity。每个语义实体包含诸多属性，可以采用三元组的形式描述实体之间的关系：



三元组中分别为头实体、属性关系、尾实体。采用三元组的形式化描述，当头实体HeadEntit代表物联网系统中的“物”，属性关系代表语义“hasProperty”即拥有某种属性，尾实体TailEntity代表某种具体的属性概念时，可以用一些列三元组的组合，抽象出“物”的属性的形式化描述，这里假设“物”为deviceA，则可以用以下描述表征deviceA的属性：



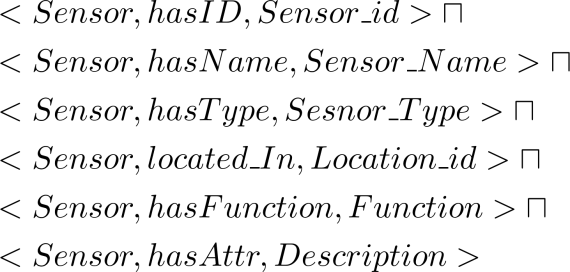
其中Attri为可选的属性，通常包括功能、类型、数据、位置等。以下是对常用的实体以及属性表示和描述进行总结。

|  |  |  |
| --- | --- | --- |
| 概念 | 描述 | 示例 |
| Entity | 头实体与尾实体，记录抽象的概念。一般对物联网场景的描述离不开传感设备、位置、时间、数据值、属性描述等。 | * Sensor * Device * Description * Location * Value |
| Property | 描述头实体与尾实体之间的关系，通常包含概念之间的转移逻辑。 | * hasClass * hasProperty * hasValue * locatedIn * hasAttr |

举一个具体的例子，对于一个传感器:

/private/var/folders/y7/_dxbgjfd16g8grhzs9syx0980000gn/T/com.kingsoft.wpsoffice.mac/wpsoffice.xPlmAYwpsoffice

括号内代表其拥有的相关概念和实体，则可表示为：



每个实体的数据可以分为静态数据（名称、固定位置、固定属性等）与动态数据（时间、测量值等）。在本书前文的感知数据语义部分有更多介绍，不再赘述。

### 物联网事件与事件流

在拥有了对物联网数据抽象的能力后，可以将物联网事件定义为传感器或设备触发而产生的物联网数据。

**1. 物联网事件**

**物联网事件event**: 物联网系统中，由传感器或设备等感知实体被触发，进而获取到的包含状态属性（时间属性、空间属性）与设备属性等信息的数据。

根据对物联网事件的定义，物联网事件具有两个重要特点，即触发性与包含属性的数据特性：

1. 触发性：来源于物联网传感层的特性，传感设备与传感网络具备监测能力，受外部与内部环境的作用触发产生数据。
2. 数据的属性特性：只有包含一定的状态、设备属性信息的数据，才是有效的事件，这些包含的信息我们希望物联网系统能够理解与运用的语义信息。

物联网的数据处理常面向数据流。在一个时间无限延续的环境中，触发的事件产生了排布在时间序列上的一个个数据，通常被成为事件流。海量的物联网事件数据所形成的事件流，作为数据流的一种，具有大数据所常有的6V（Volume, Velocity, Variety, Veracity, Valence, Value）特性，除此之外事件流相较单个事件，包含更多的语义与上下文信息。事件流同时也带了了更多挑战，这些挑战体现在相较处理单个数据，事件流需要考虑物联网的非均匀分布特点，解决数据传输过程中由硬件、网络、存储、计算等带来的数据偏差与事件乱序现象，这些数据包含一定的误差，数据不能按真实产生的真实时间戳形成有序输入输出。

**2. 物联网事件的分类**

由于物联网数据的语义隐藏在数据中，因此需要物联网系统具备对象的语义表达能力。具备语义能力的物联网系统处理对象为事件流，通过多个事件流筛选、聚合等实时计算产生语义更丰富的事件，称为复杂事件。根据事件本身的语义复杂性，将事件分为原始事件、基本事件、复杂事件。下面分别对三种类型事件进行介绍。



图 物联网事件分类

原始事件（primitive event, PE）。原始事件来源为物联网感知层中的感知设备。感知设备对目标对象的感知，产生一些列具有时间戳、空间位置、设备属性的结果。这些结果一部分能被直接利用，但考虑到感知设备获得的原始数据需要被进一步处理，以增强其数据质量、提取更多有效语义信息，所以往往不被直接利用。原始事件的属性通常由感知设备和感知目标共同决定。一个包含感知设备、感知目标、感知数据、感知时间的原始事件可以进行如下的形式化表示：

wpsoffice

基本事件（basic event, BE）。基本事件是由原始事件构成的事件流，一般作为上层复杂事件的输入。其一般可以包含简单的语义信息，表示一定语义目标对象的时空关联行为。相较于原始事件，基本事件更多被用户定义，是一些列用户感兴趣的监测结果，这样的监测结果往往并不复杂，能够被简单的规则所表达，因此也为用户对基本事件和事件参数的定义提供了可行性。

复杂事件（complex event, CE）。事件是系统中的重要情况，复杂事件是一种聚合事件，通过使用事件算子，将基本事件或复杂事件进行复合产生而产生的。复杂事件抽象上层语义对象之间的关系，以基本事件的事件流为核心。

### 物联网事件处理

物联网系统中包含三类事件，原始事件、基本事件与复杂事件。不同层次事件的事件处理目标存在差异，一般情况下，原始事件处理主要需求为对原始数据的清洗、对数据偏差的修正；基本事件处理考虑将原始数据依据一定规则形成中间数据，这部分中间数据是上层复杂事件处理的输入，而这些规则往往面向众多简单事件中的一些具有特定语义的事件的提取过程与抽象过程；复杂事件处理则是对多个原始、基本事件构成的事件流进行筛选、聚合等实时计算。

本节主要介绍原始事件处理和复杂事件处理概述，介绍事件语言和事件处理过程中的表达式与操作符。根据上节定义，基本事件的处理主要由用户兴趣和用户规则决定，因其自定义性较强，为了不失一般性不在本节介绍。

#### 原始事件处理

物联网系统中感知设备对数据的采集、传输、收集集中工作，由于感知设备的感知精度、存储性能、网络传输性能之间的差异性，为原始数据的可靠性带来挑战。为了提供可靠语义的原始事件，原始事件处理需要解决各属性数据上的偏差。

考虑到物联网的开放特性，其面向的环境之间的差异可能是巨大的，针对具体场景的传感问题往往需要考虑特殊的原始数据的处理方式。为了不失一般性，本节提出和介绍在任何场景都可能用到的原始数据与原始事件处理方法，可以为特殊场景的原始事件处理设计提供参考。首先在下面的表格中给出概念性介绍，其次对其中涉及到的具体数学方法与处理方法进一步介绍。

表 原始事件处理

|  |  |
| --- | --- |
| 原始事件的属性与特征的明确 | 明确原始事件应包含哪些属性、哪些特征，这些属性与特征是否连续，是否为类别信息。 |
| 缺失值处理 | 缺失值检查，采用适合的方式对缺失值进行弥补，保证数据完整性。 |
| 噪声处理 | 原始数据噪声极大的影响其他任务的结果，需要通过滤波等方式去除噪声。 |
| 类别型信息的编码 | 离散的类别信息需要通过编码的方式，使传输和后续处理成为可能。 |

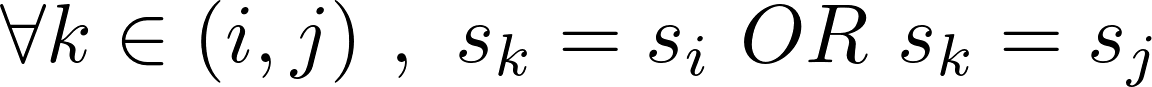
##### 缺失值处理

缺失值产生的原因，主要是因为物联网场景中，感知设备精度与工作状态无法得到保障，存在可能性不能在合适的时间做出正确响应，感知设备可能面临失灵、失效等复杂情况。物联网系统中常用低功耗、无线通信设备，这些设备终端为了功耗和性能的节约，可能采用一些不包含认证过程、不具有可靠性的协议，因此从网络上带来数据丢失的可能。在现实中，无线低功耗设备发出的数据分组丢失是常见状况。因此，即便物联网系统尽可能保证设备正常工作，对缺失值的处理仍然是必不可少的，原始事件处理后续的复杂事件处理往往需要数据具备一定的完整性与可靠性。

缺失值常与时间相关。无论是感知设备工作机制，还是通信协议的机制，造成的原始数据流在某个时间应该存在感知结果，但实际缺失感知结果的情况，是本节面对的缺失值。对此问题进行抽象表达，对原始数据流对应的时间戳wpsoffice，两个相邻时间戳之间的事件上的原始数据是缺失的数据。对于中间的缺失值，以下是三种缺失值补充方法。

1. **最后观测原则**

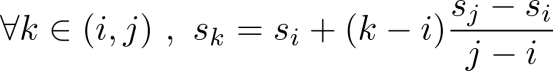
最后观测原则认为两个事件之间的缺失值，应该与最近一次观测的结果保持一致。为观测到的情况代表没有产生新的事件，因而事件的结果没有发生变化。假设wpsoffice，其中i<j，是两个时间戳，两个时间戳之间的值为缺失值wpsoffice，那么采用最后观测原则补充缺失值应为：



其中前者采用向前最后观测原则，后者采用向后最后观测原则。

**2. 线性插值**

对于时间i与时间j之间的缺失值，采用线插值表示为



**3. 非线性插值**

非线性插值不是一种插值，是许多种不是线性插值方法的统称。其中三阶样条插值是常用的插值方法。

**缺失值处理方法比较**

缺失值处理的效果因物联网场景而异。线性插值与非线性插值对于数值型属性的处理效果较好，最后观测原则对于类别型属性则更合适。下图是3种缺失值处理方式在一定数据上的效果，红点表示原始事件的真实值;3种不同颜色的曲线中，蓝色曲线代表最后观测原则，黄色曲线代表线性插值，绿色曲线代表非线性插值（三阶样条插值）。这些原始事件在时间上是不等时间间隔的，前6个事件之间时间间隔较大，其余事件事件间隔较小。将3种处理方法的比较结果总结在下表种。

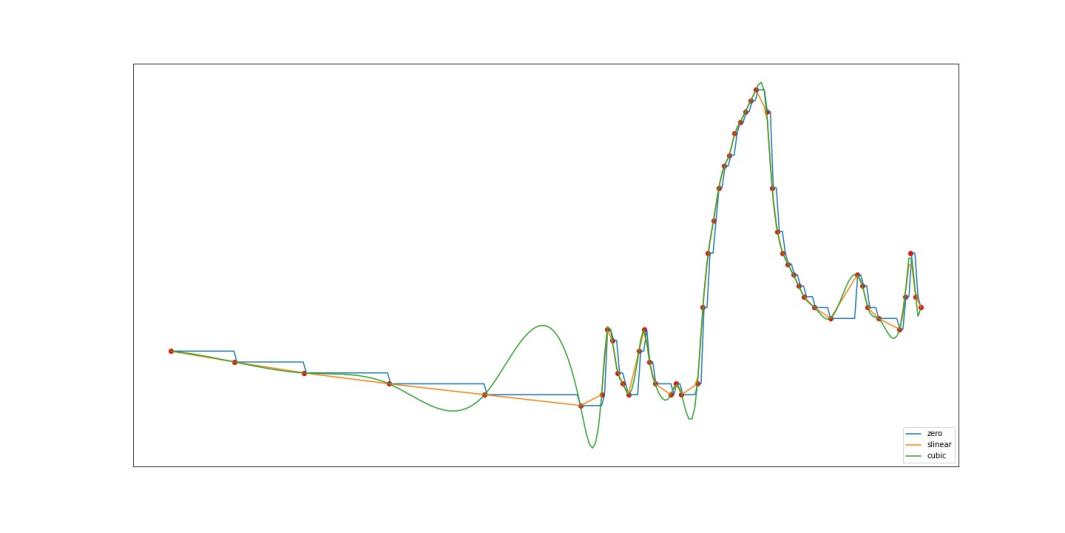


图 3种缺失值处理方法的效果

表 3种缺失值处理方法比较结果

|  |  |  |
| --- | --- | --- |
| 缺失值处理方法 | 优势场景 | 劣势场景 |
| 最后观测原则 | 类别型属性；  长时间间隔事件； | 短时间间隔场景；  精度要求较高的场景； |
| 线性插值 | 数值型属性；  长时间间隔与短时间间隔事件； | 产生复杂数值事件中缺失值较多的情况； |
| 非线性插值 | 数值型属性；  短时间间隔事件；  产生复杂数值事件中缺失值较多的情况； | 长时间间隔事件，容易产生过拟合现象； |

##### 噪声处理

原始事件伴随的原始数据常充满噪声，物联网诸多应用场景——比如广泛应用的可穿戴设备，在设备上部署的加速度计与回旋计常能采集到与动作无关的原始数据。这些无关原始数据可能影响后续任务，比如当使用一定的机器学习模型对动作数据进行处理时，无关数据影响整个识别过程的学习、识别与泛化能力。噪声为人工智能赋能物联网带来了障碍，源于人工智能往往以数据驱动，对数据的可靠性存在一定的要求。物联网事件处理，需要尽可能的在贴近数据来源的层次，对噪声进行处理，提高原始数据的可靠性。噪声处理的方法在信号处理等学科中被广泛介绍，其因信号的类型和层次而异，常涉及频谱变换、频率信号分析、滤波等。

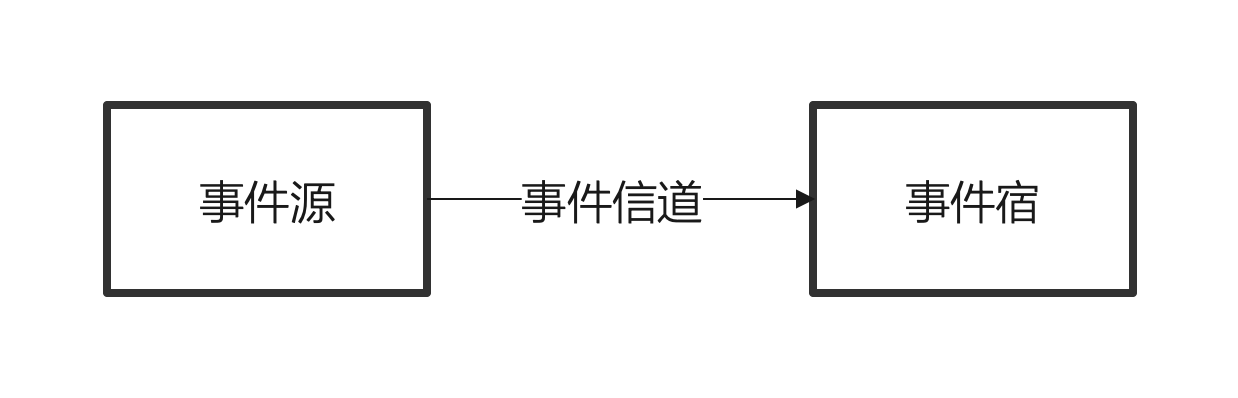
本节的噪声处理面向感知设备采集的结果。对于原始事件与相应的原始数据的噪声处理，通常对物联系统中的噪声进行一定的假设。物联系统中的噪声分为系统噪声和随机噪声。系统误差来源感知设备与感知网络所处环境，以及环境特殊因素。随机误差来源于感知场景中发生的随机事件、感知设备精度、感知传输过程等。

**1. 系统噪声**

系统噪声是比较棘手的噪声，其往往由于对物联场景考虑不周，而导致获得的原始数据与真实情况之间存在一定程度的偏差。举一个简单且有些极端的例子，使用某个感知范围为1米的传感器监测是否有人通过，但其安装位置与人常经过的位置距离在1米以上。这样的失误反应在原始事件上，监测结果为一直无人经过，无论数据层面进行何种处理，都无法解决其错误结果。所以消除系统误差，往往需要优化感知系统，这是一项复杂的工作，往往伴随大量的实验与测试。

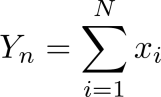
**2. 随机噪声**

相较系统噪声，随机噪声对感知结果的影响是一种时间和空间独立的随机过程，不受时间、空间对幅度、波形、相位进行随机影响。物联系统中的感知设备，感知获得原始事件结果存在许多子过程。一个原始事件从事件源处产生，经过事件信道最终达到事件宿，在事件宿处产生为原始事件。



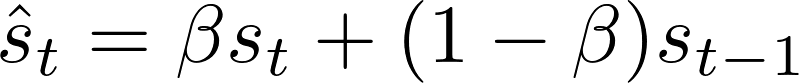
原始事件传递模型

事件源、事件信道、事件宿中又包含许多过程，而这些过程又因具体物联系统的不同而存在很大的差异性。考虑其中的随机噪声问题，假设在事件传递的过程中，子过程i对原始事件的某一项原始数据产生的影响为xi，于是原始事件传递过程中的N个子过程对原始事件中该项原始数据产生的影响为：



由于这些因素很多，子工程N被细分时，N的数目相对较大。子过程对原始数据的影响往往是随机的、人们无法控制的、时有时无、时大时小、时正时负。Yn记录了这些噪声带来的误差之和。由于单个子过程的影响十分微小，这些子过程的噪声往往服从正太分布和均匀分布。根据林德伯格-莱维中心极限定理，当N趋向无穷，服从均匀分布的子过程噪声之和弱收敛于正态分布，而正太分布之和也为正太分布。于是，原始事件中的原始数据适合用正态分布去逼近该变量的分布。因此，对于单传感器的随机噪声，常采用滤波进行处理；对于多传感器，则常用以卡尔曼滤波为基础的方法进行处理。

一种简单的滤波方法，采用相邻两次原始事件的原始数据进行滤波，通过控制参数wpsoffice进行加权求和，得到



当β为0时所有时刻全部等于0时刻。该滤波器就可以简单的削弱随机噪声对结果的影响，使原始事件结果中的突变更少、结果更加连续和平滑。

#### 复杂事件处理

复杂事件处理是物联网事件处理的一个重要研究领域，是物联网事件驱动框架的重要组成部分，作为一种实时数据的处理方法受到大数据领域的大量关注。在物联网应用系统中，依据用户需求对事件进行语义提取、语义转换和语义抽象的过程，往往集中在复杂事件处理技术。如果说原始事件处理是以数据为中心的处理技术，那么复杂事件处理技术是以事件为中心的事件处理方法，基于在线事件流对数据进行建模，提高事件作为处理对象的处理效率，也是多个基本事件符合形成语义丰富的复杂事件过程。

复杂事件处理技术可以追溯至RAPIDE[[[1]](#endnote-0)]，该工作提出了复杂事件处理的事件层次结构。在物联网场景中，现实世界的活动可以表示为子活动，这些活动产生、映射到物联网系统，形成事件流和事件云。David Luckham等人在1998年的工作[[[2]](#endnote-1)]中提出了事件分层及分层视图。对复杂事件系统进行层次结构抽象，复杂事件对业务分析建立是啊金层次结构模型，使用复杂处理规则还原活动中分解丢失的信息，利用事件过滤、聚合、模式匹配等方法将抽象程度较低的事件流融合构造出抽象程度较高的、信息密度较大的事件流。

复杂事件处理分为两个层次：

1）事件层：主要功能为事件语言与查询、事件表达与操作；

2）事件流层：主要功能为事件过滤、聚合、模式识别；

本节针对事件处理，所以本小节介绍复杂事件处理中的事件层，下一大节的事件流处理部分介绍复杂事件处理的事件流层技术。

##### 事件语言与查询

复杂事件处理要求对事件进行过滤、聚合等操作，发觉事件的约束，对事件进行关联，提供事件的表达与查询能力。物联网语义技术常采用资源描述框架（RDF）与网络本体语言（OWL）对事件及资源进行描述。

面向传统的传感器网络的非本体语言中，SASE[[[3]](#endnote-2)]语言是具有代表性的事件语言。该语言最初用以对RFID无线传感网络的事件进行复杂事件过滤、事件关联和模式匹配，将相关事件转换到匹配的特定模式上，为外部的监测应用提供服务。通过基于时间的约束和基于事件值的约束，SASE对特定事件进行过滤。SASE主要的语言结构为下：

EVENT <event pattern>

[ WHERE <qualification> ]

[ WITHIN <window> ]

该语言中，event pattern表示事件表达式，由基本事件、复杂事件、和事件操作符组成；WHERE子句通过逻辑判断，对qualification构成的简单的参数化的谓词进行判断，该谓词表达通过关系运算符连接构成；WITHIN子句限定了一个时间窗window，与事件的生命周期相关。

事件语言的一个特点就是支持扩展。任何事件语言都不应该是完全封闭的，可扩展性提供了优化能力。举例而言，SASE事件语言基础上可以进行一定的扩展，比如引入如下语言结构：

[ FROM <stream> ]

[ EVERY <time interval> ]

其中FROM子句提供输入流名称，stream表述输入流数据；EVERY子句用来确定执行周期，time interval表示执行查询的周期。这样的可扩展性使得事件语言在面向复杂任务时，具有优化自身表达能力的任务适应性。

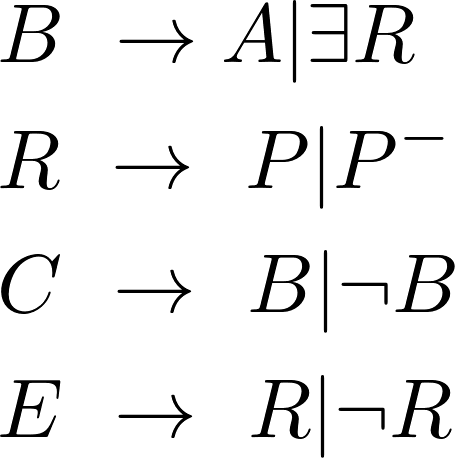
以SASE为代表的传统复杂事件处理，可以看出其特点为基于实体-属性模型，复杂事件处理系统根据用户给出的schema对数据进行理解和分析。比如SASE语言中，被<>包含的部分往往是一系列逻辑运算符与实体的组合，就是典型的schema。

随着语义技术的发展，领域内越来越多的研究采用资源描述框架（RDF）和网络本体语言（OWL）对物联场景进行描述，包括物联场景中的事件。RDF使用SPO三元组表述任何事实，其形式正如本章第一部分物联网数据抽象方法中介绍，这里SPO分别代表（subject，predicate，object）。视subject/object为节点，predicate为边，则RDF构成一张图。RDF图中的节点可以是国际资源标识符IRL、字面值以及空节点。相比实体-属性模型，RDF文档可以更灵活的描述半结构、非结构化的数据，支持高效的融合操作。

RDFS（RDF schema）在RDF进行了扩展，允许用户使用自定义的类和属性，其关于数据模型的集合TBox和包含的事实集合ABox构成了知识库。OWL是一种基于RDF描述的逻辑语言，支持了数据建模和推理。新兴的复杂事件处理，利用RDF文档进行事件表示，使用事件处理本体语言构造本体库对事件语义进行描述，复杂事件处理引擎利用本体描述推断事件所对应的复杂事件处理规则，对事件进行进一步处理。

处理复杂事件的语义技术方面，针对事件描述采用事件处理本体语言和事件处理规则语言。事件处理本体语言使用语义本体结构对事件处理逻辑和整体框架进行描述，事件处理规则语言使用过程话的方法描述事件处理的算法细节。

**事件处理本体语言**，可以基于OWL生命逻辑进行表达。采用语义化事件表示模型，通过本体库提供事件数据的语义信息和领域知识。下面介绍集中描述逻辑：



其中A表示原子概念，P表示原子角色，P-表示简单角色的逆；Ｂ表示基本概念，基本概念可以是一个简单概念或者由遗存关系表示的概念；Ｒ表示基本角色，基本角色可以是一个简单角色或者简单角色的逆。Ｃ表示概念，概念可以是一个基本概念或者基本概念的补集，Ｅ表示角色，角色定义为一个基本角色或者基本角色的补集。其中对描述逻辑进行建模时常使用到的RDF词汇有一下几个：

* rdfs:Class 类在本体模型中表示一类包含相同属性/约束的对象的集合；
* rdf:Property 属性在本体模型中表示类之间、实例之间的二元关系；
* rdf:subClassOf 表示类之间的继承关系，是一种具体的rdf:Property
* rdfs:domain 描述属性的定义域；
* rdfs:range 表示属性的值域；
* rdf:Thing 本体模型中包含所有实例的类；
* rdf:Nothing 定义空集；

通过不限于上述的词汇，可以设计满足描述逻辑所需要的本体模型。

**事件处理规则语言**，可以融合SPARQL的部分特定，提供基于结构化查询语言的事件处理语言。事件处理规则语言常包括以下几部分：

1. CONSTRUCT triple\_template\_def

定义规则输出，三元组模版表达式。

1. FROM stream\_def

FROM语句定义规则依赖的数据流，事件流可以指定特定事件窗口、长度以及包含简单事件过滤规则。可以引入JOIN语句对多个事件流进行关联查询。

1. WHERE condition

定义查询对象的过滤条件。

1. GROUP BY

对查询结果进行分组聚合。可以配合SELECT语句以及聚合函数进行结果聚合。

1. ORDER BY

指定结果的顺序。

1. LIMIT

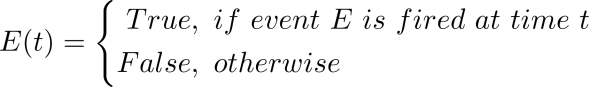
对结果进行限制，一般是数量限制、顺序限制、表达式限制。

其中较为重要的是FROM语句，语句中包含事件流表达式，提供事件类型、规则、模式的匹配过滤。

##### 事件表达与操作

事件表达通常由事件表达式构成。事件表达式是使用事件代数来表示复杂事件与子事件的结合关系的表达方法。物联网场景复杂多样，数据具有时空关联性，导致物联网事件间的关系多态化。事件操作符表达事件之间的关系，体现在事件的时间和逻辑上，因此大致分为两类：一类表达事件的持续事件，一类表达事件的逻辑执行方式。从事件的执行方式，事件相关的操作可分为描述自身状态、描述多个事件之间的关系。事件之间的关系包括时序关系、逻辑关系。

事件表达式的核心在于关系与属性的描述。关系和属性被成为特征时，事件表达式是离散映射，从时域映射到布尔域，采用特征函数的形式化表达如下：



基本事件操作符，功能涵盖描述事件的发生顺序、发生逻辑关系。基本事件操作符的形式是由用户定义的，但其功能往往涵盖以下几种，这里采用一些形式化描述。

1. SEQ(;)

事件E1和E2的顺序发生可以表示为E1;E2，(E1;E2)(t)表示该命题为True。

1. NEG()：用以表示没有检测到事件的发生。
2. AND()：表示事件的共同发生。
3. OR()：表示事件中只有一个事件发生。
4. WITHIN()：事件在某个限定范围发生。

事件聚合函数是对事件的操作，下面介绍一些常用的聚合函数。

1. SUM： 对事件进行求和操作。
2. AVG：对事件在给定维度或属性取平均。
3. COUNT：事件在给定维度或属性进行计数。
4. MAX：事件在给定维度或属性的最大值。
5. MIN：事件在给定维度或属性的最小值。

## 事件流检测

上节中指出复杂事件处理分为事件层和事件流层两个层次，并介绍了事件层相关技术。结合事件层技术，解决事件形成事件流所带来的挑战，是事件流层技术的主要目的。事件流层的主要任务是事件流检测，常用以应对物联网场景中常出现的复杂事件乱序问题。

### 事件流检测基本问题

物联网场景中的硬件故障、网络连接障碍、网络拥塞、传输与存储延时等问题，为物联网事件流的感知、存储、转发、执行带来挑战。随着流数据的应用不断扩大，各种应用根据各自的需求可能会处理来自不同数据源的数据。应用所依赖的数据源之间存在复杂的依赖关系，复杂事件处理系统的事件流层也需要针对此进行处理。

由物联场景的特点和物联应用的多元数据依赖特点，物联网复杂事件处理、事件流检测面临的诸多挑战中，事件时间乱序问题和分散来源事件分布式处理组成物联网事件流检测中的双流问题，即时间流问题与空间流问题，只不过这里的空间流可能是实际物理空间也可能是虚拟的数据空间。双流问题由于其不同特点，采用不同的解决思路和处理方法。

1. 时间流问题：时间乱序

时间乱序问题在事件流中的体现，通常是时间戳乱序问题。时间戳乱序可能会引起以下问题：

A. 可能引起异常匹配；

B. 可能因乱序事件的延迟到达而产生错误的事件删除；

C. 可能引起无法判断复杂事件是否发生；

频繁发生的乱序事件会大大降低复杂事件处理和事件流处理的效果，导致无法检测出复杂事件。错误的事件检测往往导致进行错误的执行或无法执行有效操作，从而导致物联网服务的失效。

1. 空间流问题：分散来源

一般复杂事件处理系统接收数量巨大且来源分散的事件流，需要采用分布式集群流处理平台来处理数据。影响运行效率的两个方面主要是事件源的匹配方法和查询的执行策略[[[4]](#endnote-3)][[[5]](#endnote-4)]。事件源匹配问题，需要能够处理数据量大、数据价值低等特点的数据流的匹配技术；查询执行，则需要对查询策略进行优化，以特定的策略与倾向指定查询计划，提高系统运行效率。

### 复杂事件流检测模型

在实际的事件流检测系统中，由于通过感知设备到复杂事件的过程中带来乱序性、海量性、强不确定性等特征，海量多源、乱序、不确定事件流是事件流检测中的重点，以适应物理系统的海量数据处理需求。复杂事件检测方法在事件流上的应用，主要在3类事件流上进行，分别为有序事件流、乱序事件流和多概率事件流。有序事件流存在一些共性的检测方法与常用技术，并存在许多针对共性技术的改进工作，以提高反应速度、解决检测时间长、检测效率低的问题。针对物联网这个特定环境下的乱序事件流通用处理方法尚无，目前的研究主要是面向特定任务的优化处理。对多概率、不确定性事件流的方法，主要在有序事件流的共性方法上进行改进。

#### 空间回收机制及其驱动机制

在复杂事件的事件流层检测过程中，满足NFA状态的事件，通过链表长期存储在内存中等待与之匹配的事件的到来。但随着检测的进行，大量被存储的中间结果可能导致系统的不稳定。时间窗约束要求只有在某一时间范围内成功复合的事件序列才有可能被输出，所以没有必送对那些过期事件进行长期保存。为了使这些无用数据不占用磁盘空间，使这部分未使用的存储空间真正成为系统的待分配空间，就必须采用空间回收技术将这些不满足条件的事件从存储区删除，使空间得以释放，供系统或其他用户使用。

内存空间需要清理通常在以下4种情况发生时：

1. 获取的原始事件不是触发事件，需要将该事件从缓存区中删除；
2. 子结点的定时机制触发，删除该子结点所在的子链表；
3. 不满足事件约束的新触发子事件所在的子链表删除；
4. 事件序列成功匹配时，子结点所在子链表的删除；

**脉动事件**：常将上面4种情况成为脉动事件，每一种情况的发生成为一个“脉动”。脉动事件的产生与很多因素有关，脉动事件的产生往往伴随对存储事件的哈希链表的修正。

空间回收机制需要一定的规则作为驱动，以驱动对脉动事件的检测。一般的驱动方式分为两种，即时间驱动机制、事件驱动机制。下面对驱动机制进行介绍。

1. **时间驱动机制**

系统根据确定的时间步长进行推进，当达到下一个步长时刻，进行一次更新。在一个时间步长内，不考虑是否有脉动事件发生。一个事件步长内可能没有脉动事件发生，也可能有多个脉动事件发生，其结果都在下一步长时刻到来时获得检测结果。

频率的变化对空间回收机制检测结果具有影响。随着步长的增加，脉动事件检测结果的准确度呈现下降趋势，但检测效率呈现上升趋势。反之，随着步长的缩短，脉动事件检测结果的准确度呈现上升趋势，但检测效率呈现下降趋势。系统的效率和准确度，呈现相互对立的矛盾。

1. **事件驱动机制**

事件驱动机制指由脉动事件产生更新。每个脉动时间的发生，都会让系统进行一次更新。与时间驱动机制相比，不会再出现多个脉动事件重叠的情况，准确度最高。系统中发生脉动事件的频率不确定，利用事件驱动机制可以避免一些不必要的更新，但如果脉动事件发生的频率较低，会降低空间回收机制的检查频率、进而降低其效率。

1. **混合驱动机制**

时间与事件驱动机制有各自的优点，在脉动事件的时间步长较大时，时间驱动可以有效提高系统的效率，而事件驱动则表现出更好的延迟性能。混合驱动机制就是以时间驱动机制为基础，对固定时间步长内发生的事件进行评估。当检测到脉动事件发生时，以脉动事件触发更新；若到达下一步长时刻未检测到脉动事件，则在此步长时刻对事件进行清楚回收内存空间。采用混合驱动，可以在脉动事件发生时进行实时更新，也可以在时间步长取较大时依旧保持准确度。

#### 面向多源数据的事件流检测模型

许多物联场景中包含容量大、来源多样化、需要快速相应的数据特点，使得要求系统直接从海量信息流中直接获取符合自身要求的信息。

多源海量数据流的处理，要求更强的复杂事件构造与提取能力，应对在海量数据流中的查询。Petri网常被用来代替在事件流检测中常用的NFA，以从多源数据流中进行原子事件检测。结合Petri网和Hash表的事件流检测方法，主要包括以下四个步骤：

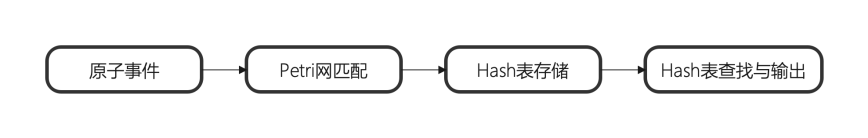


图 Petri网+Hash表的事件流检测步骤

1. 原子事件：这里特指从多源数据流中提取基本事件；
2. Petri网匹配：使用Petri网对原子事件进行匹配；
3. Hash表存储：将匹配原子事件作为中间结果，使用Hash 表结构存储；
4. Hash表查找与输出：通过使用Hash表查询来查询有关事件序列并输出检测结果；

#### 检测原理及过程

从事件流检测步骤来看，该检测方法的工作原理可以分为两部分，前半部分涉及从多源原子事件中通过Petri技术匹配原子事件，后半部分利用Hash表结构存储、查询和输出原子事件序列所对应的复杂事件。

具体检测流程图如下图所示。

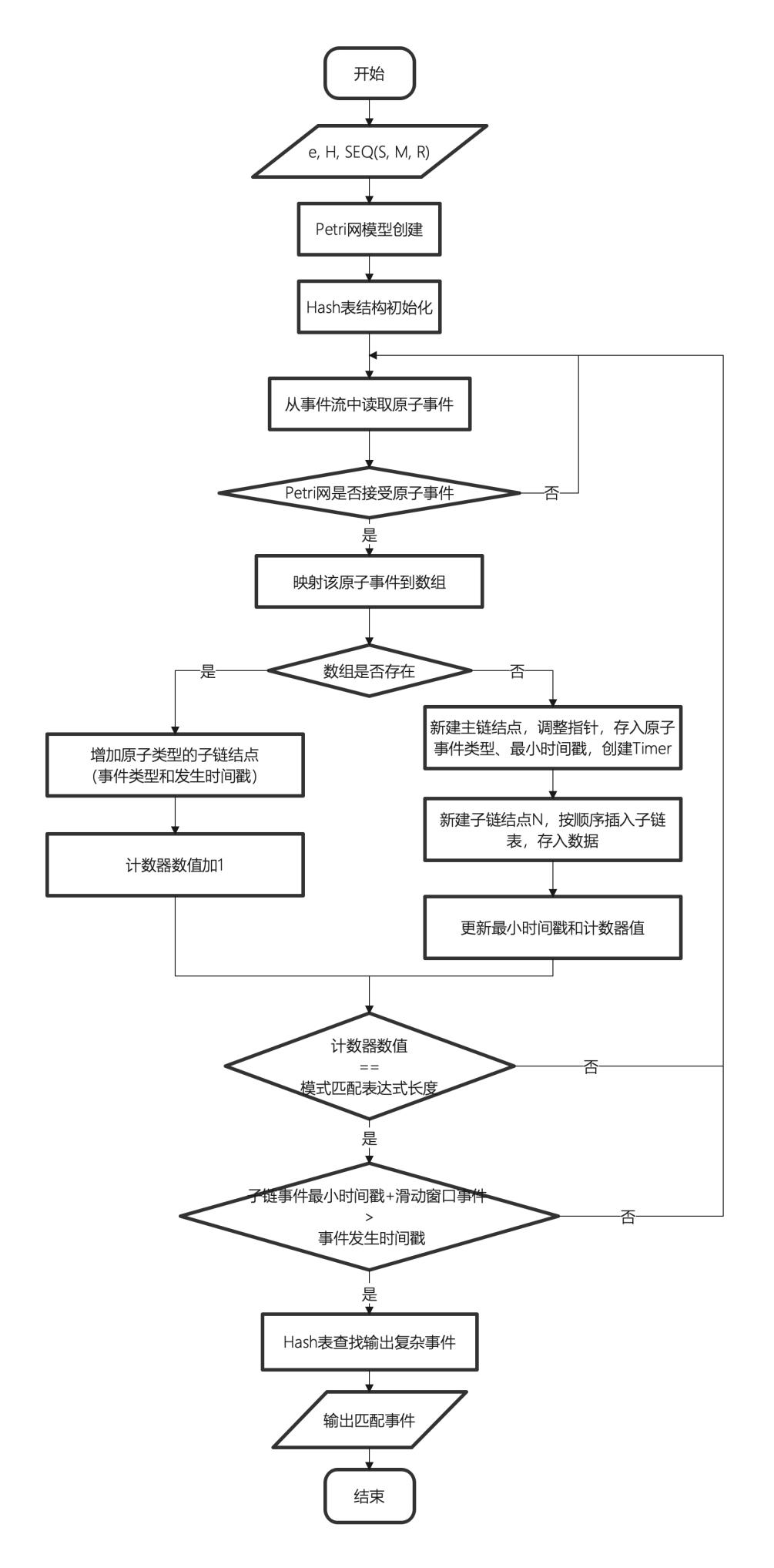


图 Petri-Hash算法检测流程图

下面具体解释算法检测步骤。

1. 输入

算法的输入，e指事件，H指Hash函数，SEQ(S, M, R)值的是复杂事件，其中简写为事件的感知（S）、存储（M）、转发（R）。

1. Petri网模型创建

通过已知的模式匹配的正则表达式创建对应的Petri网。

1. Hash表结构初始化

计算已知的模式匹配的正则表达式长度，创建新的Hash表并初始化。

1. 原子事件读取并判断Petri网接受度

将原子事件从原子事件流中读取，并判断原子事件能否被Petri网接受；若能接受，则继续执行，否则继续从原子事件流中读取下一事件。

1. 数组映射与Hash表更新

若上一步Petri网接受了原子事件，将读取的事件通过Hash函数映射到数组上，对比数组中是否已经有这项。若有，则在子链结点上插入一个子链节点，同时主链节点的计数器数值加1；若没有，则在主链结点上插入一个主链结点，在新插入的主链结点上加入一个子链结点，最后在主链节点中更新最小时间戳和计数器数值。

1. 计数器数值比较

比较计数器数值与给定的模式匹配表达式长度是否相等。若不相等，则返回原子事件读取步骤；若相等，则继续执行下一步骤。

1. 时间条件判断

比较该子链事件的发生的最小时间戳加上滑动窗口与此时事件发生时间戳。若大于，则继续执行；否则，返回原子事件读取步骤。

1. 输出

通过Hash技术查找输出相关原子事件，获得检测结果。

## 时空语义的数据融合

通过多样化、海量的传感器实现信息采集和信息交换的物联网中，物联网感知层作为物联网技术的前端，其感知的原始数据存在高度的冗余性[[[6]](#endnote-5)]。冗余性的重要来源是数据的时空相关性，海量数据的大量冗余大多由此产生，而时空相关性又是物联网的另一个显著特点。数据中包含的时空特性，是物联网数据体现出的重要语义。因此，描述和处理物联网时空语义特性具有重要意义。

在物联网时空语义数据的处理方面，主要研究问题在于时空语义的描述和时空语义特定任务的处理。时空语义描述的重要性不断的被历史研究所证明。Ioan Toma等在工作[[[7]](#endnote-6)]中指出了物联网的信息处理过程中语义技术的重要性。相较于应用语义技术解决传感器网络中的数据处理问题，时空数据起初是地理信息系统GIS中的常见概念。在GIS中已有许多时空数据模型，可以解决时空数据的描述。一些研究者利用GIS中的描述方法，来处理例如RFID信息在事件和位置上的动态关系[[[8]](#endnote-7)]，以及基于轨迹查询的RFID时空数据压缩[[[9]](#endnote-8)]。当时空语义不仅局限于RFID传感网络，真正进入物联网全部场景中时，时空语义数据面临着许多挑战，例如，如何描述物联网的时空语义数据以体现物联时空相关特性、如何描述时空关系、如何处理时空数据中的语义信息以最大化利用等等。本节对以上面临的挑战，介绍相关的解决技术，特别是对时空数据的语义化描述和建模进行探讨，对时空特性和处理技术进行研究。

### 何谓物联网时空语义数据？

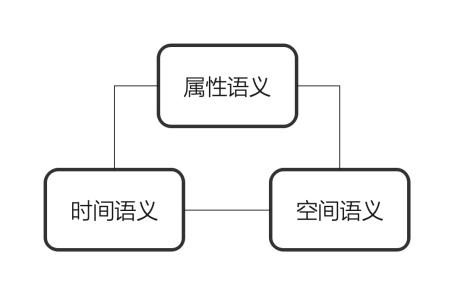
#### 时空语义数据概念

随着互联网、物联网与大数据技术的发展，各种类型的业务每天产生着TB到PB数量级的数据。这些数据来自不同规模的物联网系统，比如城市规模的交通、气象、地质、天文数据，地域规模的农业、工业、商业数据，家庭规模的小区、停车场、智慧家庭数据，个人规模的健康监测、体质检测、生活辅助数据。由于业务规模、数据采样方式的差异，不精确的、非结构化的数据成为主要构成部分，同时也体现了物联网多源性、数据多样性。这些关于人类现实生活的数据被记录的过程中，数据中蕴含这巨大价值，有待相关从业人员的挖掘。

在数据被记录的过程中，无论数据有何意义，往往至少包含两方面的信息，即时间信息与位置信息。传感技术和移动互联网所提供的数据往往是时空数据。比如，GPS与北斗卫星系统在24小时提供着高精度的位置、速度信息，能够提供高精度的民用定位，让城市中的人类活动能够留下精确的位置轨迹数据，这些轨迹数据具有位置精度和精细的时间粒度，能够较为完整的反映人类的活动轨迹，从而为研究者和服务方提供挖掘潜在时空语义的可能。

时空数据，是活动在时间和位置上留下的时空信息。就人类活动而言，定位传感器收集个体移动，产生海量的时空轨迹数据，描绘了移动对象的时空信息及对象与地理环境之间的交互，提供了对象在时空环境中的移动特征、行为偏好、活动规律。由此，时空数据延伸出时空语义的概念。在数据科学中，语义通常指数据内部及数据自身与现实世界之间的关联[[[10]](#endnote-9)]。这种关联往往是一种映射关系，将人无法直观理解的数据大小、数据属性、数据变化等映射到人能够理解的思维逻辑，最终体现为人类对客观世界的认知与规律刻画。举个最简单的例子，不断上升的大气相对湿度预示着降水是人之常识。在物联网场景中，感知设备获得的包含时间、空间的相对湿度数据，如果不加以限定条件和外部知识，则完全无法判断其是否包含预示降水的语义概念；当我们加上对获得的相对湿度的语义描述与一定的外部知识，则可以利用这样的时空数据预测未来的降水时间、降水区域。从传感器获得的系列数据，映射到降水这一现实世界中人能够理解的现象，说明这一系列数据中包含特定的语义信息，揭示了客观世界的状态变化。这样的语义信息不是靠观察就能获得的，必须通过科学的挖掘方式进行计算。

时空语义数据，是包含能够体现世界客观规律的时空数据，其语义的重要性来自于数据对人类认知和理解世界的帮助性。与时空语义数据对应的系统是时空系统，时空系统提供三个重要的要素：时间、空间、属性。这三个要素的语义信息构成时空语义三角，如下图所示。



时空语义三角

时空语义包含时间语义、空间语义和属性语义。深层的时空语义是三种语义的结合与关联产生的。不同时空系统具有不同的反映世界的角度，在时空语义上也有不同的抽象方法和侧重。这要求时空系统具备一些共同的基础能力，同时具有个性化的时空语义挖掘能力。基础时空语义能力表现为对事物和现象建模的能力，如对象建模技术表达时空离散和连续的变化。对于应用，个性化的时空语义能力表现为在基础时空语义能力的基础上进一步抽象特定领域的时空语义，突出在特定方面的表达能力。利用时空语义能力表达的时空对象之间的存在一些关系，下表给出了关系的一些例子。

表 时空对象关系

|  |  |
| --- | --- |
| 关系类型 | 关系实例 |
| 时间关系 | 序列、延迟...... |
| 空间关系 | 拓扑、距离、方位...... |
| 属性关系 | 相似、邻接、属于、互斥...... |

时空语义数据所包含的时空数据记录了活动的时间位置信息和活动语义信息，单个记录形式通常形式化定义为一个四元组(u, t, v, a)，表示目标对象u在t时间出现在地点v进行活动a。其中，位置信息v是地理位置，可以用经纬度等定位信息或一些如距离的相对定位信息。活动a可以有多种表示形式，可以是文字、图片、功能信息描述等等，是上文提到的个性化时空语义能力的重要体现之处。

当许多单个数据汇聚，数据构成一定的组织方式，比如网格、序列。具有一定结构的海量数据之间的关联和变化则包含更多的语义信息。由此延伸出时空完整性，要求时空系统对时空语义数据的描述，在空间上面向能够表达语义的实体特征进行建模，在时间上面向完整时态发展序列建模，在时空结合上记录时空数据的生命周期和演化关系。因此，时空完整性为如何在充分表达语义的基础上如何高效、合理的组织时空数据带来了极大的挑战。

为什么时空语义数据成为大数据的核心呢？时空语义数据，作为大数据的一种类型，是对移动或固定样本的时空覆盖，突破了小样本量数据的采样性、局部性、目的性。时空语义数据所形成的轨迹，在结构、粒度、模态、广度、表达、特征、类型、质量等方面引起了研究的变革，在研究方法上逐渐要求建立基于数据驱动的结构化处理模型与方法，以实现精细建模与时空语义层次的行为模式理解和环境感知。

时空语义数据能够为我们提供一些什么呢？时空语义数据作为一种新型的资源，为我们提供了涵盖不同尺度问题的解决途径，不仅能帮助解决城市交通、城市空间监测、环境质量监测、动植物生态探究等大规模问题，还能对于移动对象的行为认知、行为模式分析、空间交互等问题提供研究手段。在智能交通领域，行车轨迹规划、交通资源分配等让交通更加便利、高效。在普适计算领域，运用签到轨迹、旅行足迹等可以对用户偏好进行分析，从而提供服务推荐和个性化服务。在气象领域，通过分析异常气象的轨迹发现其移动规律和异常模式，监测气候和实时预警。在个人领域，关联活动轨迹和人的生理特征、情感特征等，可以实现生活辅助系统、特殊人群看护、个性化推荐等等。当今社会，时空语义数据与人类已然密不可分，人类的复杂活动和复杂目的为时空语义数据的相关技术带了挑战的同时，也带来机遇。

#### 物联网的时空语义特性

上一节给出的是时空语义数据的概念，对于物联网，为何时空语义十分重要？物联网具有怎样的时空语义特性？本节针对以上问题进行展开。

物联网有多种分层结构，其中一种从技术结构的角度将物联网分为三层，分别是感知层、网络层和应用层。分层结构并不统一，没有严格的限制与规定，但是按照技术结构划分的三层结构有效解耦物联网各组成部分。在这样的视角下，物联网各层技术的发展和需求是不尽相同的。近年来，物联网感知层的发展趋势为设备种类的增多、接入数量的增加，感知和控制能力和需求达到了新的层次；物联网网络层要求提供大带宽、低延时的传输能力，对接入、通信管理、内容分发能力都有了更高要求，各种分布式系统的地位日渐提升；物联网应用层方面，新型应用层协议不断优化通信逻辑，基于机器可理解的语义技术逐步推广应用于资源描述共享和信息管理。

物联网的时空特性首先体现在空间特性，空间特性则首先体现在前端感知层。感知层由传感设备、传感网关、智能终端、接入网关等组成，这些物理设备和虚拟设备遍布在不同地理空间中，为采集的数据赋能空间语义特性。比如城市中不同地理位置的视频监控反映各自位置的车流密度，提供城市空间拥堵状况的高层空间语义信息。不仅传感设备脱离位置特性会造成空间语义丢失，在当下的物联网网络与应用层的要求下，许多网络传输设备的空间位置也至关重要。网络设备提供的不只是地理位置信息，还有网络空间的拓扑信息，这些网络空间信息对于网络层的负载均衡、任务卸载、边缘计算、边缘分布式存储等技术甚至至关重要。感知层和网络层是应用层提供服务的基础。于是，自底向上，空间特性贯彻物联网体系。

物联网的时空特性也体现在时间特性，时间特性刻画了物联网中的对象的存在时间、变化状况、时间相关性。如我们所知，物理学中的瞬时速度描述某个时刻物体运动的快慢，只有当引入多个时刻的速度时，才能延伸出描述某个过程的平均速度、加速度等信息。物联网的时间特性，首先是确定性的描述了对象存在的时间，是某件事实发生的时间证据。当然，即便这样的时间证据存在精度问题，也为我们提供了重要的信息。其次，物联网时间特性，记录着状态的变化，为状态变化背后的语义信息提供证据，如某状态变化快慢、频率，及其背后变化快慢、频率隐含的语义信息，如正常变化、异常变化。

物联网时空语义三角中还包含一种语义，那就是的属性语义。属性语义需要以时间特性和空间特性为基础，结合感知层设备的功能，发挥语义的作用。以智能手表为例，其感知获得的心跳状况的语义结果，需要以时间特性作为参考，固定佩戴位置的空间特性作为保障；否则，如果错误佩戴智能手表，或者手表无法正常提供时间戳信息，则无法得到有效的心跳状况语义结果。

物联网的时空语义特性，要求原始数据在缺省状态下，自身应具有时间、空间和设备戳，表示在特定时间、地点、设备上收集的数据。

#### 物联网时空系统框架

上文提到，与时空语义数据对应的系统是时空系统，时空系统提供时间、空间、属性的要素并构成时空语义三角。时空语义数据是抽象的概念，而时空系统是具象的概念，由实际的技术和功能组成的复杂系统。物联网具有感知层、网络层、应用层的三层结构，物联网时空系统的组成应该存在与物联网系统对应的结构关系。

为了方便理解，时空语义系统是基于时空语义数据的四层物联网信息处理系统。本节介绍物联网时空语义系统的四层结构。下图展示物联网时空系统框架，四层分别为感知层、网络层、处理层、应用层。给出了每层的名称和功能，圆角矩形是每层对应功能下一些实际技术和典型技术的示例，无法给出全部的技术的总结，受限于篇幅和技术的日新月异。下面对四层分别进行介绍。

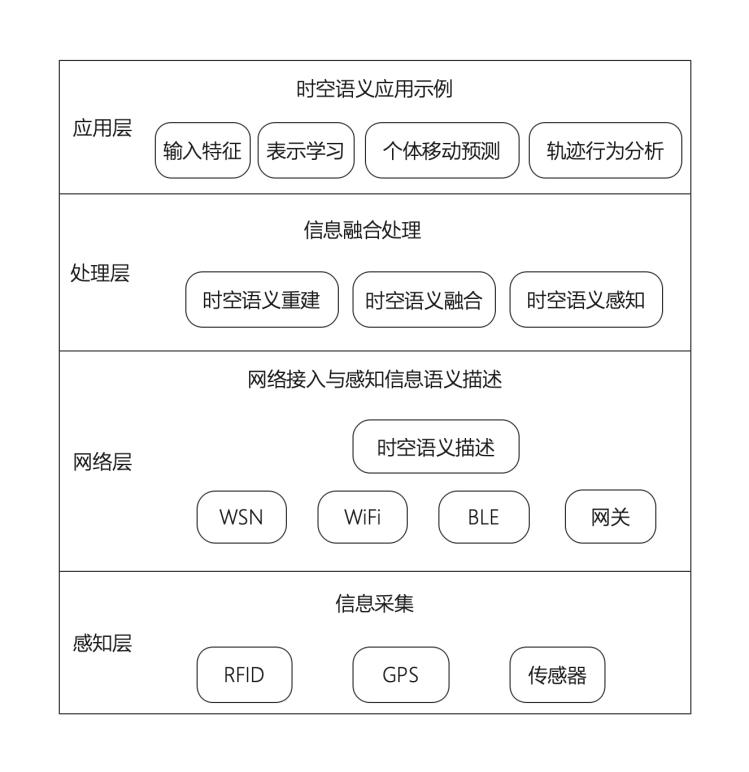


图 物联网时空系统框架

1. 感知层

感知层采集物理世界中的物理事件和原始数据。根据时空语义数据的要求，原始数据至少包含时间、空间、和设备戳，包含时间、空间和属性的语义信息。除了可以实现快速识别、实时动态感知外，感知层还可以通过语义技术实现物理实体的构建、感知设备的描述等。

1. 网络层

网络层的主要工作是数据资源的传输，从各种形式的原始数据中收集信息，通过有线或无线的方式为上层和下层提供数据和资源通路。结合感知数据的语义、网络层设备和网络的时空特性，可以进一步包装时空数据，对时空数据进行语义描述的完善。

1. 处理层

处理层主要利用时空数据的关联性，对时空语义数据进行处理工作，这里的处理包括但不限于重建、融合和感知工作。基于数据的高度冗余性，在感知和网络层完成语义话表示的基础上，可以有效的对缺失数据进行语义重建，对冗余数据进行融合处理，有效提高数据利用效果、清除时间冗余和空间冗余数据，提高资源利用效率和效果。同时，时空语义数据的预处理与部分浅层次感知也可以在此层进行，将应用层任务中存在重复、交叠的数据任务下放，提高系统的共享能力、减少重复性计算。

1. 应用层

应用层利用前期处理好的数据，完成特定任务以实现应用和服务。物联网应用层的应用涵盖范围巨大，形式多种多样。特定任务往往构成一门学科，相较于关心应用层技术，我们更关心如何利用物联网，为不同应用提供它们所需的、具有共性的时空语义数据，以促进应用层多方面技术。

### 时空语义描述

#### 时空数据类型

时空数据类型来源于时空数据建模过程，一般定义为时间数据类型、空间数据类型以及时空数据类型。对三种类型的总结如下表。

表 时空数据类型

|  |  |  |
| --- | --- | --- |
| 数据类型 | 内容 | 子内容 |
| 时间数据类型 | 时间点、时间区间、时间区域 | 年、月、日、时、分、秒 |
| 空间数据类型 | 空间位置、空间区域、空间轨迹 | 点、线、面、体 |
| 时空数据类型 | 时空拓扑、时空轨迹 | 时空点、时空线、时空面 |

#### 时空对象与属性描述

下面介绍时空对象于属性描述的相关概念、方法。

（1）时空对象

时空对象由语义特征标识自己，于特定的时间和空间状态无关。时空对象于具体时间、空间的分离体现了时空于对象的独立性，也是对时空、属性的封装。时空语义对象的一般做法是通过文档化建立用户需求的数据模型，着重突出用户对数据的理解，从模拟真实世界实体或数据库环境的角度进行相对独立的操作，以便设计出较为使用的数据库结构。

时空语义对象中的语义对象，是环境中可标识的事物，包括但不限于传感设备、环境实体等。对象的属性可以是简单属性，表现为单值属性，也可以是属性组、属性序列，也可以包含其他对象。

一般可以用四元组O={ID, S, A, H}表示时空对象。其中，ID表示空间对象的标识，S表示空间描述子，A表示属性描述子，H表示历史拓扑。这个结构把时空变化集成到了时空对象的内部，表示了时空对象的自身属性及时空对象特有的时空变化。

时空变化是时空对象O的空间描述子S、属性描述子A和历史拓扑H的集合。其中，空间描述子S表达了时空对象的空间变化，属性描述子A表达了属性变化，历史拓扑H表达了标识变化。一个时空对象在时间轴上与其他时空对象的相互关联是一个历史拓扑，也就是在时间轴上的前驱对象和后继对象的相互关联。

所谓描述子，是一定定义域上的函数，时间描述子定义一为时间域，空间描述子定义域为空间域，属性描述子则为属性域。

1. 时空变化类型

时空对象的抽象概念相对固定，但是时空对象的实例往往存在变化。对象之间的变化体现在时间、空间上，与时空关系交织在一起。时空对象的变化类型可以按照以下方面进行考察。

**时空对象的自身变化**

时空对象的自身变化，是指时空对象在包含的时空数据类型上特性的变化。因时空数据类型包含三方面，各方面包含一定的字内容，于是可以依据这些子内容的变化表征时空对象的自身变化。

举例而言，时空对象自身变化，如果未发生变化，则其状态相对静止。如果发生了变化，则可能发生了几何、拓扑、属性等方面的变化。

**时空对象时空变化特性**

时空对象在时空上发生状态变化，这样的变化包含一定的语义，反映了时空对象空间与属性变化的一些特性，例如快慢、频率。这样的特性可以称为时空对象时空变化特性，一般这种特性分为连续变化和离散变化，以及没有规律的变化。不同类型的变化反映不同的语义，三类不同的时空特性变化的含义如下。

* 连续变化：连续变化通常指在时间上不断变化的对象，在以时空为坐标的三维或高维空间中可以表示为从起点到终点的连续变化。举例而言，城市中人、交通工具的活动轨迹等可抽象为连续点的变化；车流的变化、气流的变化可抽象为连续线的变化；人群的聚散、电力与能量的转移等可以抽象为连续面的变化；复杂城市空间的变化可以抽象为连续体的变化。
* 周期变化：周期变化是重要的变化，对象的时空变化中存在的规律性是周期变化的基础。周期变化与连续变化不矛盾，变化是否是周期变化并不代表其是否属于连续变化。周期变化在时空变化中有非常多的体现，例如城市中上下班周期所带来的人员迁移、交通中的潮汐现象等。周期变化也可以分为点、线、面、体的不同维度。
* 间断变化：间断变化往往与周期变化相对，是时空上无规则的变化。周期变化可能与随机因素有关，也可能与随机因素无关。间断变化可能只发生一次，也可能会重复发生。间断变化有可能对环境产生较小影响，也有可能产生质的影响。间断变化也可分为点、线、面、体的不同维度。

1. 时空变化描述框架

在时空对象有上文介绍的种种特性和变化下，如何对时空变化、时空对象属性进行描述？基于对时空变化的描述框架是一种通过对象级和属性级两个层次来描述时空对象、时空变化的技术，能够对时空变化、时空对象、时空属性进行统一描述。

在时空对象中，对象包含两方面重要信息：对象标识、对象内部属性。其中，对象标识是确定对象的方式；对象内部属性包括主题属性、空间属性是空间对象的内部属性。于是，空间对象随时间变化可以分为以下两类：一类是对象标识产生变化；一类是对象内部属性产生变化。同时，从连续变化和离散变化的视角也可以对时空变化进行区分。于是，时空变化描述框架针对四类时空变化，如下表所示。

表 时空变化描述对象

|  |  |  |
| --- | --- | --- |
|  | 对象级：对象标识 | 属性级：内部属性 |
| 连续变化 | 对象标识连续变化 | 内部属性连续变化 |
| 离散变化 | 对象标识离散变化 | 内部属性离散辩护 |

时空变化的层次分为对象级和属性级。在时空对象的四元表示中，已经引入了描述子的概念，并且区分了空间描述子、属性描述子。在时空变化描述框架中，通过离散标识变化来表示对象级时空变化描述，通过空间描述子和属性描述子来表示属性级时空变化的描述。这样的物联网描述框架构成一颗时空变化描述树，如下图所示。

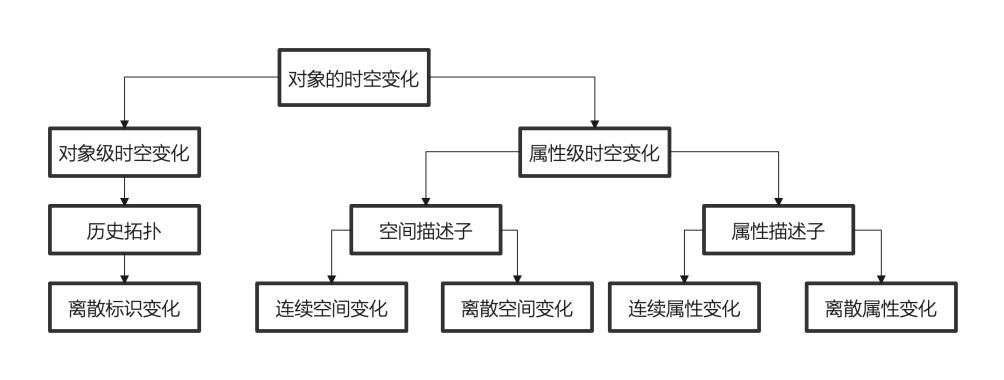


图 对象时空变化描述树

对象级时空变化采用历史拓扑标识，更进一步，采用离散标识的变化来描述对象标识变化的时空变化。离散标识变化描述了一个对象的历史演变过程，可以获得一个对象与其他对象的历史联系，历史拓扑就是这种关联的很好的表示方法。

历史拓扑描绘的是时间轴上空间对象与其他空间对象的前后关系，在每次更新时都会显式的记录对象所发生的离散标识变化类型。具体的，离散标识分为以下几类：

* 创建：创建新的空间对象。
* 合并：对歌空间对象合并成一个空间对象。
* 分裂：一个空间对象分裂成多个空间对象。
* 消亡：空间对象消失，伴随其生命周期的结束。

属性级时空变化采用两种描述子，描述子通常定义在时间域上，通过定义在时空对象不同部分上的描述子可以实现不同类型的物联网变化描述。属性级时空变化描述了单个时空对象的内部物联网时空变化，不引起对象标识变化的物联网时空变化即为单个时空对象的内部属性变化。空间描述子可用连续空间和离散空间存在状态序列来表示。属性描述子则与空间描述子存在不同，一个时空对象可能有多个属性描述子，但只有一个空间描述子。属性描述子也分为连续变化和离散变化两种。

### 时空语义处理技术

时空语义处理技术主要面向物联网时空系统框架中的处理层。处理层需要解决重建、融合的问题。为了揭示数据中的时空规律、内在联系和发展趋势等复杂语义信息，将时空数据挖掘技术引入时空语义处理层是十分必要的，也是当前各类学术研究的热点。不同于一般数据挖掘，时空数据挖掘需要考虑时空数据在时空语义描述下的时空约束，通过时空数据准备、时空数据挖掘、时空知识表达等阶段实现。

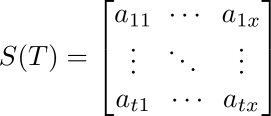
下面针对处理层的相关技术，选取典型方法进行介绍。

#### 时空语义数据重建技术

传感器数据的缺失问题常在物联网系统中发生。区别与物联网事件处理部分所讲的缺失值处理，这里的缺失考虑对真实环境的检测数据由于信息不全而导致数据发生的分组、聚类、删除或截断，是一种从结果而非来源下的考虑。在进行时空数据挖掘时，采用的数据并不是为了某一特定目的而收集，因此可能会有相关数据没有收集到、导致信息不完整，进而使一些处理完整数据的监测方法没有办法直接使用。所以，对噪声以及外界因素影响导致的数据不完整进行数据重建就尤为重要。

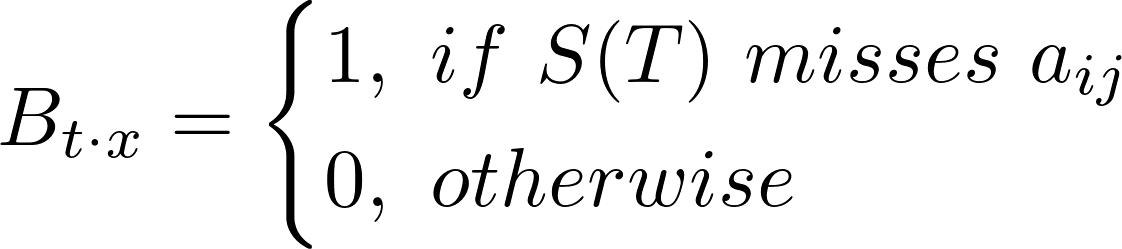
##### 缺失数据定义

对缺失值进行重建之前，首先要缺失数据。在某一监测区域内，假设区域存在a个节点，其中，每个节点有x个传感器，监测周期为t，则每个节点在监测周期内采集的某种属性的感知数据S可表示为一个矩阵：



其中，T代表在T时刻的监测数据，t是监测周期，在一个监测周期内有t行数据；x为属性个数，共有x列，每列代表一个属性。

状态矩阵B用来记录感知数据S的缺失情况，如果发生缺失则状态为1，表达如下：



重建矩阵M对在缺失状况为B的情况下，S中缺失值进行恢复得到的结果。在状态B下，M矩阵应尽可能接近原始感知矩阵S。于是，缺失数据问题是一个最优化问题，优化目标为：

/private/var/folders/y7/_dxbgjfd16g8grhzs9syx0980000gn/T/com.kingsoft.wpsoffice.mac/wpsoffice.jyRfJrwpsoffice

其约束条件为状态矩阵B，举例度量的方法采用F范数。

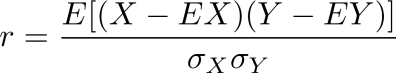
##### 传感器数据相关性分析

物联网系统中传感器数据存在相关性，这些相关性利于传感器数据的重建工作。时空语义数据特性与重建工作的关系，具有以下几个特点。

1. 时间相关性：节点传感器通常在较短时间内具有连续性和稳定性，在很短的时间内感知数据变化较慢。对于同一感知即诶点，相邻时间采集的数据比较相近甚至相同，这种性质也被成为时间稳定性。
2. 空间相关性：感知节点在物联空间中相互靠近的情况下，一些采集相同语义信息的节点所采集的数据之间具有相似性。
3. 属性相关性：属性数据之间存在关联，比如温度、湿度之间的相互影响，时空规律对时间空间的影响，可以利用相关的属性对缺失数据进行恢复。

相关的度量通常用相关洗漱进行度量。统计学常用的三个相关系数都可以用来反映传感器数据所代表的变量之间的相关程度。三个相关系数分别是Pearson系数、Spearman系数、Kendall系数。

1. Pearson系数



1. Spearman系数

通过将两个变量排序后的秩带入Pearson系数，因此也被定义为等级变量之间的Pearson系数。

1. Kendall系数

计算分类变量秩相关系数，来表示有序分类变量之间的相关性。

Pearson系数作为最基础、最常用的统计学系数，常被用来测量传感器数据相关性。

##### 缺失数据重建算法

近年来随着数据重建技术的研究，越来越多的算法被引入到利用传感器数据之间的相关性进行数据重建的工作中。基于感知数据的特征分析，线性回归与BP神经网络是两种重要的基础算法，代表着线性数据重建和非线性数据重建领域的两种经典方法。本节简单介绍两种经典算法的理论部分，给出关于物联系统数据重建的设计要素，介绍相关的重建模型。本节重点并不在于对算法理论部分的探讨，而在于算法与物联数据重建任务的关联设计、模型构建等问题。

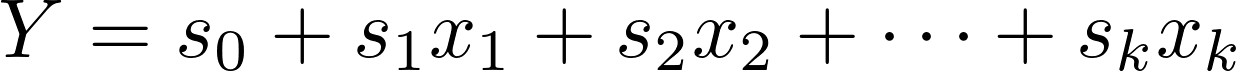
##### 基于多元线性回归的数据重建模型

当传感数据采样间隔满足较小的条件时，传感器节点监测数据往往具有一定的稳定性。依据这种时间相关性，通过时间关联，利用多元线性回归模型，可以解决一部分数据缺失问题。

**多元线性回归的理论**

多元线性回归是多个自变量与单个因变量之间存在的线性关系的回归方法。通常情况下，传感器数据手多个时间近邻数据的影响，适合采用多元回归方法。同时，在传感器感知数据采样间隔不大的前提条件下，线性模型能够较为直观的表达多变量之间的关系，线性回归可等效为一种线性滤波器，得到的回归参数即滤波器参数，这些往往对应着低通滤波器、高通滤波器等具有明确含义的模型。更进一步，由于选取时间近邻，如果选取非线性模型，往往会导致模型过于复杂，产生过拟合现象，从而导致泛化能力的欠缺。因此，多元线性回归是比较适合的方法，因变量可以是某一节点上的缺失数据，自变量为同节点邻近时刻的监测数据。

多元线性回归模型的函数为：



其中包含常数项s0，回归系数s1至sk。回归系数是因变量Y对在某一自变量上受到的影响程度，称为偏回归系数。常数项和回归系数共同构成多元线性回归模型的参数θ。

**参数设计**

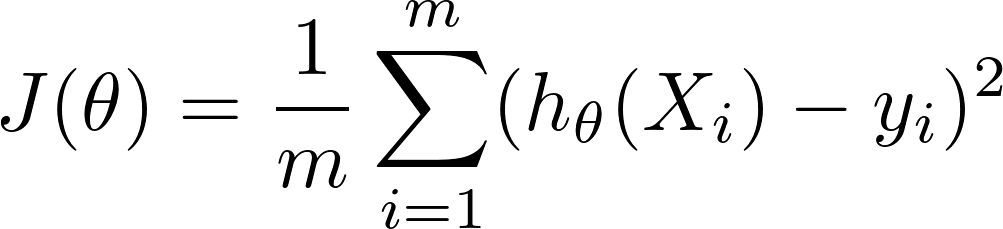
多元线性回归模型的理论并不复杂，但是，该如何选择自变量？哪些自变量是最有效的？这些问题就需要结合实际问题进行探讨，其解决难度往往很大，而且对最终得到的效果有着本质影响。

为了保证数据重建有良好的效果，近邻时刻数据的选择是十分重要的。需要选取哪些时刻的数据呢？下面给出一些基准原则：

1. 近邻时刻数据对缺失数据具有显著的相关性，且线性相关程度越高越好；
2. 近邻时刻数据之间的相关性不能过高，不能高于近邻时刻和缺失数据之间的相关性；
3. 近邻时刻数据应为完整数据；

以上原则用以有效筛选近邻时刻，保证最大程度发觉自变量与因变量的关联，同时减少自变量内部的关联，减少数据的冗余。

在确定了自变量后，进一步要确定回归模型的参数θ。模型的代价函数为：



其中hθ为回归模型，J(θ)为代价函数，一般采用均方误差。

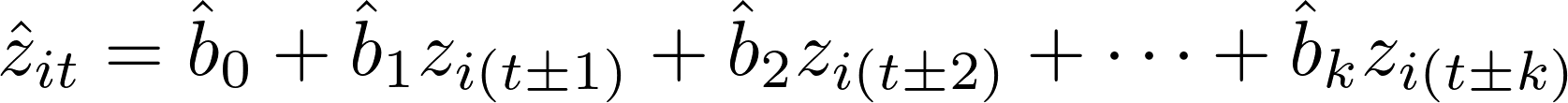
**重建模型**

当某一传感器采集数据时，可以将数据集合看成一个时间序列，对于任意时刻，如果这个时间的监测值丢失，则求其估值并使估计值与真实值之间的偏离最小，就是这部分重建模型需要解决的问题。采用形式化语言描述，对于时间序列

wpsoffice

其中Sik是这个节点在Tik时刻的观测值，对于任意时刻Tj，目标是最小化估计值Mj与Sj之间的距离| Mj - Sj |。

为了提高数据估计的准确程度，应选择空间距离比较接近C组数据的数据作为样本数据。通过多元线性回归估计：



代表通过某一时刻前后各k个观测数据来重建当前时刻数据。模型的训练常采用最小二乘法或梯度下降法。

##### 基于BP算法的数据重建模型

BP神经网络是反向传播神经网络，是训练人工神经网络的基本方法。多元线性回归能够解决线性问题，但对非线性问题缺乏解决能力。神经网络则能较好的解决非线性问题，通过添加隐藏神经元与激活函数，使模型具备拟合任意非线性函数的能力。利用BP网络可以解决物联系统数据重建问题中较为复杂的部分问题。

**BP神经网络的理论**

BP神经网络通常分为三层结构：输入层、隐藏层、输出层。其中隐藏层可以为一层或多层。输入层中的每个神经元接受一个数值数据；隐藏层完成特征提取、特征处理等工作；输出层利用特征进行分类或回归，每个神经元往往代表一个类别。反向传播（Back Propagation）是BP的由来，也是该模型的核心所在，其主要目的是将误差传回各层的参数，通过特定的策略对参数进行更新，使预测结果更接近真实结果，优化模型的性能。

**网络设计**

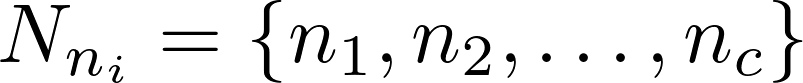
在考虑网络设计时，与多元线性回归类似的，需要考虑变量的选择、神经元规模、隐藏层层数等问题。为了使数据重建效果尽可能优化，需要通过实验不断修正神经元节点的权重、阈值，直到满足预期的误差。在设计网络时需要结合物联系统所在场景，同时考虑以下介个方面的因素。

1. 隐藏层层数和神经元个数：需要通过不断测试，适当调节隐藏层层数和神经元个数。一般针对数据重建工作，不宜使用过于复杂的网络；
2. 训练过程：为了防止欠拟合和过拟合现象的发生，训练次数需要先选取少部分实验数据对训练次数仅从测试，获得优化的训练次数后在整体训练数据上进行训练；
3. 训练样本选择：从映射关系复杂度和样本噪声角度考虑；
4. 样本数据预处理：需要进行规范化等数据预处理步骤；

**重建模型**

BP模型主要是通过不同节点在同一时刻感知数据之间的关系来构建的。与近邻时刻的概念是相对的，近邻时刻考虑的是同一节点不同时刻的观测，而本模型考虑相同时刻不同节点观测之间的关联，由此引入近邻节点的概念。近邻节点指对某节点所有拓扑空间上邻居节点构成的集合，节点与近邻节点具有相关性，近邻节点之间也具有相关性。而BP网络就是利用近邻节点集合的相关性，建立感知数据的重建模型。

采用形式化描述，假设节点ni监测数据存在缺失值，该节点有c个近邻节点，则近邻节点集合表示为



包含ni在内的集合内部节点之间均存在相关性，这样的相关性由相关性分析部分中给出的方法进行度量。

由此，BP网络模型包含以下几个步骤：

1. 数据预处理：处理异常数据，选择数量不同的近邻节点，构成不同的数据集；
2. 数据集分割：将数据集分割成训练集和测试集；
3. 模型初始化：通过均匀分布初始化所有权值和参数，设置迭代次数、最小误差等超参数；
4. 正向、反向传播：正向传播，计算各层神经元节点输出；计算网络的输出与预期之间的误差；从输出层反向传播，按照连接的权重和偏置对参数进行调整；
5. 重复训练：重复训练直到输出满足误差或达到最大迭代次数为止；

面向物联系统数据重建的BP网络，输入层神经元个数设置为数据集的感知节点数。通过相关性分析，得到某一时刻近邻节点作为输入，缺失节点的数据作为输出。通常根据实际数据规模进行选择，通过测试获得合适的输入规模。

隐藏层层数不宜过多，通常面向重建任务时，一两个隐藏层足以解决其非线性问题。隐藏层层数在四层及以上时会带来复杂性、训练时间、泛化能力上的问题。隐藏层总神经元个数为输入层神经元个数的2倍被证明是比较好的设计。

#### 时空语义数据融合技术

数据融合技术是用以对不同观测信息在一定准则下分析、综合以获得供决策使用的信息的处理技术。物联系统中，数据融合具有重大意义，其原因在于物联系统往往具有多信息源、多平台、多用户系统，任务目标通常涉及环境判断、目标感知和决策。来自多传感器的信息融合，将多传感器信息和人机界面的观测信息进行决策级融合，提取信息供后续推理使用。

数据融合技术分不同层次。数据层融合是直接在采集到的原始数据上进行融合，通常面向传感器的原始测量结果，用以提升观测结果的可靠程度。特征层融合属于中间层次上的融合，前提是传感器原始信息经过某些特征提取的过程，这些特征被综合分析和利用；特征层融合可以用以压缩冗余数据，融合结果为后续决策分析提供更加可靠的特征信息。决策层融合往往通过不同类型、不同模态的传感器观测同一目标，在每个传感器系统内部完成初步的决策，然后通过初步决策的融合获得最终判决，给出推理结果。

时空语义数据的融合，通常属于数据层融合和特征层融合，以提取数据中的时空语义特征。时空语义是包含在时间、空间、属性中的语义信息，与特征层融合的目标特性融合十分接近。本小节介绍一种从简单原始数据中融合时空语义信息的方法。

##### 物联网RFID数据融合任务

RFID系统是物联网典型系统，其感知数据较为简单，单个感知数据的信息量少。RFID系统的感知数据形式化描述为三元组：

wpsoffice

其中TagID表示物品的在系统内的唯一标识，ReaderID表示读写器的标识，Timestamp是记录的时间戳。这样的三元组记录了物品、物品在系统中的相对位置、数据产生的时间戳，包含了对象、时间、空间信息，表示一种物品的存在性语义，略去了物品对象自身属性。利用RFID数据的时空关联性，基于RFID系统感知数据的高度冗余性，对感知数据进行融合，减少存储所需要的数据量，为后续任务提供更加直观、有效的信息和特征。

时空语义数据融合的主要思路是串行的，先进行时间融合、再进行空间融合，抑或先进行空间、再进行时间融合。这样做的好处是，可以得到层次化的结果，中间结果相对可靠，利于在出现异常状况时排查融合方法中出现的问题。先空间后时间的融合方法中，首先对相同时刻、不同位置的数据进行融合，然后按照时间顺序，对相同设备、不同时间进行融合，得到最终状态。

##### 空间融合

空间融合是解决感知数据空间冗余的重要方法。感知数据的空间冗余来源于在相同时间区间内，多个感知设备的感知目标区域存在交集。以RFID系统为例，对于某个物品TagIDx，在一个时间区间内存在多个ReaderID对应的读写器对这个物品进行了感知，因而其总感知结果存在关于TagIDx的冗余，是典型的空间冗余。需要对产生的空间冗余数据进行合并和删除，减少总数据量，对利用感知数据的其他任务是十分有利的。

在RFID系统中两台读写器ReaderID1与ReaderID2产生空间感知冗余的场景如下图所示：

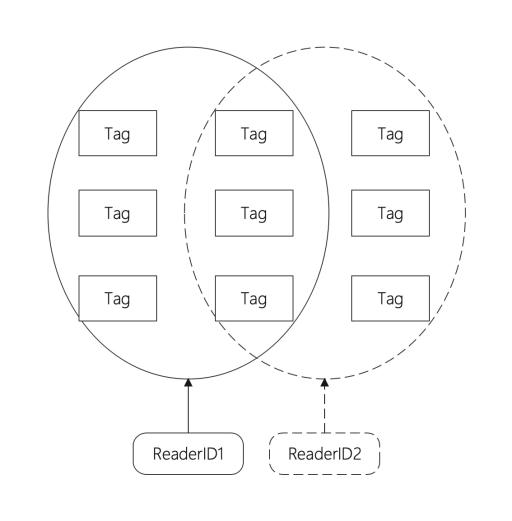


图 空间冗余数据产生场景

其中Tag代表一些列物品，实现圈出的区域为ReaderID1对应读写器的感知范围，虚线圈出的区域为ReaderID2读写器的感知范围，交叠部分表明存在的空间冗余数据。以上表示两个读写器，当情况为多个读写器时，可以类比两个传感器的情况。极端状况下，两个或多个读写器覆盖完全相同的区域，即空间完全重叠。

若干传感器的空间冗余数据处理过程中，输入为原始感知数据，输出为清理空间冗余的感知数据，数据处理算法思路如下：

1. 定义用于给文本数据赋值为0的元胞数组；
2. 求解原始数据总条数n；
3. 找出出现在多个传感器读取目标下的相同标签数据行；
4. 用元胞数组将数据标记为0，对应读取时间片也赋值为0；
5. 删除元胞数组值为0对应的原始数据的数据行。

##### 时间融合

在完成空间冗余清理后，进一步清理时间冗余。以RFID系统为例，读写器的读写范围内，物品停留时会产生多条数据，按照时间先后顺序，对同一台读写器在不同时刻读取到的感知数据进行融合处理。对同标签的多次重复读取，设定一些规则，保留相应规则的数据。例如，可以只保留同一标签最早读取到的数据，基于这样思路的时间融合处理算法输入为清理空间冗余后的感知数据，输出为清理时间冗余后的感知数据，具体如下：

1. 重新计算感知数据总条数n；
2. 找到不同时刻、相同读写器读取到相同标签的数据行；
3. 对这样所有的数据行，在每个标签下进行时间排序；
4. 找到时间片相对打的数据行；
5. 删除时间片相对大的冗余行，保留时间片最小的数据行的数据。

以RFID系统为例，通过以上的空间、时间融合，获得时空语义数据融合的最终结果。

## 环境上下文感知技术

环境上下文感知技术来源于面向物联网的上下文感知技术，上下文感知技术的英文称为Context Aware Computing或Context awareness。Context awareness特指语境、情境、环境、场景的感知，是普适计算（ubiquitous and pervasive computing）领域的核心概念之一。语境、情境与场景，主要是对客观世界的抽象。物联网构成的系统往往包含复杂的语境、情境与场景，是上下文感知技术天然的沃土。物联网包含、传递的海量传感器数据，对数据中包含的上下文信息的采集、建模、推理和分发带来了挑战，而具备上下文感知能力的系统在应对挑战的过程中表现优异。本节内容对环境上下文感知技术进行介绍，将解答什么是上下文感知技术，介绍上下文感知中信息的特点，介绍感知生命周期。推理与决策让为上下文感知技术更加智能，因此在上下文感知的生命周期中选择推理部分，着重介绍上下文推理与决策的相关技术。

### 环境上下文感知技术概述

#### 上下文感知相关概念

上下文（context）的概念被许多研究者研究过。Dey等人[[[11]](#endnote-10)]在研究的过程中比较了前人给出的上下文的概念，并对上下文进行了如下的解释：

“Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.”[[[12]](#endnote-11)]

其解释揭示了上下文是一系列实际物理概念的抽象，如果把用户与应用作为核心，那么上下文指一切与用户、应用相关的人、位置、物体等等实体与信息。并且，根据这样的解释，可以非常简单的判断数据是否属于相关的上下文。比较原始数据与上下文信息，原始数据是直接从数据源获得的数据，而上下文信息是原始数据加工后得到的，而且其数据具备一致性，包含元数据（meta data）的信息。

例如，GPS传感器产生的传感器读数可以视为原始传感器数据。一旦我们以代表地理位置的方式放置GPS传感器读数，便称其为上下文信息。因此，通常可以将传感器产生的原始数据值视为原始数据。如果此数据可用于生成上下文信息，则我们将这些数据标识为上下文。因此，我们从传感器捕获的大部分数据不是上下文信息。

从事件层面而言，上下文是一系列相关事件，以及事件中包含的逻辑关系、时序关系。事件可以简单分为离散事件与连续事件。离散时间发生的时间不连续，例如开门、开灯等；而连续事件会持续发生一段事件，比如洗澡、开车等。

上下文感知（context-awareness），也被称为sentient，在计算机应用和系统领域的含义有如下解释：

“A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user’s task.”12

其解释很好的区分了一个系统是否具备上下文感知能力。那么，如何将用户的任务、应用的任务、相关的信息有效的整合起来呢？这就要求我们对上下文具备更好的抽象能力，将上下文表达为可以用于计算的数据。上下文建模是对上下文信息的抽象，根据Henricksen等人[[[13]](#endnote-12)]的描述，上下文建模（context modelling）具有如下的含义与目的：

“A context model identifies a concrete subset of the context that is realistically attainable from sensors, applications and users and able to be exploited in the execution of the task. The context model that is employed by a given context-aware application is usually explicitly specified by the application developer, but may evolve over time.”

同时，上下文建模中涉及到的上下文信息的属性，被予以如下解释：

“A context attribute is an element of the context model describing the context. A context attribute has an identifier, a type and a value, and optionally a collection of properties describing specific characteristics.”

对上下文属性的确定性表述、抽象的表达方式，是上下文建模的核心之一。能够更好的标识上下文、表示上下文信息的类别、存储上下文信息的数值特征、提供上下文信息特点的描述能力的系统往往是更好的上下文感知系统。

#### 上下文感知的三类特性

从物联网的角度，上下文感知系统提供的能力可以被概括为三类能力：Presentation、Execution、Tagging。三类能力也被成为上下文感知系统能够支持的特性。下面对这三类特性分别进行介绍。

* Presentation

上下文可用于决定需要向用户提供哪些信息和服务。考虑一个智能空间场景，当用户进入超市并拿出智能手机时，他们想要看到的是他们的购物清单。具有上下文感知能力的移动应用程序需要连接到厨房设备，例如家庭中的智能冰箱，以检索购物清单并将其显示给用户。 这代表了一类想法，根据位置、时间等上下文来提供需要的信息，这样的信息提供被称为Presentation。物联网系统在理想情况下，具备提供随时、随地、随人、随物的服务。

* Execution

物联网系统的另一大能力为自动执行，可以节省大量的人力、物力。举例而言，智慧家庭场景中，用户从公司回家，部署在家庭中的物联网系统自动开始执行一系列服务，来为主人回家做好准备。这些服务的自动化执行依赖主人回家这一上文信息，同时，还涉及M2M通信这一物联网重要的通信技术。

* Tagging

在物联网系统中大量的传感器连接到实际物体上，被赋予了标识物体的能力。单一的传感器往往不能提供足够的信息，而多传感器的数据需要融合。在融合的过程中，上下文信息需要被收集。上下文信息需要标记到传感器信息上，以供处理与后续的理解。因此，上下文信息的标记在上下文感知计算研究中具有重要地位。

#### 物联网上下文的类别

物联网系统中的上下文信息如何进行分类，往往影响对上下文信息的提取和管理方式。直观上，上下文信息中，有些信息比较初级，有些信息则比较高级。举个例子，我们考虑两个来自不同位置的传感器的GPS数据。我们可以根据GPS数据来判断两个传感器的位置，更进一步可以计算出两个传感器之间的距离。距离这一信息，虽然属于位置相关的上下文信息，但是其是初级信息还是高级信息？这个问题的存在是因为距离是非直接测量得到的，是我们通过两个位置坐标的上文信息计算得到下文。

我们定义一个上下文信息的分类方法（即分为基本primary和辅助secondary）。该方案可用于根据操作的角度（即数据的获取与使用方式）对上下文的给定数据值（例如单个数据项，例如当前时间）进行分类。但是，在一种情况被视为主要的上下文信息，在另一种情况下可能被视为次要的。例如，如果我们直接从连接到患者的传感器收集患者的血压水平，则可以将其确定为主要上下文信息。但是，如果我们从患者的健康记录中获得相同的信息，我们将其称为次要上下文信息。因此，相同的信息可以通过不同的方法得到。理解获取信息的渠道、方法、质量、有效性、准确性是十分重要的。

基本上下文信息（Primary context）是直接获得的上下文信息，其获取过程既不依赖已有的上下文信息，也不通过传感器数据融合等操作。按照这样的规则，上个例子中的位置信息是基本上下文信息，而距离信息则不是基本上下文信息。

辅助上下文信息（Secondary context）是通过基本上下文信息计算出的上下文信息，也可以通过传感器融合的方法得到。上文中提到的距离，是通过两个传感器的位置信息融合得到的结果，那么可以属于辅助上下文信息。

上下文信息的分类并不局限于以上操作维度。从多个维度理解上下文信息，有助于更好的理解上下文信息的特点。从概念和操作两个角度，位置、身份、时间、活动等上下文信息有以下的跨分类角度的特点，如图所示。纵向表示概念维度，横向表示操作维度。可以看出，属于位置等概念的信息包含不同的操作层次，属于同一操作层次的信息包含不同概念的语义。

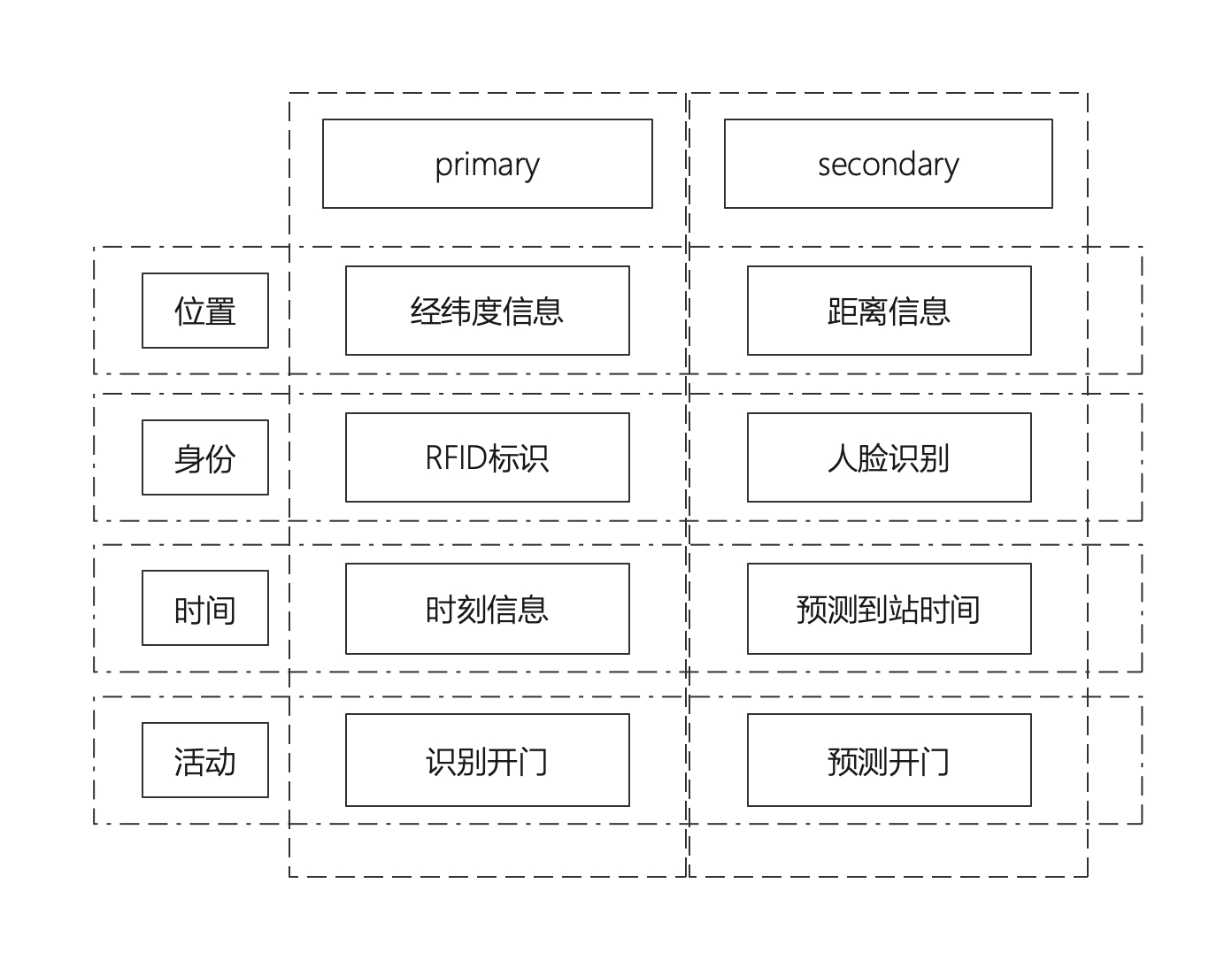


图 不同维度理解上下文信息

物联网利用上下文信息提供上下文感知服务，上下文感知的相关服务可以根据用户与系统的互动方式分为三类，分别是个性化上下文感知、被动上下文感知与主动上下文感知。个性化上下文感知服务是具备用户界面的服务，让用户能够根据自己的喜好为系统提供相关的上下文信息。被动上下文感知服务中，系统不断的监测环境、为用户提供合适的选项，如广告服务等。主动上下文感知服务中，系统连续和自动的感知着场景信息，这里的场景信息具有更高的维度，往往包含着场景中正在发生什么的语义。

### 上下文与上下文感知的生命周期

#### 上下文生命周期

数据的生命周期指在软件与系统中数据从一种形态转换到另一种形态的过程，解释了数据通过何种方式生成，直至最终被怎样使用。在上下文感知系统中，上下文具有怎样的形态与转换过程，呈现何种生命周期，是本部分解答的主要内容。上下文感知系统不局限于桌面、网页、移动应用，它已经变成一种服务，称之为上下文即服务CXaaS（Context-as-a-Service）。上下文内容的管理已经成为了软件系统中的重要组成部分，这种趋势在物联网领域更加明显。已有的网页端的上下文管理服务提供上下文生命周期内的上下文信息管理，这里的上下文生命周期被分为两类：Enterprise Lifecycle Approaches（ELA）与Context Lifecycle Approaches（CLA）。ELA往往是根据工业需求与工业标准制定的数据管理方法，具有一定的鲁棒性，但是不关注上下文自身特性。CLA则特别关注上下文管理，但是其鲁棒性往往未经测试。ELA与CLA典型的数据生命周期往往涵盖数据收集、数据管理、数据存储、数据分发、数据维护等部分，不同的生命周期还可能引入数据的恢复、反馈、分析、测试、执行等，这样的生命周期不具有统一的特性，为实际设计系统带来了障碍。特别是对于物联网系统，物联网相较传统网络本身具有结构复杂、多样的特定，因而凌乱的生命周期不适合为建立具备上下文感知能力的系统提供范式。

综合众多ELA与CLA所提的生命周期，Bernardos等人[[[14]](#endnote-13)]将上下文生命周期提炼出了三个部分：上下文获取、上下文处理、上下文推理决策。在此基础上，Perera与Zaslavsky等人[[[15]](#endnote-14)]将上下文生命周期概括为四部分：上下文获取、上下文建模、上下文推理、上下文传播。我们认为这样的概括是较为合理的，既不过于具体，也能为上下文感知系统提供层次上的范式。上下文生命周期如下图所示。

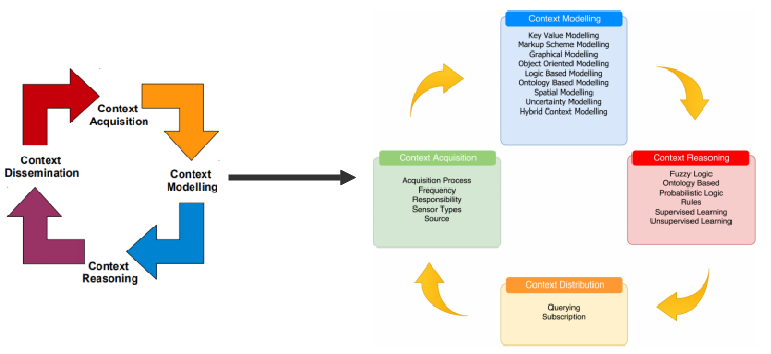


图 上下文感知系统中的上下文生命周期

这样的生命周期包含四个部分，形成了一个闭环。首先，上下文需要从各种来源中被收集，这部分对应着上下文获取。之后，收集到的数据需要通过一定的方式成为可计算的资源，这个过程成为上下文建模，这个过程中还包含了信息的表示与转换。经过建模的数据需要被计算以得到高层次的上下文信息，这部分称为上下文推理。最终，高层次与低层次的上下文信息被传播和分发到目标位置，这个过程称为上下文传播。下面分别介绍四个环节涉及的技术。

#### 上下文获取

在上下文生命周期中，上下文的获取是整个生命周期的起点，没有上下文的获取，整个系统内就不存在有效的上下文资源。上下文获取是从系统外部获取上下文的过程，是物联网中间件设计中的重要部分。获取上下文的技术多种多样，需要一定的条理才能梳理清技术的脉络。因此，本节从上下文获取的责任机制、频率、上下文来源、传感器类型4个角度，来剖析上下文获取的关键技术。

##### 上下文获得的责任机制

上下文（例：传感器数据）的基本获取方式分为2🥱类：推送（push）与拉取（pull）。

Push机制：感知层中，物理和虚拟的感知设备将数据推送到软件组件中，负责获取周期性或即刻获取感知上下文数据。周期性或即刻的推送可以通过部署发布/订阅模型来实现。

Pull机制：软件向感知层发送请求，感知层硬件设备周期性或即刻回复，从而将上下文信息传递到软件系统中。

下表总结和对比了两种机制在上下文信息获取的优缺点。

表 Push与Pull机制在上下文信息获取中的比较

|  |  |  |
| --- | --- | --- |
|  | Push机制 | Pull机制 |
| 优点 | * 感知层硬件负责感知和通讯的主要决策； * 可以即刻或周期性的感知和通讯； | * 软件主要负责感知和通讯上的决策； * 支持即刻和周期性感知与通讯； * 可以通过更大规模的数据来推理决策感知与通讯的逻辑； |
| 缺点 | * 只能通过少量数据推理和判断何时进行感知； * 当需求发生改变时感知层硬件逻辑需要重新部署； | * 需要更大的通信带宽来实现数据的拉取； * 数据收集往往涉及更长的时间； |
| 适用性 | * 可以在感知设备确定何时发送数据的场景使用； * 感知设备需要更充足的计算与推理资源； | * 可以用在感知设备不了解何时发送数据的场景； |

##### 上下文信息获取的频率

上下文信息获取，从获取频率上分为两种模式：即刻模式与周期模式。即刻模式是指事件发生的同时将上下文信息传递到软件系统中。这些事件通常不会持续发生，比如开门、开灯等事件，都存在一定的发生时刻。在这样的发生时刻后，上下文信息被立刻的获取。周期模面向会持续发生一段时间的事件。比如天气状况、季节状况、温度状况等，这些变量都存在持续性。为了监测这种类型的事件，传感器数据需要以一定的周期不断的传递数据。

下表对比了上下文信息获取中的即刻模式与周期模式。

表 即刻模式与周期模式在上下文信息获取中的比较

|  |  |  |
| --- | --- | --- |
|  | 即刻模式 | 周期模式 |
| 优点 | * 节省通信冗余、通信能量； * 一旦遇到出触发条件，更准确的数据被收集与传输； | * 按照预定的频率上下文信息在感知层与上层之间传递； * 感知设备不需要具备强大的智能或计算能力； * 允许通过收集跨时间的感知数据来获取对行为或趋势的理解； |
| 缺点 | * 需要额外的只是来判断是否满足触发条件； * 硬件与软件需要清楚的知道目标； * 当需要多来源、多类型的感知数据才能判断触发条件时是十分困难的； * 感知层硬件在计算上消耗更多能量； | * 存在通信冗余、通信能量消耗更大； * 在数据通信传输的区间内可能存在传感器数据精度的降低； * 软件层需要进行决策，数据精度的缺失可能引起事件监测的缺失； |
| 适用性 | * 硬件和软件层清晰的了解输入输出的规则与结果； | * 广泛采集多种来源、跨时间的上下文数据； * 不需要硬件、软件对输入输出结果全部清晰； |

##### 上下文获取的来源

上下文获取的来源可以分为3类，依据是上下文从系统中的来源。这3类来源分别是感知层硬件、中间件基础设施、上下文服务器。下面分别介绍这3中来源。

从感知层硬件获取上下文是指直接从传感器和相关API获取上下文。软件驱动和相关的库需要部署在传感器本地。大部分设备通过USB、COM或串口等方式部署驱动。随着无线网络技术的不断进步，可以实现不通过驱动部署就能传输数据。在物联网场景下，更多的传输方式是通过无线实现的。

从中间件基础设施获取上下文，是通过类似GSN的中间件解决方案来获取上下文数据。应用不直接从感知层获取数据，比如，一些GSN实例会从感知设备直接获取上下文信息，另一些GSN实例从其他GSN实例处通过通信的方式获取信息。

从上下文服务器获取上下文，是指上下文通过数据库、web service等渠道进行扩展与存储，外界从这些上下文存储中获取上下文。这类方式在设备只拥有受限的计算资源时变得格外可行。资源充足的上下文服务器可以获取和处理上下文。

下表对比了上下文获取中的三种来源。

表 不同来源在上下文获取中的对比

|  |  |  |  |
| --- | --- | --- | --- |
|  | 感知层硬件 | 中间件 | 上下文服务器 |
| 优点 | * 允许与硬件的直接通信； * 对硬件配置、数据获取具有更加直接的管理； | * 中间件承担主要的管理，因此管理和获取上下文十分容易； * 可以更简单、更快速的获取上下文； * 要求的技术知识更少； | * 需要最少的资源； * 获取上下文信息的速度快； |
| 缺点 | * 需要对硬件嵌入式编程和配置具备丰富的技术知识； * 需要显著的时间、精力、代价来部署； * 由于设备与应用之间的带宽受限，系统更新往往困难； | * 需要部署中间件，因此引入额外资源； * 对传感器的配置不直接； * 效率中等，因为数据需要在中间件内部传输； | * 无法对感知设备进行配置； * 上下文需要从服务器或网络中拉取出来，因此削弱了便捷性； |
| 适用性 | * 适用于小规模的科学实验； * 有限的感知设备使用情况； | * 物联网大多数场景使用； * 大量的、异构的传感器使用情况； | * 适用于显著数量的上下文获取、同时仅存在有限的部署资源的场景； |

##### 上下文获取的感知设备类型

在上下文信息获取的过程中，感知设备类型的不同对上下文存在着重要影响。这里的感知设备不仅指有形的传感器硬件设备，还指任何能够提供数据和上下文的设备。因此，感知设备可以分为3类：physical、virtual与logical。

physical感知设备往往是有形的、最常见的感知设备，这些设备自身产生感知数据。从这类感知设备中产生的上下文一般是低层次的上下文信息，低层次的信息往往缺少含义，还对微小的变化十分敏感，所以物联网系统需要理解物理世界传达的不准确、充满矛盾的信息。

virtual感知设备通常不自己产生感知数据，虚拟设备从多数据源中获得上下文信息，并且当作一个传感器来提供数据。这类设备之所以是虚拟的，因为其不存在物理的形体，常结合web service等技术来收发数据。

logical感知设备，又叫做软件传感器，结合物理传感器与软件进而使感知结果的意义更加丰富。这类设备可以由web services构成，也可以由物理与虚拟设备构成，往往是多种服务的综合。

下表对比了上下文感知中的三种感知设备。

表 不同感知设备类型在上下文获取中的对比

|  |  |  |  |
| --- | --- | --- | --- |
|  | Physical | Virual | Logical |
| 优点 | * 错误检测更加方便可行； * 缺失值处理更加容易； * 对底层传感器的部署和配置更加直接； | * 提供意义更加丰富的数据； * 提供更高层次的上下文信息； * 提供的数据未经深度加工； * 不需要处理硬件层次的问题； | * 提供意义丰富的数据； * 提供高层次的上下文信息； * 通常数据更加准确； * 不需要处理底层硬件问题； |
| 缺点 | * 硬件开发和维护的难度大； * 需要解决传感器和硬件的编程、涉及、测试等问题； * 感知数据更底层，缺少意义； | * 不容易发现数据中的错误； * 缺失值处理难度大，因为通常都是非数值和难以预测的数据； | * 不容易发现数据错误； * 缺失值处理困难； * 对数据产生的过程没有任何控制； * 可能产生许可证及其他限制所带来的费用； |
| 适用性 | * 可用于收集可观察的现象，诸如光，温度，湿度，气体等现象； | * 可用于收集无法物理测量的信息，例如日历详细信息，电子邮件，聊天，地图，联系方式，社交网络相关数据，用户偏好，用户行为等； | * 可以用来收集信息 * 成本高昂，并且不可能通过单个物理传感器直接收集，其中 * 预先处理和融合来自 * 需要多个传感器（例如天气信息，活动识别，位置识别等）； |

#### 上下文建模

上下文建模在上下文生命周期中上下文获取之后，对进入系统的上下文信息进行模型化，提供可用于计算的上下文资源。上下文建模也被称为上下文表示（context representation），也就是上下文在软件中的表达方式。上下文模型可以是静态的，也可以是动态的。静态的模型预定义一系列的上下文信息，将收集和存储的上下文按照规则进行整理。动态的模型具有更加灵活的特点，往往是数据驱动与预定义信息的结合。

上下文信息建模时需要考虑诸多问题，比如数据的异构性、移动性、相互关系，以及数据的完整性、精度等问题。通常根据一个模型表达上下文信息由两步构成：

1. 上下文建模处理：新的上下文信息进入系统，根据属性、特点、相互关系等之前上下文中包含的内容，对新的上下文进行定义；
2. 根据模型组织上下文：验证上下文建模的有效性，新的上下文整合到现存的上下文仓库中，使这部分上下文变为系统的可用资源。

第一步是上下文建模中建模的部分，但是其因素和参数相对比较主观，不同的解决方案里往往有不同的模型和模型参数。目前，对上下文建模尚未出现统一的标准。在物联网领域，主要的上下文建模方式，到目前为止有6种。这六种方式是主流的方法，也是高层次的建模方法，我们更加关注每种建模方式的概念和主要技术。下面介绍这六种建模方法。

1. 键值对 key-value modelling

键值对建模的方式是将上下文信息通过文本文件或二进制文件下的不同格式的键值对进行保存。这事最简单的上下文表示方法，非常易于在数据规模较小的情况下进行存储。但是，键值对不适合进行大规模的扩展，也不适合存储类型非常复杂的上下文。键值对的数据结构也限制了其获取模型化信息的能力。同时，键值对技术是面向应用的，比较适合作为临时存储使用，不需要太多配置用，也不涉及太多户选择。

1. 标记模式建模 markup schema modelling

此方法也叫做标记编码（Tagged Encoding），标记编码中上下文信息的存储都伴随其标签。这种方法在键值对建模的基础上改进而来，为上下文信息提供了便于检索和获取的标记，增强了数据检索的效率。通过shema定义，这种方法也支持了数据验证。一些标记模式，例如XML技术，已经被广泛用于数据存储、数据在软件中的转移、数据在软件之前的转换。相对的，标记模式不提供推理能力，上下文的互操作性、复用性在不同标记模式之间是较差的。例如CC/PP、ContextML等建模技术都是基于标记模式的建模技术，这些技术里元组也常被使用。

1. 图建模 graphical modelling

图建模使用关系船来建模上下文，UML与ORM是两种代表性技术。在表达丰富性方面，图形模块优于标记和键值建模，因为它允许将关系捕获到上下文模型中。图形建模的实际底层表示技术可能会有所不同。例如，它可能是一个SQL数据库、noSQL数据库、XML等。数据库可以保存大量数据，并且提供简单的数据检索操作，可以相对快速地执行。相比之下，不同的数据库和其他解决方案使得互操作性变得很困难。更多，数据检索机制存在局限性，例如SQL。此外，复杂的上下文检索要求可能要求使用非常复杂的SQL查询。许多当前的工具还不足以支持创建如此复杂的SQL查询语句。而且，添加上下文信息和更改数据结构也很困难。 noSQL是一种解决的趋势，随着研究和技术的进展，图建模技术可以用作上下文的持久存储。

1. 基于对象建模 object based modelling

基于对象（或面向对象）的概念用于使用类层次结构对数据进行建模和关系。面向对象的范式的优势在于封装和可重用性，作为大多数高级编程语言支持面向对象的概念，建模可以轻松集成到上下文感知系统中。因此，基于对象的建模对象适合用作内部、非共享、基于代码的、实时运行的上下文建模。但是，它不提供内置的推理能力。由于标准和规范的缺乏，面向对象设计的验证也很困难。

1. 基于逻辑建模 logic based modelling

基于逻辑的建模是将事实、表达和规则用于表示有关上下文的信息。 规则也被其他建模技术使用（例如基于本体的建模）。规则主要用于表达策略、约束和偏好，它提供了更强大的表现上下文的能力，与之前的其他模型相比，其上下文语义的丰富程度更高。因此，基于逻辑的建模，可以实现一定程度的推理能力。但是，缺乏标准化导致可重用性与适用性的降低。此外，现代化的图建模手段也可以被用来改进基于逻辑和规则的上下文表达。非技术人员也可以向系统中添加逻辑规则。基于逻辑的建模实现了从低层次信息中提取高层次的上下文信息的能力。因此，其也可以作为辅助，增强其他建模技术的能力。

1. 基于本体建模 ontology based modelling

基于本体的建模是将上下文信息通过语义技术中的本体技术进行组织和整合。不同的语义技术（RDF, RDFS, OWL）和推理能力被使用。但是，随着数据量的增大，上下文检索所涉及的计算开销、时间开销也会增长。在上下文感知计算和传感器数据管理领域，本体技术是首选技术。

以上六种上下文建模方法的总结与比较汇总在下表之中。

表 上下文建模方法的比较

|  |  |  |  |
| --- | --- | --- | --- |
| **方法** | **优势** | **局限** | **适用性** |
| **关键值** | * 简单 * 灵活 * 数据量小时易于管理 | * 与应用强耦合 * 不可扩展 * 没有标准结构和模式 * 不易于信息的检索 * 无法表示关系 * 没有验证支持 * 无可用的标准处理工具 | 可用于对有限数量的数据(如用户首选项和应用程序配置)建模。大部分是独立和不相关的信息。这也适用于有限的数据传输和任何其他不那么复杂的临时建模要求。 |
| **标记方案编码（eg:xml）** | * 灵活 * 更具有结构化 * 能够通过一定的模式进行验证 * 存在可用的处理工具 | * 应用依赖于没有标准的结构 * 当涉及到信息的多个层次时会比较复杂 * 信息检索难度适中 | 可作为中间数据组织格式以及数据在网络上传输的模式。可用于解耦系统中两个组件使用的数据结构。（例如：sensorML用于对存储传感器的描述，JSON作为一种通过网络传输数据的格式） |
| **计算图（eg：数据库）** | * 允许关系建模 * 信息检索难道相对简单 * 可以使用不同的标准和部署方法 * 能够在一定约束下进行验证 | * 查询复杂 * 需要配置 * 不同部署方式之间的互操作较为困难 * 没有标准，由设计原则决定 | 可用于长期和大容量的永久数据归档。历史上下文可以存储在数据库中。 |
| **基于物体** | * 允许关系建模 * 可以很好地整合使用不同编程语言 * 存在可用的处理工具 | * 不易于信息检索 * 缺少验证 * 与应用强耦合 | 可用于在代码层次表示上下文信息，允许在上下文允许时期进行操作。在非常短的时间内、暂时性地大部分存储在计算机内存中。也支持通过网络传输数据 |
| **基于逻辑** | * 允许使用低级上下文信息生成高级上下文 * 易于建模和使用 * 支持逻辑推理 * 存在可用的处理工具 | * 没有统一标准 * 缺少验证 * 与应用强耦合 | 可用于使用低级上下文生成高级上下文(即生成新知识)，也可用于建模事件和操作(即事件检测)，并定义约束和限制。 |
| **基于本体** | * 支持语义推理 * 允许更有解释性的上下文表示 * 强验证 * 应用独立并允许共享 * 具有强有力的标准化支持 * 存在可用的成熟工具 | * 表示可能复杂 * 信息检索可能是复杂的和资源密集型的 | 可用于对关系进行基于本体的定义，实现对领域知识和结构上下文的建模。数据可以存储在适当的数据源(即数据库)中，而不是存储在本体上，而结构是由本体提供的。 |

#### 上下文推理决策

上下文推理决策被定义为利用可用的上下文信息推导出新的知识、新的理解，也可以被解释为从一系列上下文中处理获得高层次的上下文的过程。上下文中包含的不确定信息与不准确信息是上下文推理需要考虑的两个重要因素。推理的性能表现可以通过效率、健全性、完整性、互操作性等进行评判。上下文推理包含许多步骤，我们将步骤归纳为3步：上下文预处理、传感器信息融合、上下文推理。

上下文预处理是负责清理原始数据的过程，感知层设备产生的感知数据由于传感、网络传输等过程带来的不确定行，需要预处理才能变得更加准确、完整。关于数据预处理，在之前的部分已经介绍过，在此不再赘述。

传感器信息融合是结合多传感器来产生更加精确的感知数据的过程。在物联网场景中，感知融合是非常重要的，因为传感器的数量是巨大的。庞大数量的传感来源会提供关于统一任务的海量感知数据。

上下文推理则是第三步，也是本部分的重点。推理产生高层次的上下文信息，这些信息来自于低层次的上下文信息。已有许多可用于上下文推理和决策的技术，这些技术主要是人工智能和机器学习领域的技术。决策树、朴素贝叶斯、隐马尔可夫、支持向量机、K近邻等机器学习方法，以及人工神经网络等深度方法，还有知识驱动的Dempster-Shafer理论、本体论、模糊逻辑等，都是上下文推理与决策中可以利用的方法。这些模型或方法并不局限于上下文推理，在诸多领域也发挥着巨大作用。

在物联网领域，海量的传感器在发挥着作用。使用全部的传感器与数据进行推理决策显然是不容易的。另外，研究表明，增加上下文信息的来源、传感器的数量，对推理与决策结果的性能影响是，当传感器数量上升到一定程度会出现饱和。也就是说，选择何时的传感器数据与上下文信息，对推理与决策是至关重要的，这一点在物联网领域尤为关键。

本部分将上下文推理模型总结为6类方法：监督学习、无监督学习、基于规则的方法、模糊逻辑、本体推理、概率推理。

1. 监督学习

监督学习是一种先要采集数据，然后标记数据，再利用标注数据训练模型的方法。训练好的模型导出后，当新的上下文信息产生时，可以在标签的范围内进行推理和预测。这项技术已经广泛用于移动端的感应和活动识别等任务。

具体到技术层面，决策树是一种典型的监督学习，可以从不同数据中获得分类饿的方法与结果。贝叶斯网络也是监督学习方法，使用有向无环图表示事件和关系，常被用来组合不确定性、来自大量信息源的信息并推理得到更高层次的上下文。人工神经网络是一种现代的学习技术，其本质往往是监督学习，通过模仿生物神经元系统，对非常复杂的关系进行建模并获得数据中的模式，已经普遍用于医疗保健中的监视。还有许多监督学习的方法，例如支持向量机、隐马尔可夫等，可以通过模式识别、机器学习相关书籍与课程进行学习。

1. 无监督学习

无监督学习可以在未标记的数据中找到隐藏的结构，由于使用未经标注的数据，没有错误或奖励信号可以给到反馈。聚类是无监督学习中常用的方法，可以获得数据中存在的类别信息，进而获得一定的标签。在底层传感器硬件级别和传感器网络级别的许多任务都涉及无监督学习方法，例如室内、室外定位之类的任务。无监督的学习方式还可以扩展为在线方式，实时进入系统的数据就可以进行处理。

1. 基于规则的方法

基于规则的方法是最简单、最直接的方法，直接推理得到决策结果。简单规则通常表达为IF-THEN-ELSE的格式，复杂规则则可表示为一阶规则。基于规则的方法允许直接从低层次的上下文信息中获取高层次的上下文信息，更可以结合其他方法以发挥更大的作用。

在实际中，大多数用户偏好是通过规则的方式进行编码的。规则也用于事件检测，在物联网场景中扮演重要的角色。

1. 模糊逻辑

模糊逻辑是一种重要的近似方法，在推理过程中，数据中的不确定性往往影响推理决策的结果与性能，模糊逻辑与近似能够带来极强的鲁棒性，在面对噪声的环境常能发挥巨大作用。模糊逻辑与概率推理有相似的地方，但是其置信度代表的是隶属关系而不是发生概率。在传统的逻辑中，可以接受的真值是0或1，但是在模糊逻辑中，真值的概念变得模糊，不确定性是可以介绍的。因此，真实世界的场景能够更自然的表示，大多数现实世界是模糊的而不是精确的。更多地，它允许不确定性表述的引入，例如对温度的描述就可以引入“稍微有点热”、“非常冷”的程度修饰，而不只有热和冷两种对立的状态。

在大多数情况下，模糊逻辑不能单独用作推理的技术，而是作为其他技术的补充，例如在基于规则概率、本体论进行推理时，引入模糊逻辑。

1. 本体推理

本体推理是基于描述逻辑的，描述逻辑是基于逻辑的知识表示家族中的一员。本体推理的基础是semantic web语言，主要由RDFS和OWL组成。基于本体推理的优势在于引入本体技术强大的推理能力，但相反，其烈士在于本体推理对物联网场景下的缺失值、模糊信息的处理能力并不强。规则的引入可以减弱这一劣势，因其可以从低层次上下文中产生高层次上下文信息。在活动识别、混合推理、事件检测领域，本体推理都被广泛应用。

1. 概率推理

根据事实所反映出的概率分布进行推理和决策的方法属于概率推理，它可以用来结合多种来源的传感器数据，也可以用来获得对上下文中存在的冲突的理解。大部分这类技术用来判断事件的发生，典型的Dempster-Shafer理论是依据概率的证据理论，用来结合多种证据判断事件的概率。隐马尔可夫模型是概率模型，可以对观测的情境进行建模，而不直接获得状态。

以上六类上下文推理决策的总结与比较汇总在下表中。

表 上下文推理与决策技术的比较

|  |  |  |  |
| --- | --- | --- | --- |
| **技术** | **优势** | **局限** | **适用性** |
| **有监督学习（eg:人工神经网络、贝叶斯网络、案例推理、决策树学习、支持向量机）** | * 准确性高 * 可选择的模型种类很多 * 具有数理统计理论基础 | * 需要大量数据 * 需要将每个数据元素数值化 * 特征选择有一定的挑战 * 可以是更密集的资源(处理、存储、时间) * 语义信息较少，因此可解释性较少 * 需要进行数据的训练 * 模型更复杂 * 很难获取现有的知识 | 可用于特征集易识别、可能的结果已知，并且大型数据集(也用于训练)可以用数字表示的情况。(例如:活动识别，缺失值识别) |
| **无监督学习（eg:聚类、K近邻）** | * 不需要训练数据 * 不需要知道可能的结果 | * 模型复杂 * 难以验证 * 结果不可预测 * 可以是更密集的资源(处理、存储、时间) * 语义信息较少，因此可解释性较少 | 可用于可能结果未知的情况(例如:异常行为检测，分析农田以确定种植某种特定作物的适当位置) |
| **基于规则** | * 易于定义 * 易于扩展 * 所需资源较少 | * 需要手动定义 * 由于需要手动操作，因此容易造成差错 * 没有验证或质量检测 | 可用于需要将原始数据元素转换为高级上下文信息的情况。（例如定义事件） |
| **模糊逻辑** | * 允许更多的自然表达（or允许更自然的表达） * 易于定义 * 易于扩展 * 所需资源较少 * 可以处理不确定性 | * 需要手动定义 * 由于需要手动操作，因此容易造成差错 * 没有验证或质量检测 * 可能会由于自然表示而降低结果的质量(例如精度) | 可用于需要将低级上下文转换为高级、且更自然的上下文信息的情况。。这种简化将使进一步处理变得容易。例如，控制自动灌溉系统，当系统检测到土壤是“干的”时，水将被释放。 |
| **基于本体(一阶谓词逻辑)** | * 允许更复杂的推理 * 允许更复杂的表达 * 结果更具有含义 * 存在验证和质量检测的可能 * 能同时推理数字数据和文本数据 | * 数据需要以兼容的格式进行建模(例如OWL、RDF) * 数字推理是有限的 * 低性能(例如需要更多的计算能力和时间) | 可用于知识重要的情况。例如，存储和推理关于农业领域的领域知识。它允许根据本体结构存储上下文信息，并在需要时自动推理。 |
| **概率逻辑（eg:隐马尔可夫、朴素贝叶斯）** | * 允许证据信息的合并 * 可以处理未知情况 * 存在多种模型可供选择 * 能够处理不确定性 * 能够提供具有一定含义的结果 | * 需要知道概率信息 * 只能进行数值推理 | 可用于概率已知、需要梳理来自不同来源的证据的情况。例如，从相机、红外传感器、声学传感器和运动探测器产生的证据可以结合起来检测风动物渗入农业领域 |

#### 上下文传播

上下文经过建模和推理决策后，产生的信息是新的上下文信息，这些信息应服务于相关任务。在物联网场景中，任务涉及诸多物理与虚拟节点，相关服务并不全在一起，因而上下文需要传递到相应的位置。

上下文传播，也称上下文分发，是直接了当的任务，提供上下文发送到需要使用和消费上下文位置的服务。从数据消费的角度看，上下文传播也是上下文获取，因此上下文形成了完整的生命周期，同时也说明上下文的传播与上下文的获取是相互配合的。

上下文传播主要有2种机制：query和subscription。其中query为询问模式，subscription为订阅模式。询问模式是上下文收取方向系统发送对上下文的请求。订阅模式是上下文收取方通过订阅/发布的关系与上下文提供方建立关联，系统周期性或事件触发的为收取方发送上下文信息。订阅模式中也会涉及到许多与query相关的操作，是实际中主要的上下文传播方式。

参考文献

1. [] Luckham D C, Kenney J J, Augustin L M, et al. Specification and analysis of system architecture using Rapide[J]. IEEE Transactions on Software Engineering, 1995, 21(4): 336-354. [↑](#endnote-ref-0)
2. [] Luckham D C, Frasca B. Complex event processing in distributed systems[J]. Computer Systems Laboratory Technical Report CSL-TR-98-754. Stanford University, Stanford, 1998, 28: 16. [↑](#endnote-ref-1)
3. [] Wu E, Diao Y, Rizvi S. High-performance complex event processing over streams[C]//Proceedings of the 2006 ACM SIGMOD international conference on Management of data. 2006: 407-418. [↑](#endnote-ref-2)
4. [] Prasan Roy,S. Seshadri,S. Sudarshan,Siddhesh Bhobe. Efficient and extensible algorithms for multi query optimization[J]. ACM SIGMOD Record,2000,29(2). [↑](#endnote-ref-3)
5. [] Rimma V. Nehme,Karen Works,Chuan Lei,Elke A. Rundensteiner,Elisa Bertino. Multi-route query processing and optimization[J]. Journal of Computer and System Sciences,2013,79(3). [↑](#endnote-ref-4)
6. [] 胡永利,孙艳丰,尹宝才.物联网信息感知与交互技术[J].计算机学报, 2012, 35(06): 1147-1163. [↑](#endnote-ref-5)
7. [] Toma I, Simperl E, Hench G. A joint roadmap for semantic technologies and the internet of things[C]//Proceedings of the Third STI Roadmapping Workshop, Crete, Greece. 2009, 1: 140-53. [↑](#endnote-ref-6)
8. [] Wang F, Liu S, Liu P. A temporal RFID data model for querying physical objects[J]. Pervasive and Mobile Computing, 2010, 6(3): 382-397. [↑](#endnote-ref-7)
9. [] Lim D, Hong B, Cho D. The self-relocating index scheme for telematics GIS[C]//International Workshop on Web and Wireless Geographical Information Systems. Springer, Berlin, Heidelberg, 2005: 93-103. [↑](#endnote-ref-8)
10. []刘扬,郑逢斌,姜保庆,蔡坤.基于多模态融合和时空上下文语义的跨媒体检索模型的研究[J].计算机应用,2009,29(04):1182-1187. [↑](#endnote-ref-9)
11. [] Dey A K . A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications[J]. HCI, 2001. [↑](#endnote-ref-10)
12. [] Abowd, Gregory D , Dey, et al. Towards a Better Understanding of Context and Context-Awareness. Springer-Verlag, 1999. [↑](#endnote-ref-11)
13. [] Henricksen K . A framework for context-aware pervasive computing applications[J]. Ubiquitous Computing, 2003. [↑](#endnote-ref-12)
14. [] Bernardos A M , Tarrio P , Casar J R . A data fusion framework for context-aware mobile services[C]// IEEE International Conference on Multisensor Fusion & Integration for Intelligent Systems. IEEE, 2008. [↑](#endnote-ref-13)
15. [] Perera C , Zaslavsky A , Christen P , et al. Context Aware Computing for The Internet of Things: A Survey[J]. IEEE Communications Surveys & Tutorials, 2014, 16(1):414-454. [↑](#endnote-ref-14)