软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2014,25(10):2325−2345 [doi: 10.13328/j.cnki.jos.004582]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

物联网环境下多智能体决策信息支持技术*

徐 杨[1], 王晓峰[2,3], 何清漪[1]

[1](电子科技大学 计算机科学与工程学院,四川 成都 610054)

[2](中国科学院 计算技术研究所,北京 100190)

[3](广西可信软件重点实验室(桂林电子科技大学),广西 桂林 541004)

通讯作者: 王晓峰, E-mail: xfwang78@sohu.com

摘 要: 随着物联网技术的不断发展,传感器网络得到了广泛的应用并成为信息技术领域重要的基础设施.尤其是传感网络提供的实时感知信息,为许多智能应用提供了充分的信息支持和必要的决策依据.然而,由于智能应用的实时感知信息需求通常无法转化为简单的查询请求与传感器底层查询接口准确匹配,因此,基于物联网的智能决策常常无法准确获取到决策相关的实时信息.针对此问题,提出一个基于语义覆盖网的物联网信息资源描述、推理和应用模型,并以多智能体系统决策支持为应用基础,研究了新型物联网环境下的多智能体决策信息支持技术.该技术以基于多智能体系统的团队导向规划的任务分解方法为核心,将复杂任务分解为若干简单子任务,并基于本体推理方法把子任务执行时需要的决策信息转化为精确、完备的传感器信息查询,从而实现从物联网中准确定位具体的传感器并获取相应感知信息的实时决策信息支持机制.

关键词: 语义覆盖层;本体;团队导向规划;物联网;多智能体系统

中图法分类号: TP18

中文引用格式: 徐杨,王晓峰,何清漪.物联网环境下多智能体决策信息支持技术.软件学报,2014,25(10):2325−2345. http://www.jos.org.cn/1000-9825/4582.htm

英文引用格式: Xu Y, Wang XF, He QY. Internet of things based information support system for multi-agent decision. Ruan Jian Xue Bao/Journal of Software, 2014,25(10):2325−2345 (in Chinese). http://www.jos.org.cn/1000-9825/4582.htm

Internet of Things Based Information Support System for Multi-Agent Decision

XU Yang[1], WANG Xiao-Feng[2,3], HE Qing-Yi[1]

[1](School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China)

[2](Institute of Computing Technology, The Chinese Academy of Sciences, Beijing 100190, China)

[3](Guangxi Key Laboratory of Trusted Software (Guilin University of Electronic Technology), Guilin 541004, China)

Corresponding author: WANG Xiao-Feng, E-mail: xfwang78@sohu.com

Abstract: With continuous development of Internet of Things (IOT), sensor network has been widely applied and become the vital infrastructure of information technology. Specially, the dynamic sensing information provided by the sensor network plays a key role for various intelligent applications in support of information retrieval as well as decision-making. However, since the real-time information requirements are less likely to be transformed into simple sensing queries well matching the low-level sensor query interface, it is hard for those intelligent applications to accurately obtain decision related information online from the sensors. To address this challenge, this paper presents a semantic overlay model with semantic resource description, reasoning and applications for IOT. In addition, an application for the decision making of multi-agent system is deployed to manifest how IOT information techniques can improve agents' decisions. The key of this approach is the team-oriented plan for agents' task decompositions. By decomposing the complex task into

* 基金项目: 国家自然科学基金(60905042, 61202211, 60950110354); 国家科技支撑计划(2012BAI22B05); 航空科学基金(20100580005); 中央高校基本科研业务费专项资金(ZYGX2011X013); 广西可信软件重点实验室研究课题(kx201325)

收稿时间: 2013-08-11; 修改时间: 2013-11-18; 定稿时间: 2014-02-20

simple subtasks, their information requirements can be mapped into accurate and sufficient sensor queries with ontological reasoning. Therefore, a real-time decision support system can be established so that task related quires can be accurately allocated to the sensors with best corresponding sensed information for accomplishing agents' task.

Key words: semantic overlay; ontology; team-oriented plan; Internet of things; multi-agent system

随着人工智能的高速发展,多智能体系统在智能交通[1]、军事应用[2]、应急事务处理[3]和灾难救援[4]等方面得到了广泛的应用.多智能体系统在完成任务时往往需要获取大量关于物理世界的实时信息,例如在灾难救援中,对受灾地区人员进行救援时,需要知道伤员的位置、周围环境状况和伤势情况等信息,这样才能快速地、有针对性地进行救援.当前兴起的物联网技术为多智能体系统提供了一种全新的、具备覆盖面广、实时性强等特点的感知信息服务机制,这种信息服务机制以物联网中具备互通信、互操作能力的传感器网系统为基础,向多智能体系统提供广泛、全面的实时信息,显著提高了多智能体系统完成任务的效率.

在多智能体系统完成复杂任务时,对于感知信息的查询是复杂任务执行中的子过程.然而,目前传感网中的传感器只能响应精确、完备的传感网信息查询请求,而多智能体系统的任务往往都是由高层语义所描述的抽象任务.同时,由于物联网中包括数量众多的传感网系统[5],通过收集各个传感网系统的感知信息查询接口参数,再根据多智能体的任务需求构建相应的查询请求是不可行的.因此在物联网环境中,对于面向实时决策的多智能体信息查询,存在着由高层语义描述的决策信息需求难以直接转化为面向感知节点的信息查询请求的问题.例如,一个智能体系统需要协调各个救援单位对某一建筑进行救火.对于"救火"这样一项复杂任务,现有的物联网服务系统无法直接给出完成任务所需信息.因此,如何建立有效的物联网信息查询机制、为基于物联网的多智能体决策系统提供必要的决策信息支持,成为一个关键的研究问题.

基于以上需求,我们需要构建一个物联网语义描述层,即物联网语义覆盖网(semantic overlay on IOT).该语义覆盖网提供了一个关于物联网各种资源的高层语义视图,根据该语义视图,物联网中的各个实体、对象可以通过基于高层语义的描述信息相互通信、相互操作.此外,也可实现多智能体与物联网对象之间的信息交互.例如,对某宾馆进行救火的任务,其中一个任务为"获取火灾现场人员信息",通过语义覆盖网提供的知识,多智能体系统可以获知到宾馆的管理系统中含有现场人员信息.因此,多智能体系统可以通过由语义信息描述的网络接口参数,直接查询宾馆数据库,获取现场人员信息,为现场救火的相关决策提供必要的信息支持.此外,基于物联网服务的多智能体决策支持系统还需要研究如何将一个复杂的多智能体任务进行分解,使复杂任务转化为若干可执行的简单原子任务.同时,为了保证原子任务能够顺利执行,我们需要保证原子任务中所需要的关于物理世界的信息能够通过物联网获得查询,因此,需要验证分解后的原子任务中的信息需求是否都能通过物联网获取到.如果存在无法通过物联网获取信息的原子任务,那么该分解方案将被认为是不合适的,系统应该重新规划一个新的任务分解方案.

综上分析,要实现基于物联网信息的多智能决策系统,需要解决两个基本问题:

(1) 如何对物联网进行语义建模,并构建一个以语义 Web 技术为核心的物联网语义描述覆盖网;

(2) 如何根据物联网语义覆盖层所提供的关于物联网的语义知识,对复杂多智能体任务进行有效的分解,将复杂任务转化为一系列可执行的简单原子任务;同时保证需要分解得到的原子任务,通过物联网能获取到其所需信息.

为了构建有效的物联网信息查询机制,为多智能体复杂任务执行提供必要、可靠的信息保障,本文提出了一个面向物联网资源的语义模型,并根据该模型提出了一种物联网语义覆盖层的构建方法.同时,为了有效执行基于物联网信息的复杂任务,本文提出了一种基于物联网语义知识的复杂任务分解方法,通过该方法,多智能体系统可以准确、有效地查询到物联网信息,并基于该信息完成复杂任务.

这里的复杂任务是指需要由多个智能体通过协同、合作完成的任务.本文以面向团队的规划(team-oriented plan,简称 TOP)[6]为基础,构建了一种复杂任务分解方法.基于 TOP 构建任务分解方法的理由是:TOP 能提供一个根据团队中每个个体的行为能力,并构建任务分解计划的框架.基于该框架,可以实现复杂任务的有效分解.

本文的主要贡献总结如下:

(1) 提出了一个具备静态语义和动态语义描述能力的物联网语义信息覆盖网模型;

(2) 提出了一种基于物联网语义信息的任务分解方法.

本文第 1 节讨论物联网语义覆盖网的构建问题.第 2 节讨论任务分解方法以及信息查询机制.第 3 节为案列分析.第 4 节介绍相关工作.第 5 节为全文总结.

1 物联网语义覆盖网

1.1 物联网结构

物联网是一个以传感器网络和 RFID 技术为技术基础、将物理世界中的各类实体相互联通的网络.物联网中的各类实体可以通过网络相互通信、相互操作.目前,物联网实体主要包括了各类传感网络、具备通信能力的嵌入式设备以及与网络连接的其他电子设备等.通过 Web 服务协议相互通信、相互操作,是目前一种主要的物联网实体通信方法.在此类方法下,任意接入到物联网中的实体都将自己所能提供的信息服务、可被操作的状态以 Web 服务形式进行封装,并将相应的服务信息注册到某个管理 Web 服务信息注册服务器.通过查询 Web 服务注册器,实体可以查询到满足其业务需求的其他实体,并按照服务注册中存储的关于服务调用的参数与其他实体进行通信.

本文假设所讨论的物联网以 Web 服务计算作为物联网中各类实体的通信基础.据此,可以将物联网所涉及的元素大致划分为如下几类:

(1) 具备通信能力的物联网服务实体;

(2) 注册实体服务信息的物联网实体信息服务器;

(3) 负责服务代理的 Service Broker.

其中,

- 物联网服务实体主要是传感网络中的各种传感节点,或者各种具备感知能力的嵌入式设备;
- 实体信息服务器主要由各类型的网关以及管理 Web 服务资源的服务器组成;
- Service Broker 是由智能体(agent)构成的服务代理.

1.2 物联网语义建模

通过第 1.1 节的分析可知,物联网是一个以服务计算为基础的新型网络架构.因此,物联网的语义建模主要是针对物联网中各类型服务的语义建模.进一步地,在物联网服务实体中,存在大量资源受限嵌入式设备,如传感节点、嵌入式网管等.由于受到有限计算能力、有限能量存储等因素的制约,不适合用基于 SOAP 的 Web 服务对这些实体进行服务封装.因此,当前许多关于物联网的研究工作都以 Restful 服务架构构建物联网中的服务.本文中,我们也采用 Restful 服务架构对物联网中的服务实体进行服务建模.

Restful 服务架构定义了一种构建 Web 服务的风格,它以资源为中心,将服务描述为资源,并通过基于 HTTP 的操作,实现对服务资源的互操作.一个典型的 Restful Service S_i 包括如下几部分组建:

(1) 表示 S_i 的 URI,通过该标示可以访问 S_i;

(2) 针对 URI 的一系列操作,该操作系列包括:GET,PUT,POST 以及 DELETE,其中,

- GET 表示对 URI 对应服务资源表示状态的访问;
- PUT 表示根据 URI 创建一个新服务资源,并以此 URI 表示;
- POST 表示针对 URI 所标示的资源的状态更新;
- DELETE 表示删除 URI 所标示的资源,这里需要指出,DELETE 操作仅删除 URI,而不是删除服务资源所对应的物理实体.

例:设 S_1 表示一个烟雾传感器所提供的某地点实时烟雾探测服务,那么可以通过 URI:http://smoke.sensing.xxx/sensor1 表示该烟雾探测服务,GET http://smoke.sensing.xxx/sensor1 表示对该烟雾探测服务的访问,PUT http://smoke.sensing.xxx/sensorx 表示新建一个烟雾探测服务资源 http://smoke.sensing.xxx/sensorx.其中,关于该

资源的具体细节,如烟雾传感器访问等信息,可以通过一个 json 格式消息来描述并由 PUT 操作传送到关于传感资源的相应服务器.POST http://smoke.sensing.xxx/sensor1 实现对于烟雾传感器状态的动态改变.

类似地,可以通过一个 json 格式的消息描述对服务资源的具体改变内容,如 state:off,表示关闭 URI 所对应的传感器.

由于物联网的服务资源主要由具备感知功能的传感设备构成,因此在物联网中,服务资源的上下文决定了服务的可用性.这种上下文包括了服务资源的时空特征、传感器所对应的当前状态等.例如,针对某个烟雾探测服务资源,需要知道其具体所在位置,才能确定该烟雾探测服务是否满足当前应用的需求.除了上下文信息,还需要了解对于资源的一些自描述信息,即关于资源的元信息,基于资源元信息,可以自动了解资源所提供的服务类型、服务接口、服务传递消息的具体含义等信息.

根据上述分析,我们提供了两类语义描述信息对 Restful 服务进行语义描述,这两类语义描述信息是服务资源语义上下文描述和服务资源语义元信息描述.这两类资源分别对应两个本体文件[7],可通过唯一的 URI 进行访问.其中,

- 资源上下文本体主要描述了服务资源的状态属性,包括时空状态以及能量状态等;
- 服务语义元信息本体描述了与服务相关的一些属性信息.

通过将上下文本体与服务描述本体相结合,可以实现对针对物联网中各类服务资源的语义信息的查询,推理实现服务的发现、匹配、组合等.下面我们具体讨论一下这两类本体.

传感器是物联网服务的主要提供对象,因此,我们根据传感器的上下文构建物联网服务上下文本体.同时,将 OWL-S 作为服务语义描述的本体.另外,我们针对部分应用领域建立了领域专业知识本体,如灾害救援本体、环境检测本体、智能环境本体等,为物联网资源描述,推理提供必要的领域背景知识.

以烟雾探测服务为例,我们构建了如图 1 所示的上下文本体,该本体描述了烟雾传感器 S_1 所提供的烟雾传感服务的上下文信息,其中,椭圆节点为类节点,圆形节点为实例节点.从该本体中可以看到:S_1 传感器以 15s 定时方式进行烟雾采样,S_1 为服务 SP_1 提供烟雾传感功能支持,表示 SP_1 的 URI 为 u_1.S_1 所在位置为纬度 39.9°、经度 116.3°,其当前状态为开启状态.

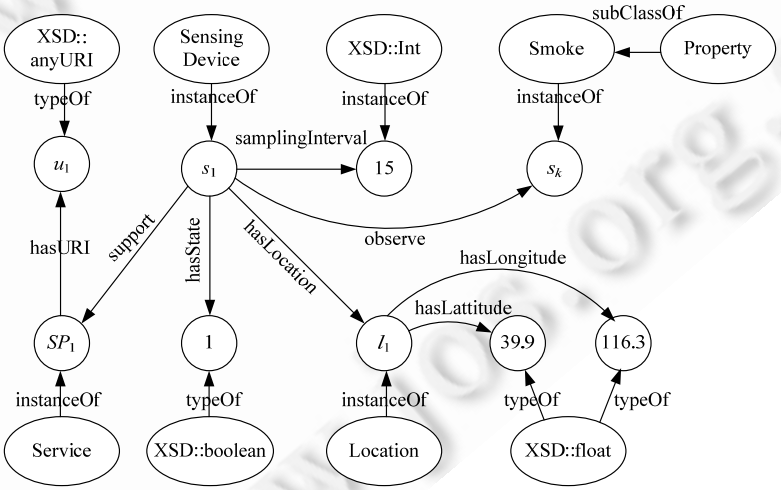

Fig.1 Smoke sensing service context ontology

图 1 烟雾传感服务上下文本体示例

对于关于服务的语义描述,我们采用基于 OWL-S 的本体来实现.OWL-S 包括了关于服务描述的 Service Profile、关于服务流程的 Service Process Model 以及关于服务具体连接参数的 Service Grounding.其中,Service Process Model 是主要的服务语义信息描述部件,它描述了服务的输入、输出、执行服务的前提和执行服务后的

结果,其中,

- 服务输入、输出描述了服务执行的数据接口;
- 而服务的执行前提和执行结果描述了服务执行所需满足的条件和执行后的最终结构.

执行前提和执行结果是判断服务是否满足应用需求的一个重要依据.因此,我们将重点讨论如何构建关于服务的执行前提和执行结果的语义描述.

为了有效地支持服务匹配、服务组合推理,我们引入动态描述逻辑 DDL(X@)[8]对服务的执行前提和执行结果进行统一描述.动态描述逻辑在描述逻辑基础上引入了动作的概念,并在逻辑系统中增加了一个动作定义集合 *ActBox*.其中,动作被分为原子动作和复合动作,原子动作为形如(*pre*,*occ*,*post*)的三元组,其中,

- *pre* 表示动作执行前提集合;
- *occ* 表示动作执行时可能改变状态的断言集合;
- *post* 表示动作执行后的状态变化集合.

以救援为例,假设我们定义一个 *secure* 表示救援动作,那么 *secure* 可以定义为

$$Secure(x?,y?)\equiv(pre_s,occ_s,post_s),$$

其中,

- pre_s={*Person*(*x*?),*Room*(*y*?),*Burning*(*y*?),*Stayin*(*x*?,*y*?)};
- occ_s={*Collapse*(*y*?)};
- $post_s$={*Saved*(*x*?)/¬*Stayin*(*x*?,*y*?),*Saved*(*x*?)/*InAmbulance*(*x*?)}.

在救援动作 *Secure* 中:

- 动作执行前提 pre_s 描述了 *secure* 必须满足有人在某间燃烧的房屋内才具备动作执行的前提条件;
- occ_s 描述了在救援动作执行时,房屋有可能会垮塌;
- $post_s$ 描述了如果救援动作执行,那么被救的人将不在燃烧的屋子里,同时被救的人应该在救护车上.

除了原子动作外,还可以与顺序连接复合“;”、选择连接复合“∪”、循环连接复合“*”、测试连接符号“?”等动作连接符号相结合,构成复合动作.例如,复合动作(?*BigFire*;*PutOut*)∪(?¬*BigFire*;*Secure*)表示如果火势过大(对应动作公式“?*BigFire*”),那么执行灭火行动(对应动作公式“*PutOut*”);如果火势不大(对应动作公式“?¬*BigFire*;*Secure*”),那么执行救援行动(对应动作公式“*Secure*”).为了简化动作描述,我们在复合动作定义中省略了形如“*x*?”的变量.

基于动态描述逻辑,可以将服务中的每一个执行前提和对应的执行结果描述为一个动作.其中,动作前提对应于服务的前提,动作执行结果对应于服务的结果.动态描述逻辑不但可以描述单个物联网服务,还可以对一些服务组合进行刻画.假设烟雾传感器 S_1 附近布置了若干温度传感器(统一记为 *Temp*),那么动作:

$$Trigger(S_1);(?Smoke;Tirgger(Temp);(?HighTemperature;Alarm))$$

描述了关于烟雾传感服务和温度传感服务如何彼此协同进行火灾监控的动态知识.根据该动态知识,系统首先触发烟雾传感服务,确认是否探测到烟雾,如果探测到烟雾,则触发温度传感服务 *Temp*.如果温度传感服务感知到温度过高,那么系统将触发火灾警报服务.

上下文本体和服务本体构成了物联网服务资源的语义模型.对于任意一个以 URI 标示的服务资源,以 GET,PUT,POST,DELETE 为基础的服务资源操作被描述为动态描述逻辑中的动作.实现动作的具体操作细节由 WSDL 来描述,并作为 OWL-S Service Grounding 中的内容.

2 基于物联网的多智能体决策系统

2.1 多智能体决策系统简介

在多智能体系统完成一个抽象复杂任务的时候,需要大量物理世界的实时信息作为决策依据.而物联网可以被视为一个具备提供海量物理世界信息能力的数据库,因此,我们提出一个基于物联网的多智能体决策支持系统——Multi-Agent Decision Support System,简称 MADSS.MADSS 的系统框架如图 2 所示.MADSS 通过任务

接口获得任务并传递给多智能体系统;多智能体系统根据知识库中的任务知识将待完成的抽象复杂任务分解为一系列耦合度较低的子任务集合;然后,将每个子任务安排给具体的智能体.

智能体在完成任务的时候需要感知物理世界的实时状态,并据此进行合理的决策.然而,由于智能体系统通常以高层语义形式(如描述逻辑)描述需要感知的环境状态,而对于物联网中海量的异构物联网实体(如网关、感知节点),一般仅能通过面向特定查询方式输出感知信息,因此,MADSS 根据物联网资源的语义模型,构建一个面向物联网资源的语义覆盖网.该语义覆盖网通过基于本体的资源描述方法实现对物联网资源的个体属性和访问接口的统一描述,使多智能体系统可以通过分析语义描述获取相应的物联网资源.通过物联网语义覆盖网,多智能体系统可以将复杂任务分解为一系列简单任务,并从物联网中获取完成简单任务所需的实时感知信息.

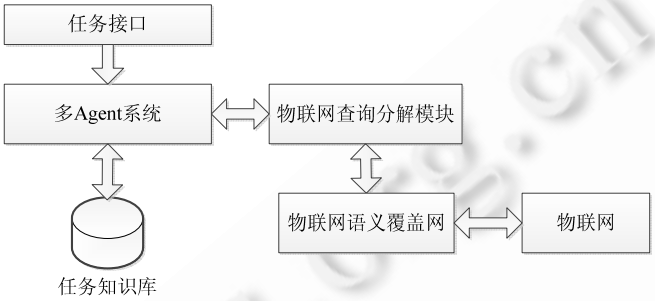

Fig.2 System structure

图 2 系统结构图

综上所述,能否从物联网中获取任务执行所需信息资源,决定了任务是否能被成功执行.因此,当进行任务分解时,多智能体系统需要确定物联网中的资源能否满足任务执行的信息需求.下面我们将具体讨论多智能体系统是如何根据物联网中的可用信息资源进行任务分解的.首先,需要对面向物联网的多智能体任务分解决策问题进行形式化描述.

定义 1. 一个任务 T 为一个形如 $T=\langle Init,Goal\rangle$的二元组.

Init 和 *Goal* 都是一个状态描述集合,*Init* 和 *Goal* 分别由一组描述逻辑公式所构成.其中,*Init* 表示任务执行前的系统状态;而 *Goal* 是任务执行后希望达到的系统状态,即目标状态.

定义 2. 一个任务 T 的分解方案是一个形如 $Schema(T)=\langle TaskSet,State,Action,QuerySet\rangle$的四元组.

$TaskSet=\{T_1,T_2,\ldots,T_n\}$为多智能体系统根据知识库分解得到的子任务的集合,$T_k$ 是第 k 个子任务;$State=\{state_1,state_2,\ldots,state_n\}$为完成抽象复杂任务所需要的环境信息,其中,$state_k$ 为完成子任务 T_k 所需要的环境信息;$Action=\{A_1,A_2,\ldots,A_n\}$为智能体完成任务所需要进行的行为决策,其中,$A_k=Decision(T_k,state_k)$是智能体为完成子任务 T_k 所采取的动作决策;$QSet_k=\{Q_1,Q_2,\ldots,Q_m\}$,$QSet_k\in QuerySet$,为要完成子任务 T_k 所需要的环境信息查询集合.显然,$QSet_k$ 由 $state_k$ 所决定.

例如,假设多智能体系统收到抽象复杂任务“对建筑 B 救火”.通过分解,可以得到多个子任务.其中,子任务 T_i 为“消防车到达火灾现场”.要完成该子任务,需要的环境状态感知为 *state*=“火灾现场到消防中心的交通状况”.虽然构建了语义覆盖网络,可以实现基于语义的感知信息查询,但是由于感知查询颗粒度较大,因此物联网无法直接理解并完成这种类型的环境状态查询,所以需要进一步对该抽象状态查询进行分解,最终生成能够被机器直接理解并执行的语义查询操作,$QuerySet=\{Q_1$=“北京一路的道路压力传感器状态”,Q_2=“北京二路的道路压力传感器状态”,Q_3=“北京一路的红绿灯状态”,Q_4=“北京二路的红绿灯状态”,…}.通过这些具体的查询语句,多智能体系统就可以从物联网中获取所需要的环境信息.

此外还可能存在一种情况,就是物联网无法提供执行任务所需要的信息.在此情况下,多智能体系统将尝试构建新的任务分解方案,使得物联网可以满足分解后的子任务的信息需求.

2.2 动态抽象任务分解机制

本文提出一种基于 TOP 的复杂任务的分解方法,该方法在 TOP 知识库的支持下,使用基于 TOP 的任务分解方法实现复杂任务分解机制 *Decompose*(*Task*,*KnowledgeLib*),得到可被智能体完成的任务分解方案——*Schema*(*Task*).

2.2.1 TOP 模型

Team-Oriented Plan(TOP)是一个面向大规模、异构智能体的分布式任务规划、分配及执行模型[9],TOP 通过构建任务协同组织结构.Agent 可以根据自身的能力和状态,在 TOP 任务分配框架下选择适合执行的原子任务.实践表明,TOP 为分布式的任务计算提供了一个合理、高效的计算模型[9].

本文中,为了能够充分地理解复杂任务的隐含语义并将其进行准确的分解,MADSS 需要建立一个完备的 TOP 知识库.知识库由表示背景知识的本体和基于动态描述逻辑的任务分解图集合所组成,其中,表示背景知识的本体除了包含关于概念的定义集合 *TBox* 和关于实例断言的集合 *ABox* 外,还包括了关于动作定义的集合 *ActBox*,*ActBox* 可以有效地支持基于动作的动态知识的表示和推理.同时,我们假设在动作执行前提成立的情况下,多智能体系统中的任务执行智能体均有执行 *ActBox* 中所定义动作的能力.另外,知识库中的每个任务分解图对应一个实际任务分解方案.任务分解图的最低层节点映射到一个具体的可执行子任务.TOP 的知识库可以由专家直接给出,也可以通过智能规划方法自动创建.

TOP 任务分解图是一个树形结构,其中,分解图中的每个节点均表示一个任务项,该节点的子节点表示对该节点的任务进行分解后得到的更加详细的子任务.在 TOP 任务分解图中,从父节点到子节点的分解表示了任务的细化和具体化.因此,TOP 树状图能够描述从根节点所对应的抽象任务逐层分解并最终细化为具体可执行的原子任务的过程.在 TOP 中,可分解的节点的所有子节点相互间通常具有某种执行上的逻辑关系.按照这种执行逻辑完成子任务,即可实现父任务.

基于上述分析,我们设计了一个任务分解图结构 $DGraph=\langle V,E\rangle$,其具体定义如下:

定义 3. 任务分解图 $DGraph=\langle V,E\rangle$,其中,$V$ 为节点集合 $V=\{node_i|node_i=(T_i,Goal_i)\}$,边集合 $E=\{rely\text{-}on\}\cup E_r$.

$Node_i$ 中的 T_i 为 $Node_i$ 所代表的任务标示,$Goal_i$ 表示执行 $Node_i$ 后希望达到的目标状态,$Goal_i$ 由一组形如 $C(x?),R(x?,y?)$的公式组成.边集合包括了联接父任务和子任务的 *rely-on* 关系,此外还有一个表示同级分解任务执行逻辑关系的集合 E_r.这种执行逻辑包括了任务之间的并行执行关系“∩”以及任务之间的串行完成关系“;”和任意执行关系“∪”.

为了便于查询任务分解树中的知识,我们定义了一个面向任务分解图结构的查询函数集合:

(1) *Root*(*graph*):返回分解图 *graph* 的根节点;

(2) *Child*(*Node*):返回 *Node* 的所有子节点集合;

(3) *Parent*(*Node*):返回 *Node* 的父节点;

(4) $Relation(N_i,N_j)$:返回 *Tree* 中 N_i 节点和 N_j 节点间的关系,当 $N_i\in Brother(N_j)$时,$Relation(N_i,N_j)\in\{\cup,\cap,;\}$;否则,$Relation(N_i,N_j)=NULL$;

(5) $Accomp(N_i)$:返回按执行逻辑关系连接的 N_i 子节点的序列.该函数可以回答关于如何完成 N_i 任务的问题,即:按照返回的执行逻辑关系完成 N_i 的子节点任务,即可完成 N_i 任务.如果 N_i 没有子节点,那么函数直接返回 N_i;

(6) *Leaf*(*graph*):返回分解图 *graph* 的叶节点集合;

(7) *LeafNodeExSeq*(*graph*):返回按照执行关系连接的 *graph* 的序列.

图 3 为救火领域相关的一个任务分解图.通过该图,我们可以看到一些任务之间的执行逻辑关系.比如:

- 救援任务执行前,必须先完成对被困人员的定位任务;
- 灭火任务可以和救援任务同时进行;
- 救护车和消防车可以同时调派;
- 完成灭火任务时,需要先对起火点进行定位,之后可以同时采用灭火器和喷水枪完成灭火任务.

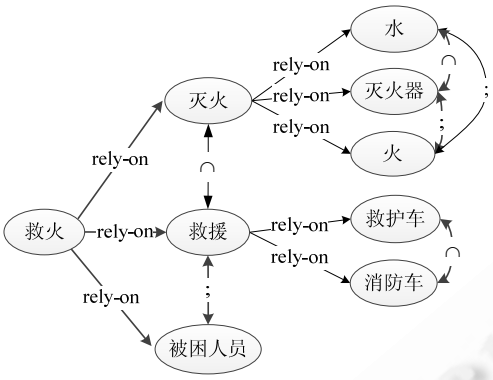

Fig.3 Fire-Fighting task decomposition

图 3 灭火任务分解示例图

2.2.2 基于 TOP 的分解方法

在 MADSS 中,基于 TOP 的复杂任务分解由函数 *Decompose*(*Task*,*KnowledgeLib*)完成.其中,*KnowledgeLib* 为 TOP 任务分解图知识库,该知识库包含了大量的 TOP 任务分解图.MADSS 根据分解图中蕴含的任务分解知识对 *Task* 进行分解,并且最终生成任务分解方案 *Schema*(*Task*).需要指出的是:由于 *KnowledgeLib* 具备丰富的任务分解知识,所以在进行任务分解时,我们忽略了 Agent Proxies 通过协商建立任务分解方案的过程,而是直接从 *KnowledgeLib* 中获取分解知识.也就是说:在 MADSS 中,Agent Proxies 仅具有任务的执行监测者和任务执行时协同者这两个角色.本小节将详细描述 *Decompose*(*Task*,*KnowledgeLib*)过程.

整个任务分解过程的基本思想介绍如下.任务分解过程主要包括 3 部分:第 1 部分发现可解决任务的分解图;第 2 部分基于 *ActBox* 中动作的推理确定实际可行的任务执行方案,并确定每个具体任务需要的决策信息;第 3 部分确定物联网是否能够满足具体任务所需要的决策信息,如果能满足,那么系统认为当前任务分解方案是一个可行的分解方案.

算法 1. *Decompose*(*Task*,*KnowledgeLib*).

输入:待分解任务 $Task=\{S_{init},S_{goal}\}$,包括了本体知识和分解图集合的知识库 *KnowledgeLib*;

输出:分解方案集合 *SchemaSet*.

(1) **For each** *DGraph* $g_i \in KnowledgeLib$ **do**

(2) **Begin For**

(3) $r:=Root(g_i)$;

(4) **If** $(\neg(Conj(getGoal(r))\rightarrow Conj(S_{goal}))$不可满足)

(5) **Begin If**

(6) Add g_i to *PSet*;

(7) **End If**

(8) **End For**

(9) **For each** $g_k \in PSet$ **do**

(10) **Begin For**

(11) $schema:=ProduceSchema(Task,g_k,ActBox)$;

(12) Add *schema* to *SchemaSet*;

(13) **End for**

(14) **For each** $s \in SchemaSet$ **do**

(15) **Begin for**

(16) **If** (!*Satisfy*(*s*))
(17) **Begin If**
(18) Remove *s* from *SchemaSet*;
(19) **End If**
(20) **End for**
(21) Return *SchemaSet*

函数 *ProduceSchema*(*Task*,g_k,*ActBox*).

输入:任务 *Task*、任务分解图 g_k、动作集合 *ActBox*;

输出:根据分解图生成的任务分解方案(*ActionQueue*,*State*).

(1) g_k:=*Rebuild*(g_k);
(2) *exSeq*:=*LeafNodeExSeq*(g_k);
(3) *i*:=0
(4) *S*=*getInit*(*Task*);
(5) **while** ((*n*:=*first*(*exSeq*))!=*null*)
(6) **Begin While**
(7) **If** ((Find action $\alpha \in ActBox$) and (($S \wedge Pre(\alpha)$)可满足) and
($\neg(Conj(Eff(\alpha)) \rightarrow Conj(getGoal(n)))$不可满足))
(8) **Begin If**
(9) Add α to action queue *AQue*;
(10) **For** $f \in Pre(\alpha)$ **do**
(11) **Begin For**
(12) **If** ($\neg(Conj(S) \rightarrow Conj(f))$不可满足)
(13) **Begin If**
(14) Continue;
(15) **Else**
(16) Add *f* to *State*(*i*);
(17) **End If**
(18) **End For**
(19) *i*:=*i*+1;
(20) *S*:=*update*(*S*,*Eff*(α));
(21) **Else**
(22) Return null;
(23) **End If**
(24) **End While**
(25) Return (*AQue*,*State*)

在算法 1 中,第 1 行~第 8 行在 *KnowledgeLib* 中搜索可以实现任务 *Task* 的任务分解图.

其中,通过$\neg(Conj(getGoal(r)) \rightarrow Conj(S_{goal})$不可满足性判定可验证分解图要分解的任务,即分解图的根节点,是否在语义上包含了 *Task* 的目标状态 S_{goal}.我们可以通过动态描述逻辑的 Tableau 推理算法[6]得到对于公式$\neg(Conj(getGoal(r)) \rightarrow Conj(S_{goal})$的不可满足判定.为了简化问题讨论,我们在算法 1 中省略了对于 *r* 中变量进行实例指派的讨论步骤.

算法 1 第 9 行~第 13 行对实现 *Task* 目标状态 S_{goal} 的分解图进行了基于动作的实例化,使分解图转化为实际可执行的动作决策序列.其中,*ProduceSchema* 函数实现了分解图的动作实例化.

在 *ProduceSchema* 函数中,首先将分解图 g_k 通过函数 *Rebuild* 进行优化并输出 *PrioGraph*.这是因为:

- 当完成某项抽象任务时,系统可以利用的各种资源是有限的;
- 同时,任务具有时效性,通常要求机器能在最短时间内最有效地利用资源取得最高的总效用.

在 *DGraph* 中,不同的 N_i 和 N_j 之间具有关联关系 $Relation(N_i,N_j)$,并且 $Relation(N_i,N_j)\in\{\cup,\cap,;\}$.由 $A\cup B$ 表示 $Accomp(A)$或者 $Accomp(B)$中任意一个实现即可,$A\cap B$ 表示 $Accomp(A)$和 $Accomp(B)$必须同时实现,可知连接符∩对资源的利用优先级应高于连接符∪.即:我们对资源进行分配时,需要首先完成 $Relation(N_i,N_j)=\cap$的节点 N_i 和 N_j 所表征的任务,接着再完成 $Relation(N_i,N_j)=\cup$的节点 N_i 和 N_j 所表征的任务.因此,系统首先通过 *Rebuild* 函数重新构造 *Tree*,使系统总是先考虑∩所表示的节点对应的任务.

通过分解图优化,系统获得了一个优化的任务分解方案.之后,通过 $LeafNodeExSeq(g_k)$函数可以获得一个分解图叶节点的任务执行序列.基于该序列,我们可以知道分解图中需要具体通过动作实现的任务节点(即分解图的叶节点)是哪些,并且它们之间的执行逻辑关系是怎么样的.针对每个叶节点,我们从 *ActBox* 中选择可以实现该叶节点的动作.为了使算法简明、易读,我们对 *ProduceSchema* 进行了一定简化:首先,在 *ProduceSchema* 中仅考察了对于一个动作是否可实现任务的情况,如果对 *ProduceSchema* 进行进一步扩展,引入自动规划推理机制,那么 *ProduceSchema* 即可实现对于多个动作的组合是否能现实任务的考察过程.此外,我们在 *ProduceSchema* 中也仅尝试搜索一种可能的对于 g_k 的动作实例化方案,如果将 *ProduceSchema* 函数改变为回溯结构,那么系统可以发现针对分解图 g_k 的所有可能的动作实例化方案.

ProduceSchema 函数按照任务执行序列的先后顺序逐一对任务节点进行动作实例化,在对动作实例化时,还需考虑动作执行的前提状态.对于任务序列的第 1 个节点,这个前提状态即为 *Task* 的初始状态,即 $getInit(Task)$的返回值 S_{init}.对于任务序列的后续节点来说,动作执行的前提状态为前一个任务的前提状态加上前一个任务执行后对于前提状态的改变.

ProduceSchema 函数第 7 行的 If 判断条件的含义为:尝试在 *ActBox* 中寻找一个动作α,该动作和当前执行前提状态 S 不矛盾;同时,该动作的执行结果可以满足当前任务节点 n 的目标.对于$(S\wedge Pre(\alpha))$的可满足性判断,即是确认动作的执行前提是否和当前状态 S 矛盾;而$\neg(Conj(Eff(\alpha))\rightarrow Conj(getGoal(n))$的不可满足性验证,即是确认动作$\alpha$的执行是否可以实现任务节点 n 的目标.

ProduceSchema 函数的第 11 行~第 18 行尝试将动作α的执行前提条件集合分离为两个子集合,其中一个子集合为动作前提状态所蕴含的前提条件,另外一个子集合为前提状态不蕴含的前提条件集合.对于第 2 个子集合,需要通过查询物联网来确定该动作是否具备可执行条件.因此,我们将第 2 个子集合中的前提条件作为需要从物联网中进行考察的情况,并用 *State* 标记.最后,第 20 行根据动作的执行效果和动作执行前的前提状态对下一个动作的前提状态进行更新.更新方法是将 $f\in S$ 且 f 与 $Eff(\alpha)$矛盾的公式从 S 中删除,并将 S 中剩余公式和 $Eff(\alpha)$中的公式合并,形成新的动作执行前提.整个更新过程由 $update(S,Eff(\alpha))$函数完成.

在完成对于任务分解图的动作实例化后,算法 1 对于每个实例化的动作图进行考察,确定物联网是否能够提供分解图中动作执行时需要的信息.考察过程由函数 $Satisfy(s)$确定,如果物联网能提供 s 执行所需信息,那么 $Satisfy(s)$返回 true;否则返回 false.算法 1 第 16 行中的"$!Satisfy(s)$"表示对函数 $Satisfy(s)$的值取反.关于 $Satisfy(s)$函数的具体实现细节,我们将在下一小节中加以具体阐述.

我们通过一个例子进一步说明 MADSS 的任务分解过程:假设宾馆 H 突发火灾,需要对其进行救火,MADSS 通过任务接口接收到"对宾馆 H 救火"的复杂任务后,将其映射到 TOP 知识库中的"救火"任务分解图 G,利用 G 对救火任务进行分解,得到如图 4 所示的任务分解的拓扑图.

在知识库的支持下,MADSS 利用 TOP 方法将复杂任务"宾馆 H 救火"分解得到若干子任务,T_1="营救被困人员",T_2="派出消防车",T_3="派出救护车",T_4="找到火",T_5="找到灭火器",T_6="找到水".要完成这些子任务,需要了解相应的物理世界信息.根据 *ProduceSchema* 过程,得到对应的抽象查询:$state_1$="被困人员名单",$state_2$="消防车救援路线",$state_3$="救护车救援路线",$state_4$="着火位置",$state_5$="灭火器位置",$state_6$="水龙头位置".

由此可以看出:MADSS 利用知识库中任务分解图知识,将复杂任务 *Task* 快速分解,并得到 *TaskSet* 集合.

TaskSet 中的每项具体任务 T_k 由一个具备领域知识的智能体负责完成,该智能体将执行任务时所需物理信息通过物联网查询分解模块分解为一系列针对物联网感知信息的查询,随后再通过物联网语义覆盖网将信息查询映射到具体的物联网资源查询.

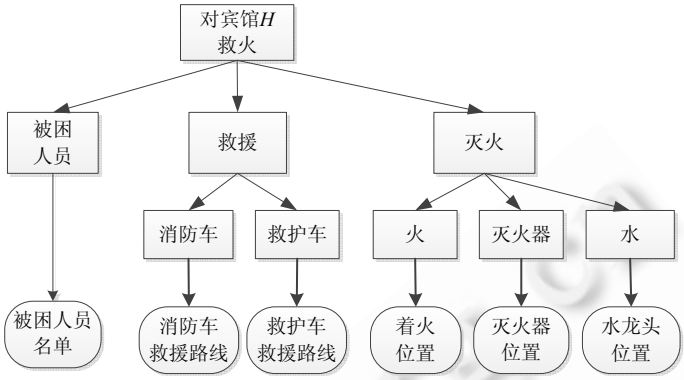

Fig.4 Task decomposition based TOP

图 4 TOP 任务分解拓扑图示例

2.3 基于TOP和本体的物联网抽象查询分解

如前所述,动作决策依赖于实时感知环境信息 *state*.然而,*state* 通常是比较抽象的环境状态,使得机器难以直接将 *state* 查询映射到可直接执行的物联网原子查询语句.因此,MADSS 需要一个将 *state* 抽象查询映射到具体原子查询集合 *QuerySet* 的查询分解过程.根据 *QuerySet*,MADSS 可以确定物联网是否能够对动作执行提供必要的信息支持.分解 *state* 查询的过程是一个抽象度逐层降低的语义分解过程,可以通过 TOP 知识库中的任务分解知识实现语义的逐层分解.然而,TOP 知识库通常是面向特定任务的,而 *state* 通常表示的是更为宽泛的物理世界状态,因此,很难对每个 *state* 都找到一个 TOP 分解图与之对应.所以,我们尝试构建一种基于本体的 *state* 查询分解方法.

由算法 1 可知,需要从物联网查询的信息是动作执行前提条件集合的子集,而由动态描述逻辑表示的动作执行前提条件包括了两种形式的公式:形如 $C(x?)$的概念断言公式和形如 $R(x?,y?)$的关系断言公式.它们的含义分别是:如果个体 $x?$是 C 的实例,那么该前提条件成立;如果个体 $x?$和个体 $y?$存在 R 关系,那么此前提条件成立.这两种类型的前提条件都可以视为关于物理世界个体的状态验证,如 *HighTemperature*($r?$)表示验证房间 $r?$温度是否过高,*InSameRoom*($x?,y?$)表示验证 $x?$是否和 $y?$在同一个房间等.

为了简化讨论,我们假设物联网仅提供关于物理世界各种物理对象的属性的感知信息,如物理对象的位置、运行速度、运行方向等.同时,此类属性信息均可通过与物理对象关联的传感器获得.因此,对于一个物理对象的状态验证问题,即可转换为关于与物理对象关联的传感器的感知信息的获取问题.进一步地,由于物理对象关联传感器信息隐含了传感器的信息获取需要具备时空约束(temporal-spatial constraints)这一信息,因此,关于物理对象的状态验证可以被映射为一个关于某时间段内,某空间领域、某些类型感知信息的获取行为.综上分析,可以将关于动作执行前提的物理对象状态验证转化为具有时空约束的感知信息分解查询,并最终形成一系列关于物联网中传感器的查询请求.

通过本体,我们可以表示针对某个物理对象状态验证需要的各种感知信息,如对于一个房间是否知道起火的状态验证可表示为关于 *RoomOnFire* 的概念定义式:

$$RoomOnFire \equiv Room \sqcap \exists hasSensingDevice.(\exists hasLocation.(InRoom \sqcup NearRoom) \sqcap \exists observes.(Property \sqcap Smoke) \sqcup \exists observes.(Property \sqcap HighTemerature))).$$

此定义式的具体含义为:如果位于一个房间内或者在房间附近的传感器感知到烟雾或者温度升高,那么该

房间发生火情.

根据此定义式,想要验证 *RoomOnFire*(No.116)的状态,即可通过获取 No.116 房间内或者房间附近的传感器的关于烟雾和温度的实时感知信息而确定.

可以注意到:描述物理实体状态的概念定义式只针对与状态概念本身相关的感知信息的逻辑关系进行描述,缺乏时间、空间的约束.因此在实际的查询分解中,需要将概念定义式与时间、空间概念联合,形成具有时空约束的查询.如下述时空约束:

$$RoomFireState \equiv RoomOnFire \sqcap \exists hasTime.(\text{“2012-7-8”}) \sqcap \exists hasLocation.(\text{“168}Express\ Hotel\text{”}).$$

该时空约束下概念对应为一个针对 2012 年 7 月 8 日,在 168 Express Hotel,是否有房间起火的信息查询.加入时空约束后,对物理对象“No.116”进行关于 *RoomFireState*(No.116)的状态验证.当“No.116”位于宾馆“168 Express Hotel”,时间为“2012-7-8”,且状态 *RoomOnFire*(No.116)为真时,*RoomFireState*(No.116)返回真值.

RoomOnFire(No.116)是关于概念断言的状态验证.而对于关系断言的状态验证,本体仅能表示关系之间的包含语义,如 *ArtificialBreathing*⊏*Rescue* 表示人工呼吸是抢救的子关系,即:

$$ArtificialBreathing(A,B) \models Rescure(A,B).$$

相对于概念定义式,关系之间的包含关系定义为查询分解提供了较少的信息.为了丰富面向关系验证的背景知识,我们在查询分解知识库中引入了部分基于 SWRL 构建的规则,以对面向关系验证的查询分解形成有效的支持,例如:

$$Hotel(x?) \wedge Hydrant(y?) \wedge Near(y?,x?) \wedge OnFire(x?) \Rightarrow Quench(y?,x?)$$

表示如果消防栓 *y*?距离宾馆 *x*?很近,那么消防栓 *y*?可用来扑灭 *x*?的火灾.

在查询分解时,可将规则的尾部中出现的关系进行新的基于规则的扩展,通过不断地进行规则迭代,直到规则尾部仅有概念断言验证为止;再通过面向概念断言的查询分解,即可实现关系验证的查询分解.当然,不规范的规则设计也可能导致规则陷入无限迭代中,因此我们设置了规则迭代的次数阈值,以保证查询分解算法能在有效的时间内停止.此外,同样地,对于关系状态的验证需要与时空约束概念相结合.

由上分析可以看出,面向概念断言的查询分解是整个物理对象状态验证的基础.因此,我们需要设计一种针对概念断言的查询分解方法.该查询分解方法的基本思想是:

- 针对一个概念断言 *C*(*x*?),首先在本体中查找 *C* 的概念定义式,如果未找到,则返回系统无法验证 *C*(*x*?);如果找到相应概念定义式,系统将概念定义式按照其概念的嵌套形式,分解为一个树形结构(如图 5 所示),并尝试从叶节点开始逐步与物联网语义覆盖网进行匹配,探查物联网信息库中是否有能满足该节点语义的实例对象;
- 随后,分解树的兄弟节点合并,生成新的概念节点,并再将新的概念节点与物联网语义覆盖网进行匹配,如此反复,直到匹配到树的根节点为止.如果匹配到树的根节点,且物联网中有对象属于该根节点对应的概念,再通过时空约束对对象作进一步过滤;
- 最后,如果存在满足时空约束的根节点对应概念实例,那么系统返回可以实现查询分解.

为了提高概念断言的分解效率,我们对概念定义式的形式进行了一定规范.该规范规定:概念定义式对应的分解树的根节点必须包含一个形如∃*hasSensingDevice*.()的子节点,且该子节点与其他兄弟节点为交的关系.即,概念定义式必须是概念和拥有某些传感器的集合的交.

进一步地,对以∃*hasSensingDevice*.()为根节点的子树解析,就可以获得相应的可以验证对象状态的传感器集合,再通过时空约束关系,即可确定相应感知查询集合.

这里需要说明的是:如果将整个关于物联网的语义覆盖网视为一个本体,那么可以直接通过最小概念包含、概念实例推理来确定是否有满足概念定义式的物联网对象.然而现实情况是,物联网中的语义描述信息一般是分布式的,同时,针对不同类型的对象有不同类型的本体,因此无法将整个语义覆盖网视为单个本体.所以,我们考虑通过将分解树匹配的形式,实现从状态验证到查询分解的转换.

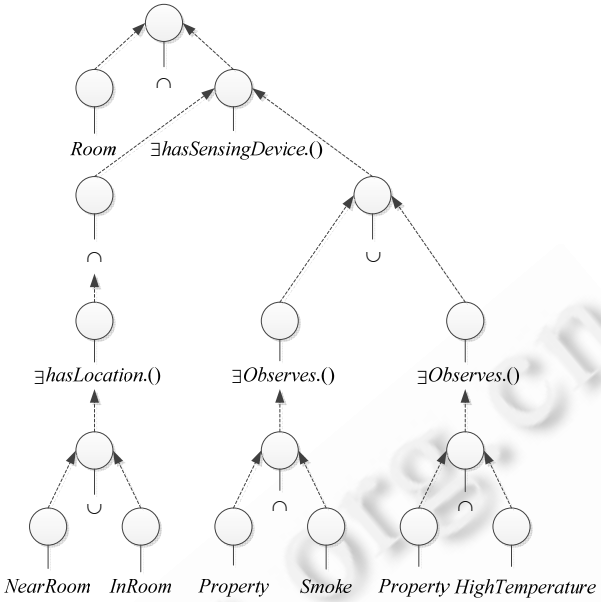

Fig.5 Hierarchical concept tree

图 5 概念分层树示例

算法 2 描述了从状态验证到查询分解的过程.

算法 2. 查询分解 *Satisfy*(*schema*).

输入:任务分解图;

输出:分解图能否被满足,分解的查询集合.

(1) *State*=*getStateSeq*(*schema*);
(2) **For** (*i*=1 to *i*=*State.length*) **do**
(3) **Begin For**
(4) **For** each *f*∈*State*(*i*) **do**
(5) **Begin For**
(6) S_{query}=*QDecompose*(*f*);
(7) **End For**
(8) **If** (*Squery*==*null*)
(9) **Begin If**
(10) Return false;
(11) **Else**
(12) Add S_{query} to *QuerySet*(*i*)
(13) **End If**
(14) **End For**
(15) **Return** true;

算法 2 首选通过 *getStateSeq*()函数从 *schema* 分解图中获取动作前提集合序列.第 4 行~第 7 行对动作前提集合中的物理对象状态逐一进行考察.其中,第 6 行的 *QDecompose*(*f*)函数根据状态验证公式 *f*,生成相应的传感信息查询集合.

函数 *QDecompose*(*f*).

输入:条件公式 *f*;

输出:查询集合 *QuerySet*.

(1) *QuerySet*:={}
(2) **If** (*f* is a relationship assertion)
(3) **Begin If**
(4) *validationSet=Decompose*(*f*);
(5) **Else**
(6) Add *f* to *validationSet*;
(7) **End If**
(8) **For each** (*C*∈*validationSet*) **do**
(9) **Begin For**
(10) **If** (*C* is a relationship assertion)
(11) **Begin If**
(12) **If** (*C* can be mapped into a query set *qRSet*(*C*))
(13) **Begin If**
(14) Add *qRSet*(*C*) to *QuerySet*;
(15) **Else**
(16) Return *null*;
(17) **End If**
(18) **Else**
(19) *DTree=BuildTree*(*C*);
(20) **While** ((*nodeSeq=getLeaf*(*DTree*))!=*null*)
(21) **Begin While**
(22) **For each** (*n*∈*nodeSeq*) **do**
(23) **Begin For**
(24) *NC=getNodeConcept*(*n*);
(25) Build individual set *ISet*(*NC*)={*i*|*NC*(*i*)==true};
(26) **End For**
(27) *DTree=ReBuildTree*(*DTree*);
(28) **End While**
(29) *NC=getNodeConcept*(*getRoot*(*DTree*));
(30) Build *ISet*(*NC*)={*i*|*NC*(*i*)==true};
(31) Filter *ISet*(*NC*) with temporal-spatial constraints concepts;
(32) **If** (*ISet*(*NC*)!=*null*)
(33) **Begin If**
(34) *QuerySet=QuerySet*∪*BuildQuerySet*(*C*,*ISet*(*NC*));
(35) **Else**
(36) **Return** *null*;
(37) **End If**
(38) **End If**
(39) **End For**
(40) **Return** *QuerySet*;

QDecompose 函数首先判断公式 *f* 是否为关系验证公式,如果是,那么通过规则迭代,将关系验证公式转换为更加具体的状态验证公式集合 *validationSet*.第 8 行~第 36 行根据 *validationSet* 中的每一个公式,生成相应的查询集合.第 12 行~第 17 行针对公式是关系公式的情况,如果关系公式可以直接映射为查询集合 *qRSet*,那么将 *qRSet* 添加到 *QuerySet* 中.第 19 行~第 34 行针对概念断言公式进行查询分解.第 19 行的 *BuildTree*()函数将概念断言 *C* 分解为对应的树结构.之后,通过 *getLeaf*()函数获取分解树的叶节点,并运用本体推理技术以及基于 *sparql* 的个体查询技术发现属于某个叶节点概念 *NC* 的实体集合 *ISet*(*NC*).根据概念定义式中的逻辑关系,将具有相同父节点的概念节点合并,并据此构建父节点概念.随后,通过 *ReBuildTree*()删除叶节点.重复上述过程,直到 *DTree* 中只剩下根节点为止.通过时空约束概念对根节点概念包含的实例进行过滤,如果最后 *ISet* 不为空,则说明物联网可以验证 *f* 的状态.

3 案例分析

为了验证本文提出的基于多智能体的物联网语义决策支持技术方案,我们通过已实现的一个保险事故处理指挥辅助决策系统作为案例来分析本设计的可用性和先进性.实现的系统整体框架如图 6 所示.

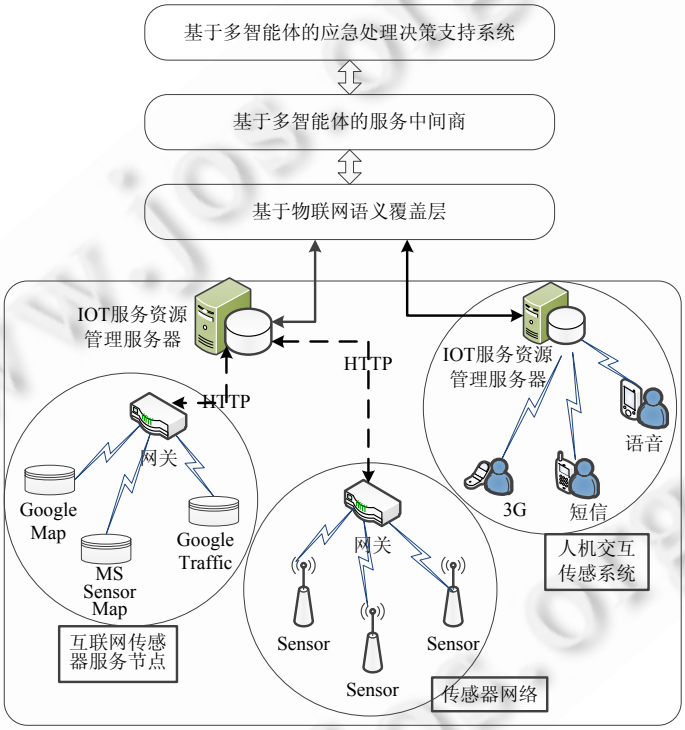

Fig.6 Framework of the decision support system for insurance accident handling

图 6 保险事故处理指挥辅助决策系统框架图

在该系统中,我们定义物联网中的传感节点可以通过 6LowPAN 等协议与网关连接.网关将与其连接的传感器节点的服务信息注册到 IOT 服务资源管理服务器.这一过程可以通过人工构建或者自动发现等方法来实现.由于目前缺乏相应的物理传感器支持,我们采用 Google Map 和 Microsoft Sensor Map 等虚拟传感器服务进行案例验证.同时,该方案将智能手机作为一类特殊的人机交互传感器,智能手机可以通过 3G、语音和短信等方式获取使用者的相关信息和状态,也可向指定执行者发布指令.在物理传感器网络上,我们定义了资源管理服务器,用于生成其关于 IOT 注册服务的语义元信息.系统中有多个 IOT 服务资源管理服务器,因此,服务资源和其对应

的语义元信息是分布式的.存储在各个资源管理服务器中的分布式语义元信息形成了物联网语义覆盖网.由多智能体构成的服务代理系统通过分布式的语义推理,实现物联网资源发现、服务代理等.在本系统中,关于查询的分解、匹配、资源定位均由服务代理系统完成.服务代理系统之上是多智能体应急事件处理系统,该系统通过服务代理系统获取 IOT 的实时信息,针对保险事故紧急事件执行相应的任务分解和执行.

针对具体的交通事故,本系统通过任务分解形成了如图 7 所示的具体执行方案.通过基于语义覆盖网的查询分解,任务中各种信息需求最终转化为一系列传感信息查询和管理信息查询.根据最终得到的任务查询系统和底层传感器网络获取信息,返回给用户作为决策支持.

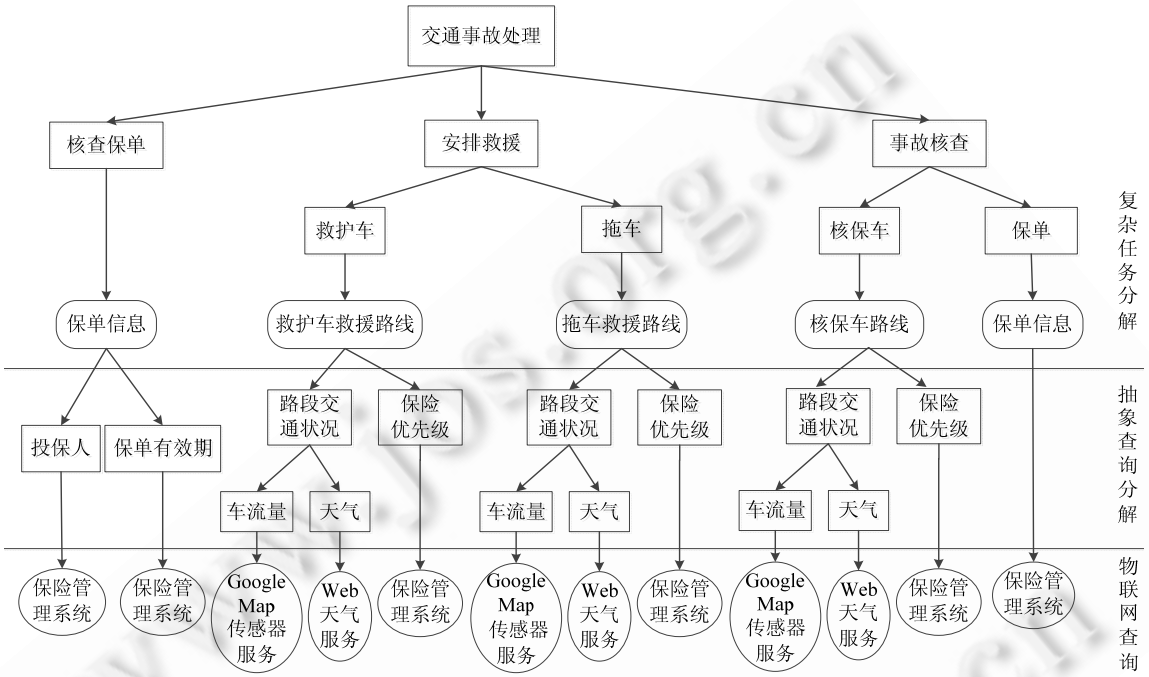

Fig.7 Traffic accident handling plan

图 7 交通事故处理方案

保险事故处理指挥辅助决策系统执行过程如图 8 所示,其基本流程为:

(1) 当有交通事故发生时,系统进行事故通报,多智能体系统对任务进行分解,首先对保单进行核查,将受理保单的详细信息发送给决策者;

(2) 调出事故发生地周边所有的核保车辆信息.系统将依据事故报告自动生成地图上的事故发生地点,同时依据车辆状态,在地图上标记核保车辆当前位置及其空闲状态,可自动设置所有核保车辆执行该任务的优先排序,以供决策者选择执行该次任务的核保车辆;

(3) 针对选中的核保车辆,多智能系统通过 Google Map 路况给出当前路况和多条路径规划,决策者根据路况选择执行任务路径;

(4) 当路径选择成功后,系统自动生成任务执行信息,通过系统发送短信到车辆随车人员的手机上,通知其执行任务;

(5) 多智能体系统通过底层传感器网络实时地获取执行任务的核保车辆信息,对核保车辆进行实时监控.如果核保车辆偏离执行任务路线,则进行异常报警反馈给决策者.帮助决策者实时追踪最新状态,保证任务合理、快捷、无错地完成,避免骗保等商业犯罪事件的发生.

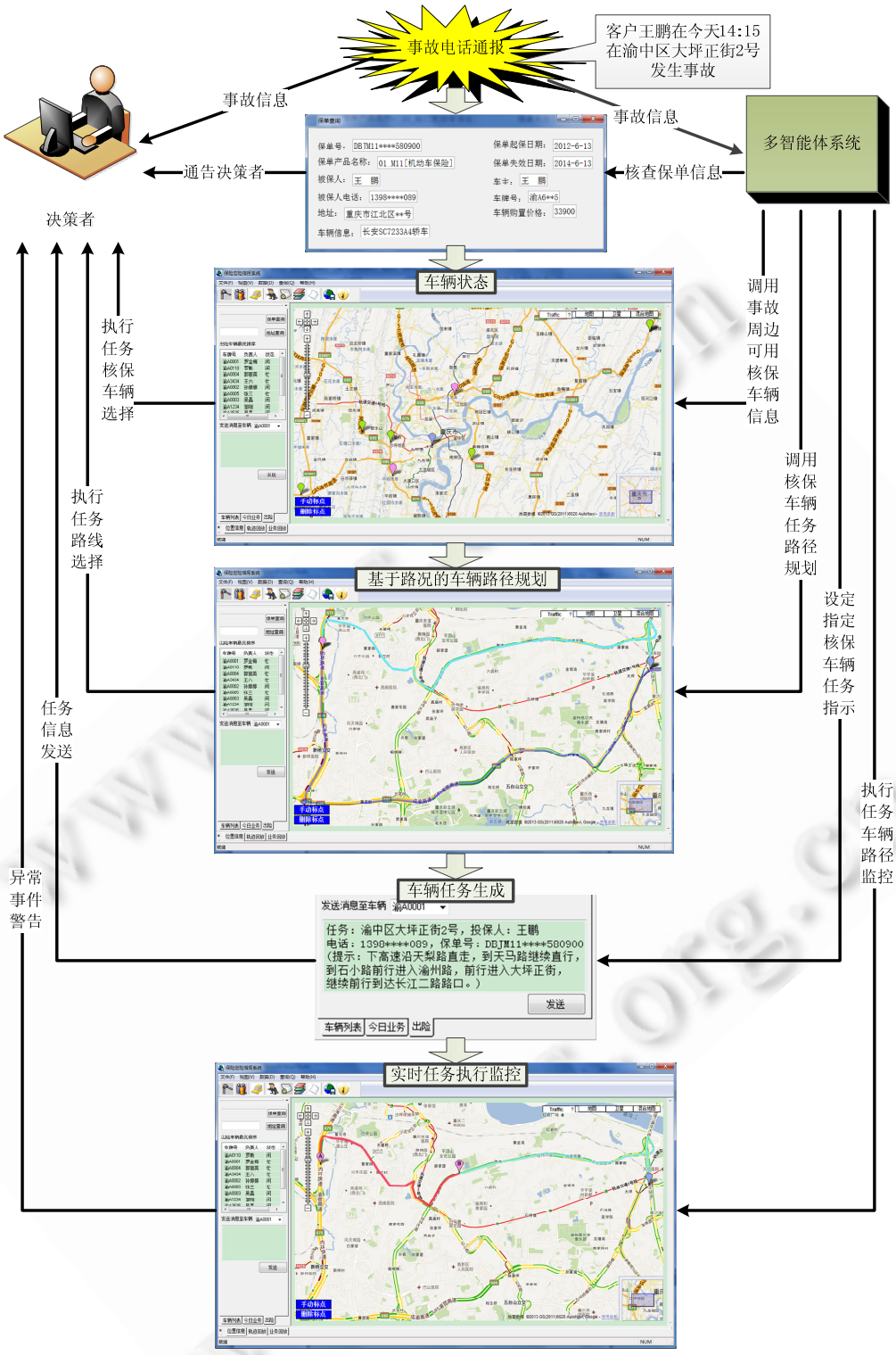

Fig.8 Illustration of insurance decision support process for an accident report

图 8 保险事故处理决策支持过程示例

在系统实现效果上,我们对本系统的事故处理方法与传统电话事故处理方法进行效率上的对比分析,包括两个方面:保险公司总部客服对客户电话报告事故处理和核保车辆的调度.在数据分析中,我们截取在节假日高峰出行阶段,即最易发生交通事故和拥堵的时间段内,该保险公司对普通事故的处理记录进行分析.

表 1 展示了客服对客户出险后处理时间的对比数据,从统计结果可以看出:基于多智能体的辅助决策系统通过智能化快速调用客户保险记录、核保车辆信息等材料,帮助客服极大地缩短了事故服务处理和服务等待时间,大大提升了用户体验效果.

Table 1 Efficiencies of different customer services to handle traffic accidents

表 1 客服对事故处理的效率对比

	一个小事故平均电话客服处理时间(/分)	每小时平均处理客服数量(/个)	客户平均等待时间(/分)	平均等待队列(/人)
传统电话事故处理方法	15	5	10	4
保险事故处理指挥辅助决策系统	5	10	几乎不等待	0

另一方面,在表 2 中,我们给出了核保车辆对普通事故处理效率的统计对比数据.从统计结果可以看出:本系统在核保车辆驾驶员端通过传感器对交通状态的分析,智能提示最快到达现场的方法,大大缩短了核保车辆到达事故处理现场的时间,并减少了由于在行进中拥堵所带来的燃油消耗,提升了核保车辆的使用效率.例如,核保车辆对事故处理的效率提升了 28.8%,同时,车辆月燃油消耗反而降低了 15.7%.

Table 2 Efficiencies of different driving route decision making

表 2 核保车辆对事故处理路径选择的效率对比

	一个事故平均到达时间(/分)	提升效率	核保车辆月燃油消耗减少
司机依据经验驾驶	45	–	–
依据多智能体决策支持系统推荐路线驾驶	32	28.8%	15.7%

4 相关研究

传感器设备接口的不统一和传感信息的异构性,是多智能体系统自主规划和决策支持在物联网应用的一个主要瓶颈[10].如何对传感器进行有效的描述,并利用传感器的描述信息进行感知信息的获取与应用,是近年来的一个热门研究课题.语义 Web 中的基于本体语义建模技术是描述传感器元信息的一种有效手段.目前,已有许多研究人员设计出了多种传感器本体模型.传感器本体主要用于描述特定传感器基本参数和相关功能.根据本体描述的传感器知识以及基于本体的自主推理,即可实现基于上层语义的抽象查询.基于本体的感知信息获取可以获得单个传感器的感知信息,然而无法实现针对多个传感器的组合推理[11,12].OGC 提出了 Sensor Web Enablement 架构[13],主要给出了 Observation and Measurement 和 Sensor Mode Language 的概念,其中,Observation and Measurement 从用户角度对传感器提供的服务进行了表示,这样能够较好地将由上层语义描述的用户请求与底层传感器的属性进行连接.然而,Observation and Measurement 只是实现了语法的交互性,没有提供逻辑推理能力.Jirka 提出了使用 Observation 来解决 Sensor Web Enablement 的服务发现问题的方案[14],侧重于通过包容推理来研究现象之间的分层关系.由于所提供的推理功能较为简单,因此,Jirka 的工作无法为复杂的物联网提供较好的信息推理服务.在 OOSTethys 中,研究者通过为 Observation 添加 has-procedure 属性,使 procedure,sensor device 和 system 获得更好的组合[15].W3C 研究小组提出的 CSIRO 架构实现组合虚拟传感器方式[16],其缺点在于当查询目标较为复杂时,查询次数会急剧增加,而文献[14]中未给出提高效率的查询策略.Sgroi 等人提出了通过定义一套服务标准和 API 来实现对底层传感器网络的细节封装[17],但是该工作没有对传感资源进行系统的定义.Global Sensor Network(GSN)是一个允许传感器扩展的整体框架[18],该框架使用 wrapper 对所有传感器进行简单抽象,并提供了统一的基于 Web 的上层查询框架,从而成功实现了对底层传感器的统一封装和上层查询.Dominique 提出了针对传感资源的服务封装与描述[19],然而该服务描述只是基于服务的语法结构描述,缺乏通用性和对智能推理的有效支持能力.

上述研究方法虽然实现了对传感器的有效语义表示,但都不适合直接应用于物联网.

- 首先,物联网是一个基于传感资源的服务网络[9,19−21],因此,针对物联网的语义建模必须是面向服务的语义建模,而不仅仅是面向传感器的语义建模;
- 进一步地,考虑到传感资源的可用性与其时空是特征紧密关联的,在进行语义建模时需要针对时空特征与服务的有效性进行统一建模.

据此,本文提出了针对物联网服务资源及其时空特性的语义建模方法.在该建模方法中:

- 我们首先通过 restful 服务对物联网中的感知资源进行服务封装,生成相应的物联网的资源服务;
- 进一步地,通过构建面向 restful 服务的语义模型,实现了对于物联网服务资源的语义描述;
- 最后,我们在语义模型中引入了时空描述方法,实现了对于物联网资源服务的时空特征的有效语义建模.

文献[9,20]中也提出了类似的研究思路,相比较而言,我们的工作更加深入.同时,我们在语义建模过程中着重考虑了资源的时空可用性.

通过理解用户查询中的关键词,实现从上至下的服务查询也是一种有效的传感信息查询手段.

Biranchini 提出了一个普遍的服务发现框架[22],在这个框架中,输入被格式化成 hasoutput 等的属性,并与标准服务原语相匹配,而后输出查询.但这种匹配方式不能实现对上层查询抽象术语的理解.

Silva 实现了一种从用户的视角来描述服务发现的方法[23],它将请求定义为目标,并将目标分解为任务,最后将任务分级为服务,然而系统无法自动将一个用自然语言表述的请求匹配到一个目标.

Ji 借助本体在允许模糊查询的条件下重定义和扩展概念特性来扩展用户自然语言输入的目的,并提高了精度[24],但其主要通过领域本体关注特定领域的服务,应用领域有限.

基于 TOP 的多智能体复杂任务分解方法可应用于物联网信息支持[25],它依据分解得到的原子任务进行决策信息支持.

Pynadath 等人首先提出面向团队的规划 TOP 方法对多智能体系统复杂任务进行分解[6],使用 TOP 将不同子任务涉及的异构多智能体系统整合并合作完成高层任务,并用于实现高效的任务分配和资源利用.Scerri 针对宾馆救火问题扩展 TOP 对复杂任务分解的思路并应用于高效的异构型多智能体团队任务规划[26].Yang[25]提出了类似于 TOP 的任务语义分解的架构进行 Web 信息服务支持,但其模型仅提出了简单的基于匹配的信息服务方法,无法直接扩展到多智能体系统的决策支持应用中.

与以往的基于 TOP 的工作不同,本文提出了面向物联网语义覆盖层的 TOP 任务规划方法.

该方法充分利用了面向物联网资源的语义描述元信息,通过基于动态描述逻辑的规划推理,实现了从高层语义描述的信息需求到底层传感器资源的动态映射.基于该方法,多智能体决策系统可以有效地利用各种现有的物联网感知信息资源,实现对于物理世界中各种紧急任务的有效任务分配、实时任务执行情况监测、动态任务调整以及任务重规划等.

5 总结与展望

物联网的动态信息服务支持,是基于物联网应用的多智能体系统实现复杂任务自动化的基础.

本文研究了物联网的语义覆盖网络模型,并提出基于语义覆盖网和 TOP 的抽象复杂任务的分解方法.

进一步地,本文利用 TOP 和本体知识库实现了针对子任务所需抽象环境状态查询的分解,并最终得到能够直接从物联网中具体的传感器中获取信息的查询语句.

通过对抽象任务的两次分解,本系统模型成功地从物联网中找到对应的传感器为多智能体系统提供了必要的决策信息支持.

目前,本文只考虑了物联网对多智能体系统的信息支持,即,返回相关信息给多智能体系统作决策支持,不对信息本身进行处理和整合.在后续的工作中,我们将考虑对信息一致性和冗余性进行处理整合,支持多智能体系统更加智能的行为决策.

References:

[1] Hirankitti V, Krohkaew J, Hogger C. A multi-agent approach for intelligent traffic-light control. In: Proc. of the 1st Asia Int'l Conf. on Modeling & Simulation. IEEE, 2007. 496−501. [doi: 10.1109/AMS.2007.11]

[2] Chaimowicz L, Kumar V. Aerial shepherds: Coordination among UAVs and swarms of robots. In: Proc. of the DARS. 2004. 243−252. [doi: 10.1007/978-4-431-35873-2_24]

[3] Xiao QF, Wang Y, Wang Y. Research on emergency disposal platform based on multi-agent with a cooperative model. In: Proc. of the Advanced Materials Research, Vol.712-715. 2013. 3106−3111. [doi: 10.4028/www.scientific.net/AMR.712-715.3106]

[4] Domnori E, Cabri G, Leonardi L. Multi-Agent approach for disaster management. In: Proc. of the Int'l Conf. on P2P, Parallel, Grid, Cloud and Internet Computing. 2011. 311−316. [doi: 10.1109/3PGCIC.2011.57]

[5] Bormann C, Castellani AP, Shelby Z. CoAP: An application protocol for billions of tiny Internet nodes. IEEE Internet Computing, 2012,16:62−67. [doi: 10.1109/MIC.2012.29]

[6] Pynadath VD, Tambe M, Chauvat N, Cavedon L. Toward team-oriented programming. Intelligent Agents VI: Agent Theories, Architectures, and Languages, 2000,1757:233−247. [doi: 10.1007/10719619_17]

[7] Compton M, Barnaghi P, Bermudez L, *et al*. The SSN ontology of the W3C semantic sensor network incubator group. Web Semantics: Science, Services and Agents on the World Wide Web, 2012,17:25−32. [doi: 10.1016/j.websem.2012.05.003]

[8] Chang L, Shi ZZ, Gu TL, Zhao LZ. A family of dynamic description logics for representing and reasoning about actions. Journal of Automated Reasoning, 2012,49(1):1−52. [doi: 10.1007/s10817-010-9210-1]

[9] Wang W, De S, Cassar G. Klaus moessner, knowledge representation in the Internet of things: Semantic modelling and its applications. Automatika-Journal for Control, Measurement, Electronics, Computing and Communications, 2013,54(4). [doi: 10.7305/automatika.54-4.414]

[10] Xie L, Chen LJ, Chen DX, Xie L. Query processing for wireless sensor networks. Computer Science, 2006,33(9):45−49, 68 (in Chinese with English abstract).

[11] Eid M, Liscano R, Saddik AE. A novel ontology for sensor networks data. In: Proc. of the IEEE Int'l Conf. on Computational Intelligence for Measurement Systems and Applications. 2006. 75−79. [doi: 10.1109/CIMSA.2006.250753]

[12] Eid M, Liscano R, Saddik AE. A universal ontology for sensor networks data. In: Proc. of the IEEE Int'l Conf. on Computational Intelligence for Measurement Systems and Applications. 2007. 59−62. [doi: 10.1109/CIMSA.2007.4362539]

[13] Botts M, Percivall G, Reed C, Davidson J. OGC sensor Web enablement: Overview and high level architecture. GeoSensor Networks Lecture Note in Computer Science, 2008,4540:175−190. [doi: 10.1007/978-3-540-79996-2_10]

[14] Jirka S, Broring A, Foerster T. Handling the semantic of sensor observables within SWE discovery solutions. In: Proc. of the Collaborative Technologies and Systems (CTS). 2010. 322−329. [doi: 10.1109/CTS.2010.5478495]

[15] Maskey M, Conover H, Keiser K, *et al*. OOSTethys/Oceans IE service registry based on catalog service for Web. In: Geospatial Web Services: Advances in Information Interoperability. Hershey: IGI Global, 2010. 97−117. [doi: 10.4018/978-1-60960-192-8.ch005]

[16] Li C, Dutta R, Kloppers C, *et al*. Mobile application based sustainable irrigation water usage decision support system: An intelligent sensor CLOUD approach. In: Proc. of the 2013 IEEE Conf. on Sensors. IEEE, 2013. 1−4. [doi: 10.1109/ICSENS.2013.6688523]

[17] Sgroi M, Wolisz A, Sangiovanni-Vincentelli A, Rabaey JM. A service-based universal application interface for ad hoc wireless sensor and actuator networks. In: Proc. of the Ambient Intelligence. 2005. 149−172. [doi: 10.1007/3-540-27139-2_8]

[18] Aberer K, Hauswirth M, Salehi A. Global sensor network. Technical Report, LSIR-REPORT, 2006.

[19] Guinard D, Trifa V, Wilde E. A resource oriented architecture for the Web of things. In: Proc. of the Internet of Things. 2010. 1−8. [doi: 10.1109/IOT.2010.5678452]

[20] Wang W, De S, Toenjes R, Reetz E, Moessner K. A comprehensive ontology for knowledge representation in the Internet of things. In: Proc. of the IEEE 11th Int'l Conf. on Trust, Security and Privacy in Computing and Communications. 2012. 1793−1798. [doi: 10.1109/TrustCom.2012.20]

[21] Miorandi D, Sican S, Pellegrini FD, Chlamtac I. Internet of things: Vision, applications and research challenges. Journal of Ad Hoc Network, 2012,10(7):1497−1516. [doi: 10.1016/j.adhoc.2012.02.016]

[22] Bianchini D, DeAntonellis V. An ontology-based architecture for service discovery and advice system. In: Proc. of the 6th Int'l Workshop on Database and Expert Systems Applications. 2005. 551−556. [doi: 10.1109/DEXA.2005.49]

[23] Santos S, Guizzardi G, Guizzardi RSS. GSO: Designing a well-founded service ontology to support dynamic service discovery and composition. In: Proc. of the Enterprise Distributed Object Computing Conf. on Workshops. 2009. 35−44. [doi: 10.1109/EDOCW.2009.5332016]

[24] Ji X. Research on Web service discovery based on domain ontology. In: Proc. of the 2nd IEEE Int'l Conf. on Computer Science and Information Technology. 2009. 65−68. [doi: 10.1109/ICCSIT.2009.5234756]

[25] Yang S, Xu Y, He QY. Ontology based service discovery method for Internet of things. In: Proc. of the Internet of Things (iThings/CPSCom). 2011. 43−47. [doi: 10.1109/iThings/CPSCom.2011.104]

[26] Scerri P, Pynadath DV, Schurr N, Farinelli A, Gandhe S, Tambe M. Team oriented programming and proxy agents: The next generation. In: Proc. of the 1st Int'l Workshop on Programming Multi-agent Systems, Vol.3067. 2004. 131−148. [doi: 10.1007/978-3-540-25936-7_7]

附中文参考文献:

[10] 谢磊,陈力军,陈道蓄,谢立.无线传感器网络的查询处理机制研究综述.计算机科学,2006,33(9):45−49, 68.

徐杨(1976－),男,湖北黄石人,博士,副教授,CCF 会员,主要研究领域为分布式人工智能,多智能体系统,知识工程.
E-mail: xuyang@uestc.edu.cn

王晓锋(1978－),男,博士,助理研究员,CCF 会员,主要研究领域为传感器网络,知识推理.
E-mail: wangxiaofeng@ict.ac.cn

何清漪(1987－),女,硕士生,主要研究领域为人工智能.
E-mail: he_qingyi@qq.com