软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2021,32(7):1957−1977 [doi: 10.13328/j.cnki.jos.006259]

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

一种基于分层适应逻辑的自适应系统实现框架*

李念语[1,2], 陈正胤[1,2], 刘 坤[1,2], 焦文品[1,2]

[1](北京大学 信息科学技术学院 计算机科学与技术系,北京 100871)

[2](高可信软件技术教育部重点实验室(北京大学),北京 100871)

通讯作者: 焦文品, E-mail: jwp@pku.edu.cn

摘 要: 自适应系统由于其能够自主地适应具有非确定性的部署环境,并持续地保持用户的满意度,受到了广泛的关注.然而,目前仍然存在未解决的挑战,例如如何在新的部署环境下,或者在开放且复杂的环境下,使得系统仍然能满足自适应性.因此,为自适应系统的设计引入了一个新的概念模型,受归因理论启发,该模型被设计成内归因和外归因两层结构.内归因层决定了内因如何影响自适应行为,这一层与部署环境解耦,可以独立设计且可以复用在不同的部署环境中.外归因层映射了外因与内因的关系,这一层在不同的部署环境中可以被替换.基于两层结构的实现框架,具有设计且实现自适应系统的适用性,以及内因层适应逻辑的可复用性,通过两个案例,一个是被广泛使用的电子商务网络应用,一个是需要躲避障碍物且避免滑倒和翻转的机器人系统,来进行评估.

关键词: 自适应软件系统;归因理论;可复用性;环境非确定性

中图法分类号: TP311

中文引用格式: 李念语,陈正胤,刘坤,焦文品.一种基于分层适应逻辑的自适应系统实现框架.软件学报,2021,32(7):1957−1977. http://www.jos.org.cn/1000-9825/6259.htm

英文引用格式: Li NY, Chen ZY, Liu K, Jiao WP. Internal-external two-layer framework for constructing self-adaptive systems. Ruan Jian Xue Bao/Journal of Software, 2021,32(7):1957−1977 (in Chinese). http://www.jos.org.cn/1000-9825/6259.htm

Internal-external Two-layer Framework for Constructing Self-adaptive Systems

LI Nian-Yu[1,2], CHEN Zheng-Yin[1,2], LIU Kun[1,2], JIAO Wen-Pin[1,2]

[1](Department of Computer Science and Technology, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China)

[2](Key Laboratory of High Confidence Software Technologies of Ministry of Education (Peking University), Beijing 100871, China)

Abstract: The development of self-adaptive systems has attracted much attention as they can adapt themselves autonomously to environmental dynamics and maintain user satisfaction. However, there are still tremendous challenges remained. One major challenge is to guarantee the reusability of the system and extend the adaptability with changing deployment environments, or open and complex environments with the existence of unknown. To solve these problems, a conceptual self-adaptive model is introduced, decoupling the environment with the system. This model is a two-layer structure based on internal causes and external causes from the attribution theory. The first layer, determining how the internal causes affect the adaptation behaviors, is independently designed and reusable whiles the second layer, mapping the relationship between external causes with internal causes, is replaceable and dynamically bound to different deployment environments. The proposed approach is evaluated by two case studies, a widely used benchmark e-commerce Web application and a destination-oriented robot system with obstacle and turnover avoidance, to demonstrate its applicability and reusability.

* 基金项目: 国家重点基础研究发展计划(973)(2016YFB000105, 2015CB352200); 国家自然科学基金(61620106007)

Foundation item: National Basic Research Program of China (973) (2016YFB000105, 2015CB352200); National Natural Science Foundation of China (61620106007)

本文由“面向非确定性的软件质量保障方法与技术”专题特约编辑陈俊洁副教授、汤恩义副教授、何啸副教授以及马晓星教授推荐.

收稿时间: 2020-09-05; 修改时间: 2020-10-26; 采用时间: 2020-12-14; jos 在线出版时间: 2021-01-22

Key words: self-adaptive software; attribution theory; reusability; uncertainty of environment

当今社会广泛地依赖软件系统来帮助人们实现特定的目标.然而,具有非确定性的部署环境,例如负载量的变化,会导致昂贵的重配置和耗时的维护任务[1],使得实现软件系统的目标并不容易[2].因此,为了降低管理的复杂度,为了在合理的成本中迅速实现预期目标,自适应系统即被提了出来.这类系统由于能够自主地在具有非确定性的部署环境中不间断地实现系统目标和用户需求,包括性能、安全性、故障管理等,因而被认为是管理现代软件系统不确定性的最有前景的方向[3].

在已有的自适应系统研究中,最主要的一类自适应决策方法是基于规则的方法[4–9],预先定义好的自适应规则规定了系统的适应逻辑,指定了在部署环境发生变化时应当执行的自适应动作.也就是说,自适应行为也是由部署环境中的事件引发的[10–14],即软件系统的自适应性是对外部环境因素变化的内部响应.因此,基于规则适应逻辑的自适应系统的生命周期,包括设计阶段和运行阶段,总是将系统与环境绑定在一起.在设计阶段,识别出具有非确定性的且对系统自适应行为有影响力的环境因素,并定义和定制出一套自适应规则形式,并通过仿真和模拟等方式来定制出具体的自适应规则,以映射环境因素与自适应动作之间的关系.在运行时,通过 MAPE-K(Monitor 监测、Analysis 分析、Planning 规划、Execution 执行和 Knowledge 知识)闭环并推理自适应规则以实现自适性行为[7,8,10,14,15].

然而,环境开放且十分复杂,存在很多设计人员在设计时未曾考虑或不能完全理解的非确定性环境因素,使得这样设计出来的自适应规则及其系统实现会不可避免地忽略一些环境因素[8].在不同的部署环境中,这些被忽略的环境因素可能存在不同的状态,从而对系统的自适应性产生不同的影响.这就使得设计阶段制定好的自适应规则很可能不适用于某个具体的部署环境,而如果针对某部署环境进行更新和调整的自适应规则,也不一定能较好地适应新的部署环境,这就不可避免地限制了系统对于各种不同部署环境的自适应性.

以一个机器人系统为例,我们可以提前设计出一些规则来描述机器人应该以怎样的速度前进,才能尽快且安全地到达目的地;或者在遇到障碍物时应调整多少角度来躲避障碍物.然而,如果我们事先不知道摩擦力的影响,也并未在自适应规则中考虑这个环境因素,那么能够适应木地板的机器人系统及其自适应规则,很可能在湿滑的地面上不能很好地完成目标任务[12].此外,在设计机器人避开障碍物的自适应规则时,需要事先知道机器人可能遇到什么样的障碍物,然后明确如何应对这些障碍物.显然,在实际应用环境中,障碍物可能是无限的,总会有新的、意想不到的障碍物出现,使得系统的自适应能力不足,无法应对.然而,对于一个好的自适应系统来说,除了在一个具体的部署环境中能够适应外,还应该具有广泛的自适应性(即在各种部署环境中均具有良好的适应性).

造成这些弊端的根本原因在于设计规则时环境因素与系统动作之间存在的绑定,从而不能处理设计时环境与具体的部署环境、部署环境与部署环境之间存在的隐式的差异.为了应对当前的挑战,本文提出了一种基于归因理论的自适应系统设计方法,找出自适应性行为的原因,并基于此,将适应逻辑分为两层.在认知科学领域,内因是事物变化或发展的根本原因;而外因只是条件,需要通过内因来对事物的变化产生影响.换言之,只有当外因导致了内因的变化时,才可能引发系统的自适应行为.因此,我们的基本思路是将自适应系统的适应逻辑分为独立设计和部署绑定两个阶段,从而将部署环境与系统动作解耦.在设计阶段,关注和强调内因如何决定系统的自适应行为,使得系统的设计和开发独立于实际部署环境.在部署阶段,建立部署环境的非确定性因素(即外因)与系统状态(即内因)之间的关系,并且可以在运行时动态调整,从而将系统与具体的部署环境(即部署)绑定,实现与部署环境相关的自适应性.

我们提出了一个新的基于归因的理论来设计自适应系统的概念模型,并按照模型提出内归因层和外归因层两层结构的实现框架.第 1 层是独立设计,通过决定性自适应规则,即内层适应逻辑,来映射出内因与自适应行为的关系;第 2 层是部署绑定,通过影响性自适应规则,即外层适应逻辑,来连接部署环境的外因与系统内因.在此基础上,我们通过两个案例来评估已提出的方法,在一个被广泛使用的基准电子商务网络应用以及一个具有避免障碍物和避免滑倒翻转的机器人上进行了实验.进一步地,我们将由内外因模型实现的系统与他人工作中的实现进行实验比较以证明内外因两层框架对于自适应系统的适用性(即对特定部署环境中的非确定性具有自适应性,以及对各种部署环境的自适应性)、内层适应逻辑(即决定性自适应规则)的可复用性.

本文第 1 节概述基于归因理论的概念模型,并为内归因和外归因提供例子说明.第 2 节给出概念模型的形式定义,包括内因和两类自适应规则.第 3 节介绍独立设计和部署时绑定的自适应架构的实现.第 4 节通过实验结果对我们的方法和现有方法进行比较,对内外因两层结构进行评估.第 5 节详细介绍一些相关工作.最后一节对本文进行总结,并指出未来的研究方向.

1 方法概览

提出和回答“为什么”问题——试图弄清楚是什么导致了一件事——这一基本过程被定性为人类最基本的活动.由弗里茨-海德(Fritz Heider)[16]发起,哈罗德-凯利(Harold Kelley)和伯纳德-韦纳(Bernard Weiner)[17]进一步推进的归因理论,试图描述和解释个人的日常行为或事件.例如,一个人生气了,可以归因于他的坏脾气,也可以归因于坏事情的发生.

归因有多种定义,但常见的定义方式是将归因定义为解释和理解个体行为背后的内部过程和外部过程.外部归因又称为情境归因,是指将行为的原因解释为人所不能控制的情境或环境特征.内部归因是指将行为的原因归结为某些人格特征,而不是外部力量.

同样地,我们也需要确定系统自适应性行为的原因,即系统为什么要适应?这也是影响自适应性行为的核心问题.一般来说,产生自适应行为的原因是由于部署环境的非确定性:i) 技术资源的变化.例如,可供选择的网络连接发生了变化;ii) 环境变量的变化.例如,网站的负载量发生了变化;iii) 用户的变化.例如,用户目标或用户偏好发生了变化[18].用户和技术资源可以看作是部署环境的一部分,与环境因素一起构成了自适应系统的外围[19],它们是系统存在的外部条件,也被称为外因.

然而,环境因素(即外因)不是系统执行自适应行为的必要条件.例如,对于部署在云上的网站,当活跃用户数量增加时,有些网站会饱和,而有些网站不会.事实上,网站是否饱和,从而需要动态地分配更多的资源(服务器),取决于响应延迟.如果响应延迟是在一个让人可接受的范围内,即使该网站有大量的活跃用户,也不需要进行自适应调整.换句话说,用户数的变化并不能决定资源分配上的自适应行为.相反,只有当变化影响了响应延迟并进一步损害了系统目标时(即内因),这种变化才会触发自适应行为.因此,内因是系统自适应行为的根本原因.

基于此,我们将一个自适应系统的适应逻辑分为两层,分别对应了内部归因过程和外部归因过程,如图 1 中的概念模型所示.

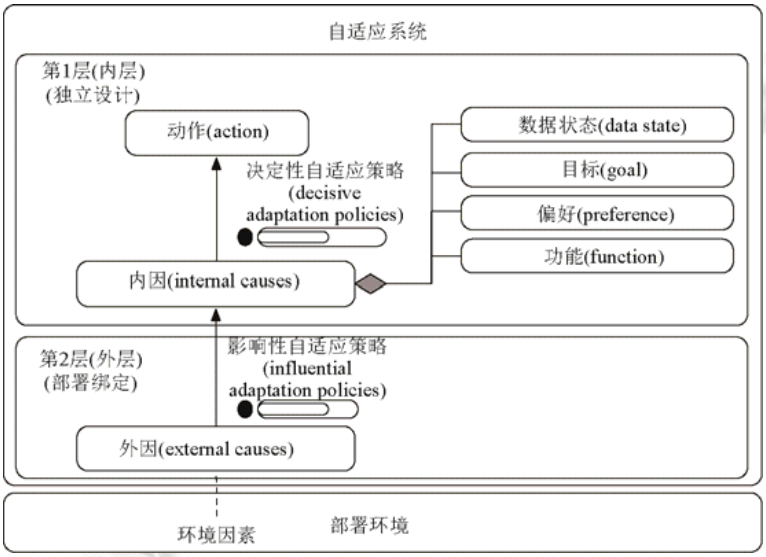

Fig.1 Conceptual model of attribution-based self-adaptive systems

图 1 基于归因理论的自适应系统的概念模型

图 1 中,第 1 层,即由内因(包括数据状态、功能、偏好和目标,将在第 3 节加以阐述)、自适应行为(即动作)以及决定性自适应规则(决定内因和自适应行为之间的关系从而应对内因的变化)组成.第 2 层由外因和内因之间的关系即影响性自适应规则表示,在部署环境确定时,这一层会被绑定,并且可以在部署环境改变时或者在部署环境扩展时替换,而独立设计的第 1 层适应逻辑则固定不变.值得注意的是,部署环境变化是指从一个部署环境转移到另一个部署环境,而部署环境扩展是指设计或开发人员随着对开放且复杂的部署环境的进一步了解,从而存在一些环境因素从未知变成已知,也可被认为是部署环境变化的另一种形式.

Fig.2 Running example—A disaster relief system

图 2 驱动例子——无人机救灾系统

举例说明:为了进一步解释本文的方法和形式化描述,我们用无人机救灾系统[7]作为简例说明,如图 2 所示.在一个灾难场景中,城市中的通信基础设施因灾害而受损;城市中的部分地区可能不安全.危险区域被划分为几个街区(block),受害者分布在不同的街区且不知道救援中心所在位置.无人机(UAV)将被派遣去搜索救援受害者,并引导受害者到达救援中心.在搜救过程中,无人机不仅要保证搜救任务快速、彻底(在可接受的时间内搜救出所有遇难者),还要保证其安全性(不坠毁)和储能(避免电池耗尽).我们期望这种无人机救灾系统能够被部署到不同的灾害环境下进行搜救,如火灾、洪水、地震等,尽管每个部署环境中能对自适应行为产生影响的环境因素大不相同.

2 概念模型的形式定义

受归因理论的启发,部署环境中的环境因素(外因)通过内因对系统产生影响,而不是直接影响或决定系统的自适应行为.我们将基于内外因的概念模型 *M* 定义为一个三元组 *M*=(*IC*,*DAP*,*IAP*),其中,

1) *IC* 是内因(即自适应行为的内在原因),可以进一步指定为元组 *IC*=(*Data State*,*Goal*,*Preference*,*Function*);

2) *DAP* 是决定性自适应规则,它定义了系统的内部原因如何决定自适应行为,一般表示为“内因→行动”形式;

3) *IAP* 是影响性自适应规则,表示环境因素如何影响内因的变化,其形式为“环境因素→内因”.

2.1 内因与系统状态

数据状态(data state).它是需要被记住的系统数据信息,由属性值的集合表示.假设 *Attr*={*a*1,...,*an*}为系统的属性集合,*Dom*={*dom*(*a*1),...,*dom*(*an*)}为这些属性的域集,那么系统的数据状态就是这些属性与其值的映射.在示例中,无人机系统需要存储一定的数据信息,如当前位置、剩余能量、飞行高度、已搜索块、未搜索块、危险块、状态(即巡航或受害人引导).无人机的数据状态是由这几种数据项的值来定义的.

目标(goal).它是系统期望达到或保持的数据状态.一般来说,目标可以分为 3 类[20]:一类目标是追踪一个参考值,又被称为设定点,在这种情况下,目标是使一个或多个变量尽可能地接近设定点;第 2 类目标是将变量居于特定的范围内,有置信区间;第 3 类目标涉及某个变量的最小化(或最大化).目标是期望的某一个或者某些数据状态:

$$Goal=\{g \mid g \in 2^{DataState} \wedge eval(g)=1\}.$$

评估函数 *eval* 用来计算数据状态是否满足期望,如果当前数据状态 *g* 满足期望,即 *eval*(*g*)=1,就可以认为系统完全实现了目标;但在某些情况下,系统只能(无限地)接近目标而不能达到目标.例如,要求无人机的飞行速度“准确地保持在每秒 2 米”是很难做到的,只能说系统在一定程度上实现了目标.在救灾任务的场景中,如果所有的区块都被搜救过了,那么这个目标的评价结果是 1.如果没有,则评价函数为 *eval*(*g*)=*num*(*searched blocks*)/*num*(*total blocks*),即目标的满足度与巡航的街区数量成正比.

偏好(preference).它是系统感兴趣的数据状态.与系统必须实现或维持的目标相反,这些偏好并不是必须要做的,但若能满足这些偏好,系统的性能会更好.例如,无人机不仅要完成搜救目标,还要有经济性(耗电少)和快速性的要求(要尽快搜索到所有遇难者).与目标类似,偏好也有一个衡量偏好满足程度的效用函数.例如,存在一个效用函数:*util=residual power/total energy storage*,显示了耗能少的倾向.

$$Preference=\{p \mid p\in 2^{DataState}\wedge util(p)\in[0,1]\}.$$

功能(function).它是实现目标的手段或方法.功能通过操纵可控变量来改变数据状态.例如,无人机具有起飞、着陆、改变方向等功能.

$$Function=\{f|f: DataState\rightarrow DataState\}.$$

系统的状态是由内部因素决定的.换句话说,数据状态、功能、目标和偏好的满足共同定义了系统状态.

2.2 自适应规则

在开放且复杂的环境中,存在着多种(已发现和待发现的)对自适应行为有影响且具有不确定性的环境因素[19,21],并不是说所有环境因素的变化都会影响系统内因,从而引发适应行为,而是只有那些导致内因变化的因素才会对自适应系统产生影响.

2.2.1 影响性自适应规则(influencial adaptation policies,简称 IAP)

IAP 描述了环境因素或外因如何影响系统的内因变化.这些环境因素,可能会影响某些数据状态,进一步影响功能、偏好和目标.IAP 在系统部署时形成,与部署环境绑定,且可以在运行时动态更新以更好地映射环境因素对内因的影响.

$$IAP=\{ap_i|ap_i: ExtFactor\rightarrow Internal\ Causes\}.$$

在火灾部署场景中,环境因素可能是火灾的严重程度,这必然会影响无人机系统内的数据状态.例如,若检测到某街区情况严重(即火势大),无人机会将其标记为危险,为了自身安全,无人机会尽量避开这一街区的巡航.但如果在地震部署场景中,来自地面的障碍物(对高空飞行的无人机没有威胁)可能不会影响到某街区的标记.

2.2.2 决定性自适应规则(deterministic adaptation policies,简称 DAP)

DAP 决定了内因如何影响系统的自适应动作.动作是功能的具体操作 $Action=\{a|f\epsilon Function, a=Do(f)\}$. 决定自适应动作的因素可能涉及数据状态,系统拥有的功能、偏好和目标.对于无人机而言,如果没有检测到需要引导的受害者,所有的街区都没有被巡航过,且暂时没有危险标记,则可以进行 4 个方向的飞行变化(东、南、西、北)的动作.

$$DAP=\{ap_d|ap_d: Internal\ Causes\rightarrow Action\}.$$

上述概念模型明确定义了内因,并通过IAP推理外因对内因的影响,在此基础上,再通过DAP获得自适应动作,从而实现自适应行为.在内外层两层结构下,这个概念模型具有适用性和可复用性的特点.适用性是指,基于归因理论设计的自适应系统,在具有非确定性的部署环境中,能够通过推理 IAP 和 DAP,从而持续地满足系统的目标.可复用性是软件工程领域的一个重要研究点,是指软件系统产品开发过程中以某种形式复用已有的产品,包括代码、软件组件、测试套件、设计和文档等[19].在本文中,可复用性描述了内层适应逻辑,即 DAP 的不变性和可用性,特别是与部署时绑定的 IAP 协调,共同满足变化部署环境或扩展部署环境中的自适应性.

3 实现框架

本节展示了基于归因理论概念模型上的实现框架.自适应逻辑建立在自适应规则的基础上,这些规则表征了环境因素(外因)与系统内因之间、系统内因与自适应行为之间的因果关系.自适应行为是通过实施 MAPE (Monitor 监测、Analysis 分析、Planning 规划、Execution 执行)[10,15]循环来实现的,分析和规划活动分别负责识别可能的目标违规和生成自适应决策,而监测和执行则负责在运行时实施决策.

3.1 知 识

为了实现自适应,需要利用知识——自适应规则.知识构件(knowledge)由所有 MAPE 构件共享.理想情况

下,所有的知识应该是可以在同一类系统中复用的,即这一类系统可以适应于各种部署环境,实现系统目标.然而,这个构件通用性的代价如下.

- 需要指明和维护所有对自适应行为有影响的环境因素,以便将系统应用于不同的部署环境中.
- 如果随着环境信息的增加而需要改变知识构件,则整个构件需要修改或调整,以帮助(正确地)满足系统的目标.

为了支持对复杂且开放环境的不完全可知、对环境认知随时间的深入以及支持自适应规则在不同部署环境之间的复用,我们将系统特定知识与部署环境特定知识分开,以呼应概念模型中的两层结构.系统特定知识用决定性自适应规则(DAP)表示,映射自适应动作与系统内因之间的关系,这类知识是固定的,支持不同部署环境中的可复用性.同时,定义环境因素(即外因)如何影响内因的环境特定知识,以影响性自适应规则(IAP)的形式存在,随着部署环境的变化或部署环境的扩展(即发现新的有影响力的环境因素),该知识是可以改变和调整的.这符合关注点分离的原则——将设计分为不同部分,且每一部分都处理一个单独的关注点[9].

3.2 MAPE闭环

对于特定的部署环境,自适应行为将由被扩展的 MAPE 循环机制来实现,其实现框架如图 3 所示.

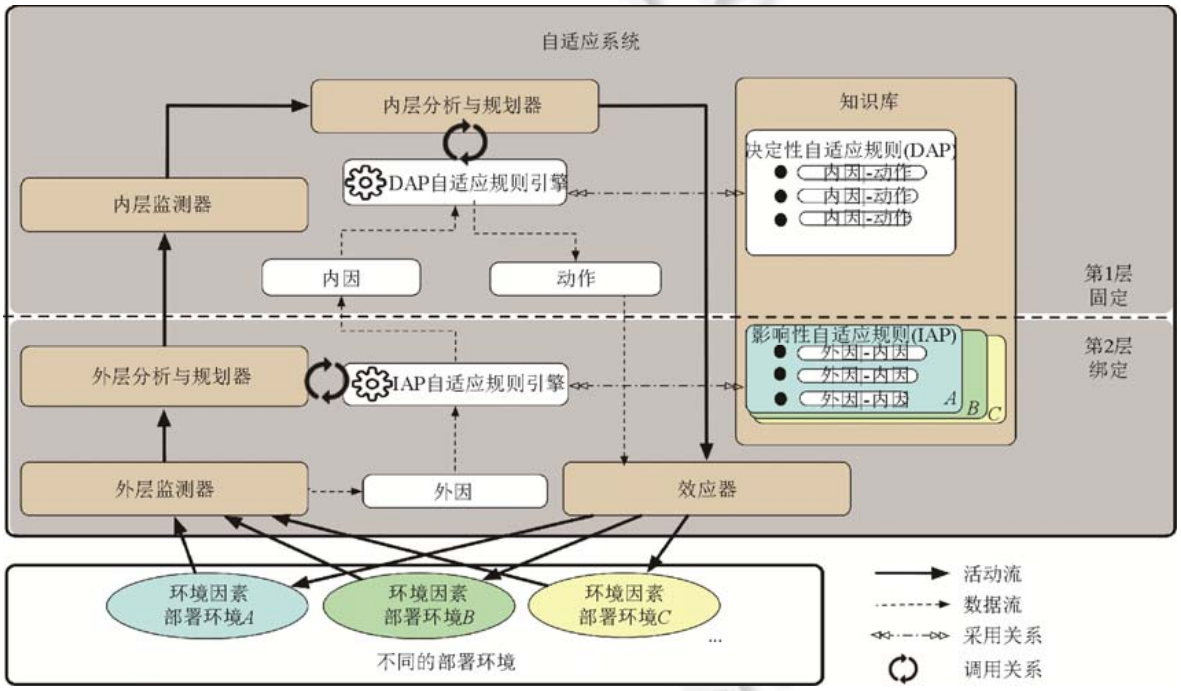

Fig.3 Implementation model of attribution-based self-adaptive systems

图 3 基于归因理论的自适应系统的实现模型

部署环境中产生的事件表明了环境因素的变化,外层监测器通过探针(或传感器)从部署环境中收集或合成特定数据,并以外因的形式保存数据.一般来说,不同部署环境中的环境因素可能不一样,甚至可能有很大的差异.以无人机救灾系统为例,环境中的事件可能表明巡航无人机在火灾救灾部署场景中检测到了火势,或者在洪水救灾部署场景中检测到了水位上升.在推测分析过程中,外层分析与规划器构件接收更新的外因(由外层监测器输入),通过调用自适应规则引擎来推理影响性自适应规则(IAP)从而获取系统的数据状态,并进行分析.在此基础上,外层分析与规划器构件进一步检查系统是否达到目标、是否满足偏好、是否需要自适应调整.一个典型的例子是,在火灾或地震的情况下,可能危及无人机安全的烟雾或坠落物,会导致新的街区被标志为危险,即改变系统的数据状态和内因.外层(即第 2 层)是在部署时根据不同的部署环境绑定的,且可在部署环境变化时加以替换.

内层监测器构件接收更新的内因(由外层分析与规划器构件输入),并传给内层分析与规划器构件,该构件调用自适应策略引擎并将内因作为引擎的输入,推理决定性自适应策略(DAP),生成自适应行为的动作流,以消除目标的违规或者更好地实现目标.动作流由一个或一组从决定性自适应策略(DAP)中推理出的行动组成,决定性自适应策略在设计阶段被制定出来且固定于变化的部署环境.对于每一种情况,引擎会匹配策略中的条件,并规划出自适应动作,如果存在无法处理的情况,则系统目标无法满足,就会触发系统设计的变更.在无人机救援系统中,面临危险情况(即新的街区被标记为危险),改变方向或安全降落是可行的自适应动作.在执行过程中,自适应动作通过效应器构件在系统上实施.

4 案 例

为了评估本文提出的内外因概念模型及其对应的实现框架,我们采用两个来自不同领域、具有不同自适应目标和自适应规则的案例.在实现框架中,还采用了电子商务网站系统和机器人系统,电子商务网站需要响应用户的请求,而机器人在趋向目的地时不仅需要避开障碍物,还需要避免摔倒,我们为这两个系统分别实现了基于内外两层的框架,实现源码公开(https://github.com/easton-chen/SASAT-exp).本节分别阐述这两个案例、实验设置、实验结果并进行探讨.希望通过这两个案例得出:(1) 基于内外两层框架设计自适应系统的适用性,即与已有的前沿方法及其实现进行比较,以获得同等水平的目标满意度;(2) 内层逻辑尤其是决定性自适应规则(DAP)的可复用性,即在部署环境变更或者扩展时,能与更新的影响性自适应规则(IAP)相结合共同实现系统目标.

4.1 案例1:电子商务网站系统

电子商务拍卖网站 RUBiS,与 eBay 相似,已在云计算研究中被广泛应用[22−24].在电商网站中,针对每个用户请求,系统需要将用户请求的特定产品信息回传,当成一个基础请求的响应.此外,系统还可以推荐用户检索的同类产品,从而提升用户体验感受.推荐率为包含系统推荐产品的请求响应数与总的请求响应数之比,增加推荐率会增加推荐产品的响应概率,但也会增加网站的资源使用,从而有可能影响到响应延迟.针对 RUBiS,我们识别并总结出概念模型中的内因以及自适应规则,如图 4 所示.

<table>
<tr><th colspan="3">电子商务系统的概念模型
Conceptual Model for E-commercial Website</th></tr>
<tr><td rowspan="4">内因
IC</td><td>数据状态
data state</td><td>response latency : resl $dom(resl) = (0, +\infty)$
preference weights: w1, w2, w3 $dom(w) = [0, 1]$
recommendation probability: pr $dom(pr) = [0, 1]$</td></tr>
<tr><td>目标
goal</td><td>g1: response latency below service level agreement
$eval(g1) = resl \le SLA\ ?\ 1 : 0$
g2: system performance maximization
$eval(g2) = max(w1 \times p1 + w2 \times p2 + w3 \times p3)$</td></tr>
<tr><td>偏好
preference</td><td>p1: high recommendation
$util(p1) = pr \times 100\%$
p2: quick response
$util(p2) = (1 + 0.2 \times resl - 1.2 \times resl^2) \times 100\%$
p3: low timeout
$util(p3) = (1/\exp(10 \times resl)) \times 100\%$</td></tr>
<tr><td>功能
function</td><td>pr = probadjust (x)
where x is the recommendation probability to be set</td></tr>
<tr><td>决定性自适应规则
DAP</td><td>规则形式:</td><td>$resl \times w1 \times w2 \times w3 \rightarrow probadjust\ (x)$
……
$0.21, 0.88, 0.06, 0.06 \rightarrow probadjust\ (0.90)$
$0.31, 0.88, 0.06, 0.06 \rightarrow probadjust\ (0.75)$
……</td></tr>
<tr><td>影响性自适应规则
IAP</td><td>规则形式:</td><td>$upr \times ccp \times usn \rightarrow resl \times w1 \times w2 \times w3$
……
$1, 300\%, 1000 \rightarrow 0.21, 0.70, 0.20, 0.10$
$1, 200\%, 1000 \rightarrow 0.31, 0.88, 0.06, 0.06$
……</td></tr>
</table>

Fig.4 Conceptual model for E-commercial Website

图 4 电子商务系统的内外因概念模型

内因.电子商务系统中的一个目标是将响应延迟(response latency,简称 resl)控制在服务级协议(service-level agreement,简称 SLA)[22]范围内.例如,1s 内,并忽略网络传输时间(即 *resl*<1s).另一个目标是使系统性能最大化(即 max($w1\times p1+w2\times p2+w3\times p3$)),其中,$p1$、$p2$、$p3$ 代表了 3 种不同的服务质量要求(即偏好).

$p1$:高推荐.尽可能地保持较高的同类产品推荐率.

$p2$:快响应.尽可能快速地响应用户请求,即响应延迟越低越好.

$p3$:低超时.表示在服务级别协议内,尽可能保持较低的超时响应比例.

其效用函数如图 5 所示.$w1$、$w2$、$w3$ 是分配给每个偏好的权重,且 $w1+w2+w3=1$.*resl* 和 3 个权重值为系统内因数据.此外,电商网站的功能是通过调整推荐率,即可配置的参数 *pr*(也是系统内因数据)来实现资源需求的调控.

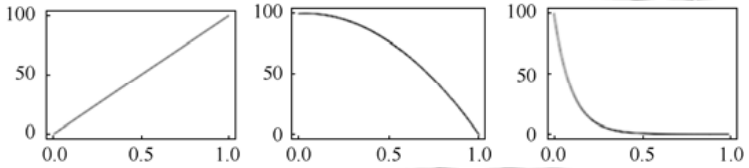

Fig.5 Utility functions for three preferences

图 5 3 个偏好的效用函数

外因.关于环境因素,对于这个运行在动态云环境中的应用,我们识别出 3 个具有非确定性的变量.

1) 用户优先级(user priority,简称 upr),表示为 3 个非功能需求,即偏好的优先级,3 个偏好对应了 6 种不同的优先级排列方式.例如,第 1 种代表高推荐的优先级高于快响应,而快响应又优于低超时,即 $p1>p2>p3$,而第 2 种代表 $p1>p3>p2$.

2) 计算能力(computing capacity 简称 ccp),由分配给该网站的 CPU 资源量来衡量.分配 400%的 CPU 意味着这个网站可以独占物理机的 4 个核心,而 50%则意味着可以访问物理机的一个核心,且仅有一半的使用时间.

3) 用户数(user number,简称 usn),表示活跃的用户数.

我们将这些连续环境变量离散化,划分为网格化的离散点,如下表所示,以缩小自适应规则空间.

环境变量	取值区间	#离散点数	间隔
upr	1~6	6	1
ccp	50%~400%	8	50%
usn	100~4000	40	100

自适应规则.与部署环境绑定影响性自适应规则(即 IAP)的形式为“upr,ccp,usn→resl,w1,w2,w3”.例如,当优先级为 1,分配给系统 3 个完整 CPU 且当前有 100 个活跃用户时,将导致响应延迟为 0.31s,$p1$、$p2$、$p3$ 的最优权重分别为 0.88、0.06、0.06.具体来说,优先级越高,分配给偏好的权重越多.而在设计阶段独立制定的决定性自适应规则(即 DAP)的形式为“resl,w1,w2,w3→probadjust(x)”.例如,当响应延迟为 0.31 时,即该数据状态远低于服务级协议设定的 1s,高推荐率的偏好可以进一步提升.因此,自适应动作将 *pr*(即推荐概率)提升为 75%,从而使得系统性能最大化.规则的获取方式将在实验设置中进一步加以阐述.

由于环境开放且复杂,可能会有一些未知的环境因素会影响系统目标的满意度.例如,在不同的云部署环境下,CPU 的处理能力(本文中,计算能力是指可用的 CPU 数量,而处理能力是指每个 CPU 的计算频率)不同,会导致相同环境因素(即 upr,ccp,usn)下的响应时间不同.另外,用户思考时间,即用户两次请求的间隔时间,也会影响实际的负载量,从而影响响应时间.如果开发者未能准确地识别出这些隐性的环境因素,适应于某一个部署环境的自适应规则很可能不能有效地适用于另一个部署环境中(即两个部署环境中的处理能力或用户思考时间有明显差异).在接下来的实验中,我们假设处理能力这个环境因素未被系统开发者知晓,从而未被显示地建模成外因以及被考虑进自适应规则中,其对系统目标和偏好的影响是隐示的.另外,我们假设用户思考时间原本也是未知的环境因素,然而随着时间的推进,该因素被发掘出来,用以代表部署环境的扩展.

4.1.1 实验设置

我们基于内外因的概念模型构建了一个电商网站的自适应系统,此外,我们将 Klein 等人[25,26]扩展 RUBiS 搭建的自适应拍卖网站 Brownout 作为基准,并将基于内外两层的自适应系统实现与之进行性能比较.然而,Brownout 中的适应策略是基于控制理论的,而本文方法实现的系统的自适应策略是基于规则的,为了使这两种方法具有可比性,我们将一个时期内推荐概率的稳定值作为Brownout的自适应动作.我们试图评估:(1) 适用性,即用内外两层框架开发和实现的自适应系统(attr-based)与前沿研究中的实现(brownout)具有不相上下的目标满意度;(2) 决定性自适应规则(DAP)的可复用性,即在部署环境变化或者扩展(存在环境变量从未知到已知的扩展)时,DAP 和新部署环境绑定的 IAP 一起依然能够满足系统目标,并优于非内外因实现的自适应系统目标满意度.

适用性实验在一个具有四核 i7 3770 3.40GHz 处理器的虚拟机上进行,我们开发了一个自定义工具 Cap 作为管理程序来控制 CPU 资源分配,并采用了一个自定义工具 httpmon 来动态调整活跃的用户数,通过客户端线程数来模拟工作负载.然后,我们将推荐率划分为 20 个区间(即从 5%~100%),并对每个值进行测试,定义算法为所有可能的响应延迟和偏好权重下找到最大化系统性能的推荐率值,作为 DAP.接下来,我们在该虚拟机上生成 1 920 条数据记录(即连续环境变量离散化表格中所示的 6×8×40),在此基础上,通过训练具有 3 个隐藏层且每个隐藏层有 1 000 个神经元的神经网络,生成 IAP.

对于可复用性实验,一方面,我们将一台采用四核 i5 5200U 2.70GHz 计算频率较低的处理器作为变化的部署环境,在保持 DAP 不变的前提下,对新环境中的 1 920 条数据进行再训练生成新的 IAP.另一方面,在原部署环境(即 i7 3770 3.40GHz 处理器)中,将用户思考时间(think time,简称 tht)视为每条记录中需要收集的另一个环境因素,即 IAP 形式更新为“upr,ccp,usn,tht→resl,w1,w2,w3”,重新训练生成 IAP.

4.1.2 实验结果

适用性实验.如图 6(1)所示,3 组实验中,每组实验关注一个环境因素的变化(另外两个环境因素保持不变).实验结果分别记录了在变化的环境因素中,基于归因模型实现的平均响应延迟和系统性能,这些环境因素(外因)通过内因影响自适应目标,通过推理影响性自适应规则和决定性自适应规则生成自适应行为.实验结果也记录了 Brownout 的平均响应延迟和系统性能,其中在变化的计算能力和用户数量中,两种实现方式的响应延迟几乎是一样的,而在变化的优先级中则有些不同.对这种差异的合理解释是,由于偏好优先级不同,在基于内外因的方法中,会更多地考虑低超时,而不太注重快速响应.在系统性能方面,将外因和内因分离,系统有能力适应环境变化,性能甚至比 Brownout 更好,这是因为 Brownout 不能调整偏好的权重.总体来说,Brownout 和基于内外因的实现,都能以相对较低的平均延迟满足系统目标,性能也维持在较高的水平.

可复用性实验(部署环境变化).如图 6(2)所示,这组实验在一个计算频率较低的处理器上进行,我们将这种差异性视为部署环境的隐示变化,即不显示地将其当成环境因素,但这种差异的确会改变环境因素与自适应行为之间的关系.例如,在具有 3 个核、1 000 个活跃用户、偏好优先级为 1 时,在这一新的部署环境中,响应延迟为 0.38s,权重为 0.48、0.46、0.6,导致了 65%的推荐率,这与原部署环境中 75%的推荐率不同.如第 2 组实验结果所示,在两种实现中,由于计算频率、平均响应延迟均升高,对于 Brownout 来说,由于自适应策略和自适应行为与上一个部署环境绑定,在新的部署环境下,它不能有效地适应环境因素的变化,如图 6 所示,其适应能力有较大幅度的下降(系统性能下降超过 10%).在基于归因的实现中,我们对这次新部署收集到的记录重新训练生成 IAP 并使用原部署中的 DAP,明确考虑隐示因素的效应.虽然由于处理能力较低,性能仍然有所下降,但与 Brownout 相比,性能下降得较少,相较优势明显.因此,尽管存在 IAP 的重新训练成本,但是基于归因实现的相对性能有所提升,即与 Brownout 的性能差异增大.

可复用性实验(部署环境扩展).在第 3 组实验中,DAP 不变,我们在新一轮的 IAP 训练中加入了用户思考时间,象征着环境因素的新发现.在第 1 组的实验中,由于已经有了良好的系统性能,性能的提升空间并不大.但是随着 IAP 的更新,第 3 组的性能得到进一步的提升,如紫色区域所示.这个结果符合我们的直觉——对环境了解得越多,自适应系统就越能进一步得以改进,性能也就越好.

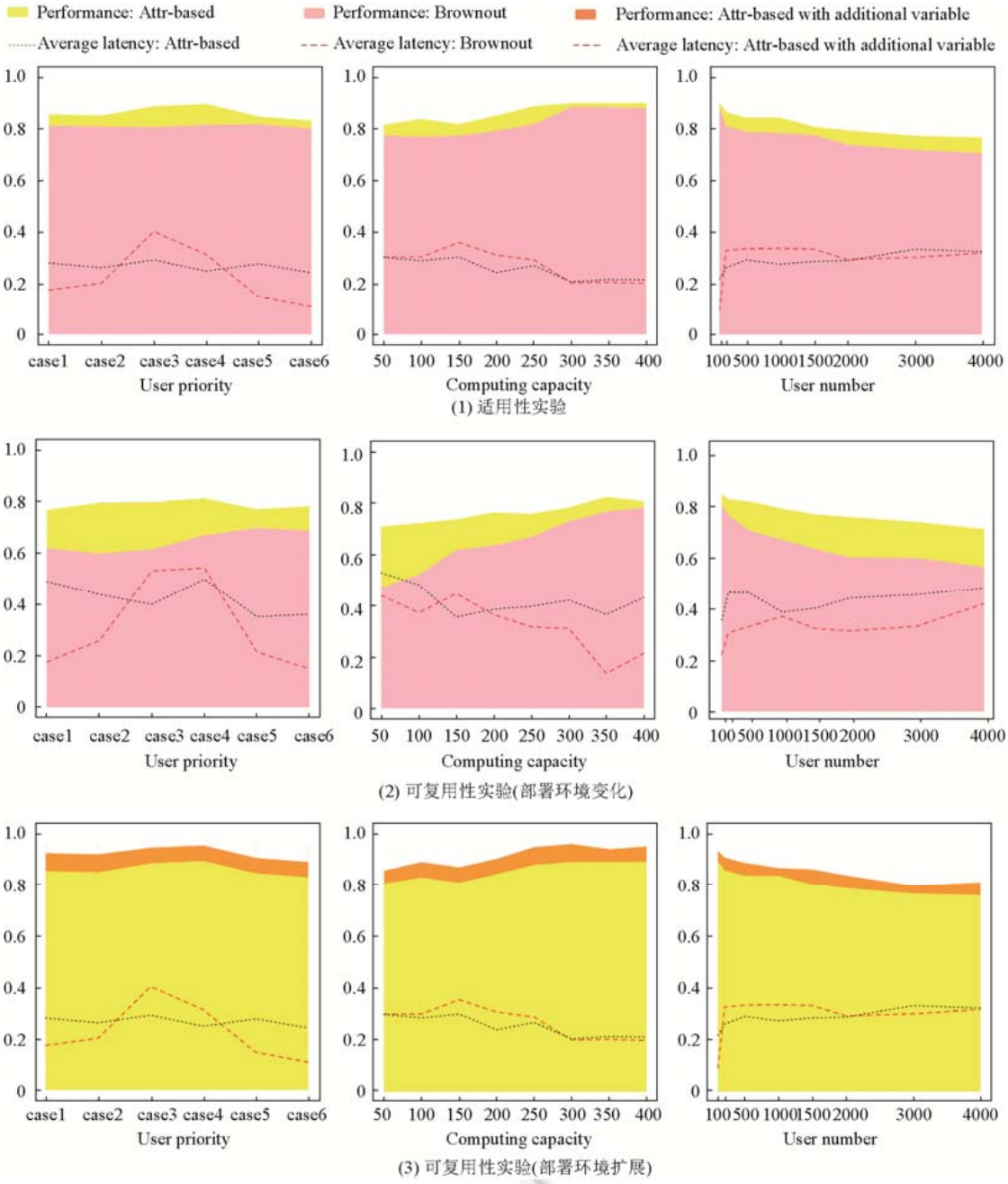

Fig.6 Experiment results for E-commercial Website

图 6 电子商务系统的实验结果

4.2 案例2:移动机器人

近年来,自主移动机器人领域得到了许多研究者的关注,设计一种在不确定环境中面向目的地并具有避障和避免翻转功能的机器人,是移动机器人在各种应用需求中的基石[27].躲避障碍物可以使机器人能够在不发生碰撞的情况下到达目的地.此外,户外环境中的移动机器人还需要在不平坦或倾斜的地形上行动,防止移动机器人滑倒和翻转也至关重要[28].

我们采用 Pioneer 3-DX 机器人作为内外因应用的另一个案例研究.Pioneer 3-DX 机器人是一款理想的小型轻量级机器人,它有两个轮子分别由两个差速器驱动,且配有红外传感器来检测障碍物.该机器人由于具有通用性和可靠性使其被很多人作为研究的首选,其概念模型中的具体内因与自适应规则描述如图 7 所示.

移动机器人的概念模型 *Conceptual Model for Autonomous Robot*		
内因 ***IC***	数据状态 data state	obstacle distance : od response values : rv_l, rv_r balance degree : bd_x, bd_y, bd_z motor velocities : vel_l, vel_r
	目标 goal	the destination on the three-dimensional coordinate *eval*(g) = (x, y, z) == (x', y', z') ? 1 : 0
	偏好 preference	time preference : as soon as possible, safety preference : p1 : no obstacle collision *util*(p1) = od ≥ threshold ? 1 : 0 p2 : no turnover *util*(p2) = bd_k ≥ 1.8 ? 1:0 k = x,y,z
	功能 function	vel_l = *setleftmotor*(x1) vel_r = *setrightmotor*(x2) where x1 and x2 are the velocities to be set for motors
决定性自适应规则 DAP	规则形式:	od × rv_l × rv_r × bd_x × bd_y × bd_z →*setleftmotor*(x1) × *setrightmotor*(x2)
影响性自适应规则 IAP	规则形式:	distance × color × occlusion → rv_l × rv_r degree × distance × load → bd_x × bd_y × bd_z

Fig.7 Conceptual model for autonomous robot system

图 7 自动机器人的内外因概念模型

内因.自动机器人的系统目标是将机器人移动到目的地,机器人的控制器通过定义自动驾驶功能,并利用机器人上的全球定位系统(简称 GPS)和北斗装置测得的数值,来实现目标.在完成目标的过程中,控制器应尽快完成任务,并保证其安全、无障碍物碰撞、避免翻转(偏好).以下是对内部数据的描述,为了简化,仅描述与自适应行为相关的数据.

在避免障碍物碰撞方面,变量 od(obstacle distance)记录了机器人与障碍物表面之间的距离,该变量需要超过一定的阈值,以确保机器人的安全.当机器人向目的地前进时,如果其红外传感器检测到障碍物,则机器人需要转向无障碍物方向,直到检测不到再转回目的地轨道.如果机器人为了躲避障碍物而偏离目的地太多,就不能保证最优的移动时间,即尽快移动到目的地.两个响应值 response value 变量[29](*rv_l* 和 *rv_r*)分别代表了障碍物对机器人左前方和右前方的影响.一般来说,障碍物越近或障碍物越危险,响应值越大.在避免翻转方面,内因数据包含了 3 个变量,分别代表 *x*,*y*,*z* 轴的平衡度,测量了其 3 个方向的相对角速度,单位为弧度/秒.相对角速度显示了机器人在不平坦地形上的移动状况,可以通过陀螺仪传感器来测量.此外,两个轮子的速度 velocity 变量(*vel_l* 和 *vel_r*)是影响偏好满意度和目标实现的另外两个内因数据.

机器人系统可操作的功能是通过可控变量 *vel_l* 和 *vel_r* 来分别调整左右两个轮子的速度,使两个轮子以所需的角速度转动,从而让机器人能够尽快到达目的地而不打滑.另外,还可以通过改变两个轮子的速度差来调整方向.例如,让左轮增速和右轮降速,机器人就会向右转.

避免障碍物的外因.在避免障碍物的过程中需要考虑以下 3 种不可控的环境因素.

1) 探测到障碍物时,机器人与障碍物之间的距离(distance).探测时的距离越近,越危险.

2) 障碍物颜色,红外线传感器对物体的颜色很敏感,浅色和红色的障碍物比深色和非红色的障碍物有更高的响应值(response value).因此,红色的障碍物被认为是更危险的.

3) 障碍物纹理,障碍物表明的纹理信息,对红外传感器的探测有不同程度的反射或吸收,从而影响定位障

碍物的时间.

此外,我们还考虑了一个未知环境因素——障碍物表面的粗糙度.高度粗糙的材料反射率较低,红外传感器难以发现,而完全光滑的材料则相反(其他因素,包括发射强度,都可能是影响因素.为了实验的简单性,我们忽略了这些因素).这个因素也会影响障碍物被发现时的距离和响应值.在实验中,我们假设粗糙度的变化代表了新的部署环境.

避免翻转的外因.影响机器人翻转的环境因素包括:(1) 坡的倾斜度;(2) 与斜坡的距离,这可通过摄像头感知激光线发生器投射的线而得出;(3) 机器人的负载,通过惯性对平衡度产生影响,负载越大,惯性越大.

物体与物体之间接触时,摩擦力无处不在,但在实际的不平整路面上,摩擦力的影响却不容易计算.在无摩擦力的接触面上部署机器人,与永不打滑的接触面上部署,机器人的运动轨迹和方式肯定有很大差异.在实验中,我们假设摩擦系数是未知环境因素,修改摩擦力代表了新的部署环境.

自适应规则.DAP 规定了内因如何触发调整左右轮速度的自适应动作(即 *setleftmotor*(*x*1)和 *setrightmotor*(*x*2)),两侧的响应值 *rv_l* 和 *rv_r*、机器人当前速度 *vel_l* 和 *vel_r* 以及障碍物距离 *od* 都会影响遇到障碍物时两轮的速度差.一般来说,当来自障碍物的响应值很大、机器人当前的速度很快、安全距离较小时,两轮之间的速度差会变大,以实现更大程度的转弯.此外,3 个维度的平衡度 *bd_x*、*bd_y*、*bd_z* 代表了斜坡环境下的潜在移动速度,直观来说,当 *db_x* 为负值且较小时,机器人需要在这个维度上减速以防止后翻.同时,针对特定部署环境的 IAP 表明了上述所有环境因素如何影响内部数据,从而影响到系统的目标和偏好.

4.2.1 实验设置

在下面的实验中,我们用 Cyberbotics 公司开发的最著名的模拟器之一 Webots 作为仿真平台,Webots 是一款用于移动机器人快速原型设计和仿真的商业软件[30],它让我们可以定义和修改一个完整的 Pioneer 3-DX 机器人,并可以为机器人配备大量的可用传感器和执行器,如红外传感器、GPS 和指南针等.通过调用机器人库中的函数 *wb_motor_set_velocity*,可以方便地操纵调整机器人速度的自适应行为.对于物体(即跑道上的障碍物和斜坡),我们可以定义一些属性,如形状、颜色、纹理、坡度、摩擦力等.有兴趣的读者可以参考 Webots 文档.

在上述概念模型中,机器人能够在同时具有障碍物和斜坡的环境中运行.但是,为了简化实验结果并增强易读性,我们分别考虑躲避障碍物和避免翻转两个场景:(1) 一条跑道上有 3 个不同颜色、大小和纹理的障碍物,如图 8 所示;(2) 一条跑道上有两个不同坡度的上坡和下坡场景,如图 9 所示.在下面的实验中,我们将左右轮子的最大速度设定为 12rad/s.

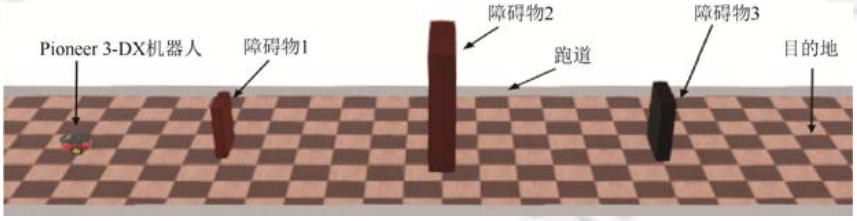

Fig.8 Runway for obstacle avoidance scenario

图 8 跑道 1——躲避障碍物场景

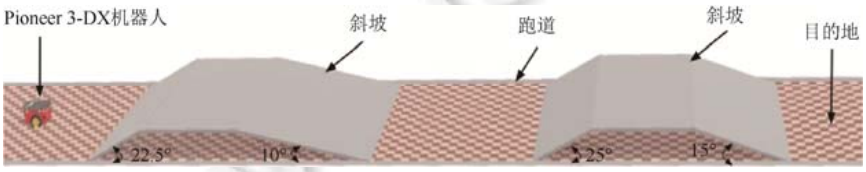

Fig.9 Runway for turnover avoidance scenario

图 9 跑道 2——避免翻转场景

同样地,我们希望评估基于内外两层框架设计的适用性和内层规则的可复用性.在适用性实验中,我们在机器人领域专家的帮助下制定了 DAP,然后不断地修改环境变量并进行模拟,以找到环境因素和内部变量之间的

关系(即 IAP).通过参考概念模型中的这些自适应规则形式,所构建的系统对部署环境中的环境因素的变化应当具有相当的自适应能力.在可复用性实验上,对于躲避障碍物场景,我们假设障碍物表面的粗糙度是一个未知因素.若粗糙度为 0,则障碍物的材质完全光滑,而若粗糙度为 1,则障碍物的表面高度粗糙,安全距离的阈值会随着粗糙度的增加而增加.然后,我们定义了不同颜色的安全距离阈值,当粗糙度为 0 时,黑色障碍物为 0.1,红色为 0.2,而当粗糙度为 1 时,黑色障碍物为 0.2,红色为 0.3.当 3 个障碍物的粗糙度都从 0 变为 1 时,代表了部署环境的变化.对于避免翻转的场景,我们将摩擦力设定为未知的环境因素,在 Webots 中摩擦力的设置空间为 0 到无穷大,0 表示一个完全没有摩擦力的接触跑道,而无穷大则意味着一个永不滑动的接触跑道,摩擦力的改变代表了部署环境的变化.在避免翻转的可复用实验中,我们将进行 3 组实验——跑道与机器人之间的摩擦值分别为 0.6、1、2.5.当部署环境发生改变时,我们会重新模拟并更新环境因素与内因数据之间的关系(即 IAP).

4.2.2 实验结果

躲避障碍物场景——适用性实验.基于归因实现的实验结果如图 10 所示,记录了两个轮子的速度和机器人与障碍物之间的距离(其中,上图给出左右两轮速度,下图给出与障碍物的距离).

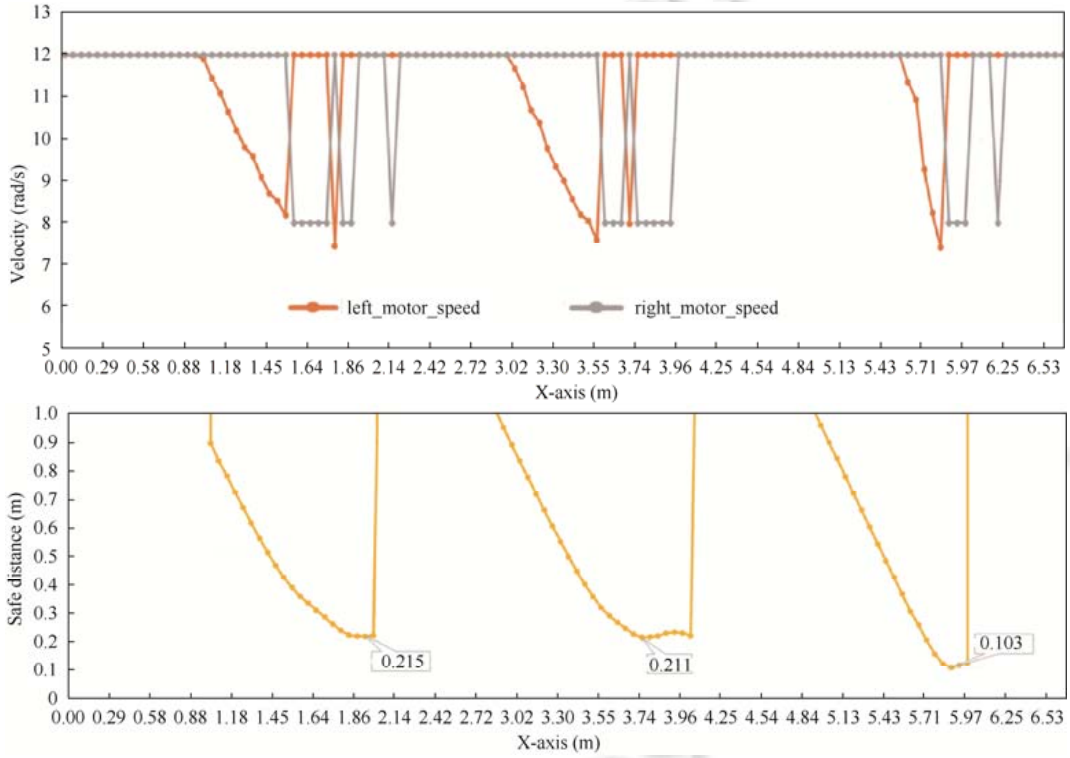

Fig.10 Obstacle avoidance scenario: Experiment results for attribution-based implementation

图 10 躲避障碍物场景:基于内外因两层实现的实验结果

我们可以看到,当机器人移动到坐标 X 轴约为 1 的位置时,第 1 个障碍物被检测到.然后,左轮开始减速,机器人稍微左转,此时障碍物的影响(即响应值)很小.随着影响的增大,左轮的转速也进一步降低,机器人左转的幅度增大,直到机器人右前方的影响降低为 0.随后,机器人为左轮增速,并适当地为右轮减速,机器人向右转,向目的地前进,这个调整可以防止过度避让和偏离目的地太多.避障和目标定位相互作用,直到第 1 次避障完成,然后,机器人陆续通过第 2 个和第 3 个障碍物.第 2 次避障过程与第 1 次避障过程相似,因为两个障碍物的颜色相同.对于第 3 个黑色障碍物来说,避障过程有所不同,因为只有当黑色物体非常接近时,控制器才会采取自适应行为.这是因为,在我们的场景中,红色物体比黑色物体更危险,具体表现为红外传感器对红色物体更敏感,有更高的

响应值.距离障碍物的值也如图 10 所示,前两个避障过程中最近的距离非常接近 0.2 的阈值,而第 3 个最近的距离大约为 0.1.

图 11 显示了不是基于归因方式实现的一般障碍物规避算法的实验结果(其中,上图给出左右两轮速度,下图给出与障碍物的距离),其中,适应策略直接受到前文所述中 3 个环境因素的影响.该算法显示了机器人遵循路径、检测障碍物,并绕过障碍物以避免碰撞的能力.机器人由于监测到环境中的事件(即检测到障碍物),根据算法向左转.这些预先定义的规则可以确保与障碍物的安全距离在阈值之上,且仍然在一个合理的范围内.总体来说,无论是基于归因的实现还是一般的避障算法框架,从出发点到目的地的执行时间相当,约 135 个时间戳,且系统的满意度很高,与障碍物的最短距离略高于但仍在阈值附近,且都能满足系统目标.因此,这两种方式的实现都能很好地展示出系统避障的自适应能力.

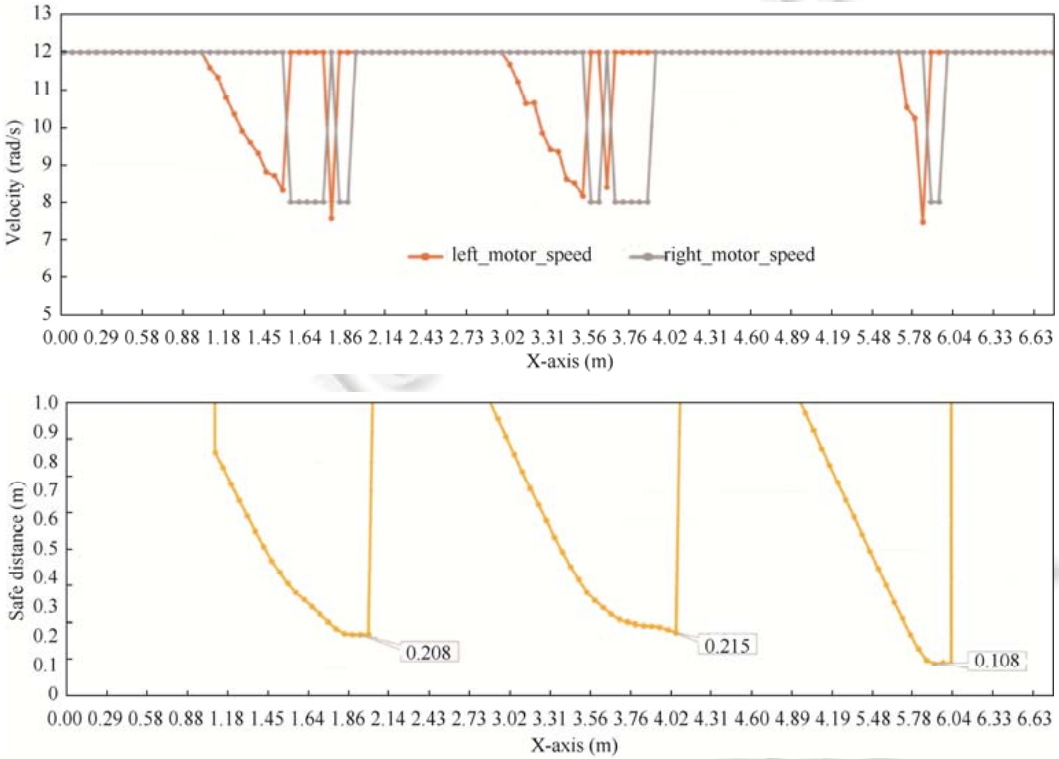

Fig.11 Obstacle avoidance scenario: Experiment results for non-attribution-based implementation

图 11 躲避障碍物场景:非内外因自适应方法实现的实验结果

躲避障碍物场景——可复用性实验.本实验与适用性实验类似,除了每个障碍物表面的粗糙度更大之外,其他设置相同.我们重新模拟并找出环境因素与内因变量之间的关系(即 IAP).如图 12 的实验结果所示(其中,上图给出了左轮速度 left_motor_speed 与右轮速度 right_motor_speed,下图给出了与障碍物的距离),在保证安全距离高于阈值的前提下,左轮的速度会随着粗糙障碍物更大的响应值而大幅度地下降,从而使得机器人能够大幅度地左转避开更危险的粗糙障碍物.对于普通的避障算法,由于预定义的适应方式并没有考虑粗糙度因素,因此在这种新的部署环境下,速度、距离结果和执行时间将与之前的部署相同(如图 11 所示).然而,在普通避障算法的结果中,其安全距离远低于需求中的阈值(机器人与第 3 个障碍物之间的最小距离约为 0.108,低于设定的阈值 0.2),偏好满意度不能维持在令人满意的水平.在基于内外因的实现上,我们在这个新的部署环境上通过模拟获得了 IAP,并重复使用了原有的 DAP.系统的目标被很好地实现,与障碍物的安全距离也能很好地得到满足,但需要 170 个时间戳,这是因为,为了与障碍物之间满足更高的安全距离,机器人不得不大幅度转向,因此也会更大程

度地偏离目的地,也就需要更多的时间到达目的地.

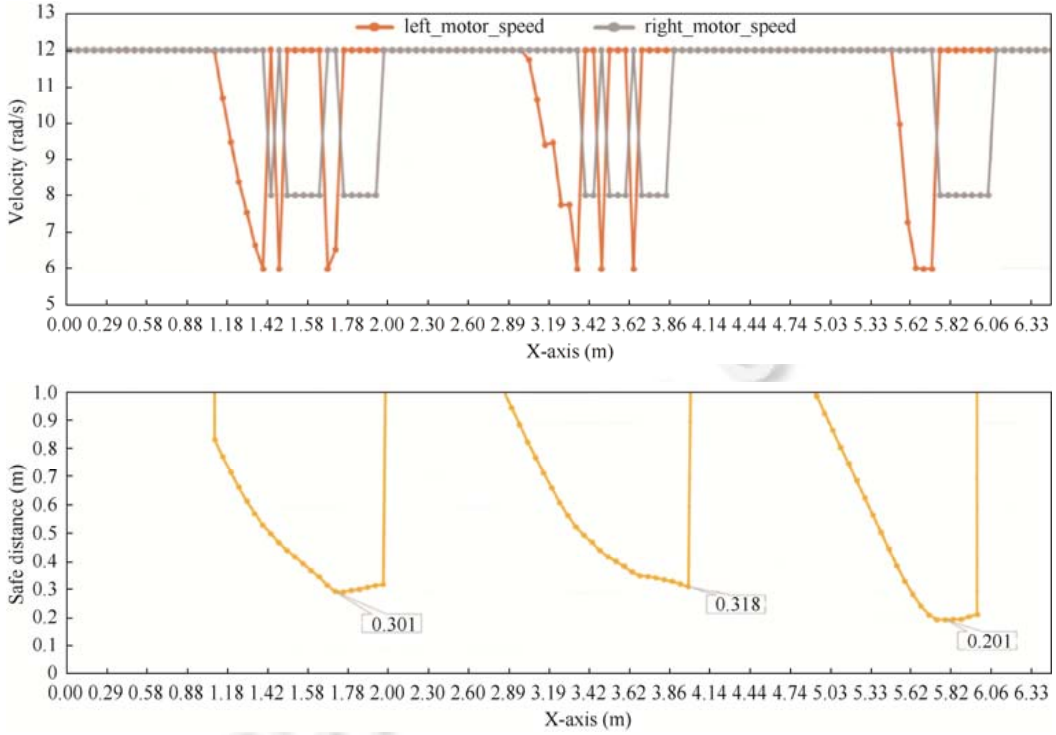

Fig.12 Experiment results for attribution-based implementation with the changing deployment environment

图 12 躲避障碍物场景:基于内外因两层实现的实验结果——变化的部署环境

避免滑倒和翻转的场景——可复用性实验.如图 13 所示,有 3 组实验(其中,上左图的摩擦力=1,上右图的摩擦力=0.6,下图的摩擦力=2.5).

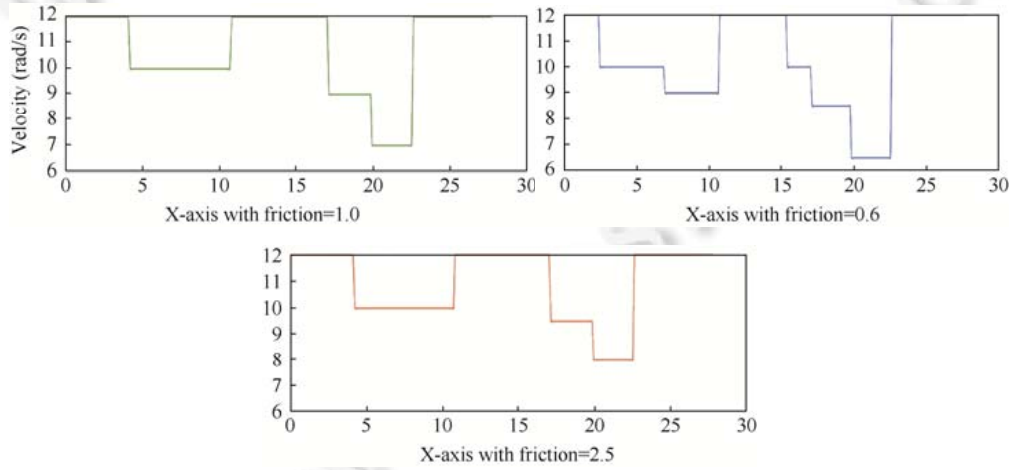

Fig.13 The successful situations on different friction values

图 13 基于内外因两层实现的实验结果

根据不同的部署环境(即不同的摩擦力),将环境因素和内部变量之间的影响关系通过 IAP 代入,可以在较

高的水平上实现目标且满足偏好.在摩擦力为 1 的情况下,机器人以 12rad/s 的最大速度向目的地移动,并以这个速度爬上第 1 个上坡.由于这种情况下不需要左转或右转,所以左右轮子的速度总是相同的.当机器人检测到第 1 个下坡和坡度时,通过对 IAP 和 DAP 的推理,将轮子的速度设为 10rad/s 从而避免下坡翻转,到平地时再调整回最大速度.为了简化这些实验,我们只在 0.5 的间隔点之间找寻速度的最佳值.在第 2 次下坡时,考虑到坡度较为陡峭,如果直接在下坡时放慢速度,机器人很容易翻车,因此机器人需要提前降低速度,以减少冲力.所以,机器人在第 2 个斜坡的平坦部分先将速度降低到 9,再降低到 7.

在第 2 组实验中,部署环境的接触面相对光滑,摩擦力为 0.6,控制器在机器人爬上坡时也需要减速.这一现象与我们的直觉一致,即机器人的速度越小,爬坡的牵引力越大.除此之外,与第 1 个实验相比,在摩擦力较小的情况下,减速的规模更加明显.当摩擦力增加到 2.5 时,如第 3 组实验所示,实验结果是相反的,即只需要减速到 8.

如果我们将预定制的规则(非内外因)应用到不同的部署环境中,不仅偏好可能不令人满意,目标也可能无法实现.图 14 所示实验显示了两个典型的例子(其中,左图的摩擦力=0.6,右图的摩擦力=2.5),在预定制策略时,我们将部署环境中的摩擦力设为 1,且没有把摩擦力作为一个明确的环境因素考虑进规则中,然后将这一定制好的自适应策略应用到不同摩擦力的跑道上.可以看到,当摩擦力为 0.6 时,相对于摩擦力为 1 时更光滑,机器人在牵引力不足的情况下无法爬升(停留在坐标约为 16 的位置).在摩擦力为 2.5 的部署环境中,机器人下坡所需要的速度比在摩擦力较小的环境下预设的速度要高,机器人会出现翻车的情况(约在坐标 21 的位置).在这两种情况下,机器人都无法到达目的地.

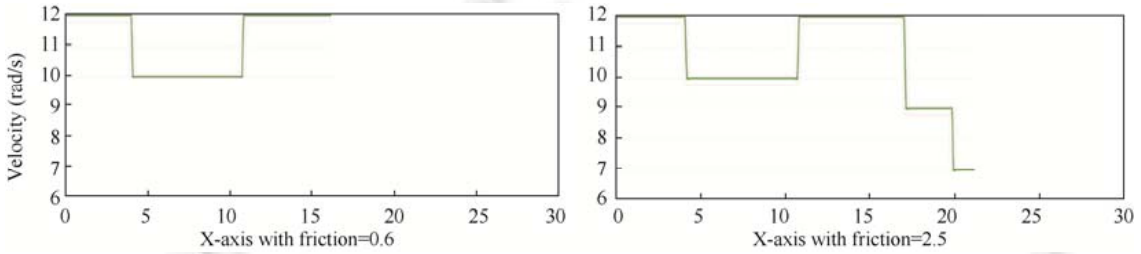

Fig.14 The failure situations on different friction values

图 14 非基于内外因两层实现的实验结果

4.3 讨 论

基于实验结果,将系统特定知识 DAP 和环境特定知识 IAP 分离的实现适用于自适应网络应用和机器人系统,两个案例中系统的目标都在内外因两层框架下得到很好的满足,且表现出了与他人工作中可匹敌的自适应能力.同时,只有 DAP 这个相对于复杂环境而言核心且相对简单的部分在设计时需要被考虑,与环境相关的部分 IAP,可以延迟到部署阶段,这有助于设计者在设计时专注于系统的核心部分,而将与部署环境相关的细节过滤掉.此外,在变化的部署环境中表现出的自适应结果中,显示出了内层 DAP 的可复用性.在 RUBiS 案例中,当在自适应规则中额外增加环境因素变量时,还可以进一步提升系统性能,因此我们的方法还适用于可扩展的、复杂且开放的部署环境,且只需更替 IAP 而复用 DAP.

局限性.基于内外因方法实现的一个基本假设是,内因,特别是那些与实现自适应目标相关的变量,应该被准确地识别并与外因区分开来,这不是一项简单的任务.在基于以上两个案例的探究中,我们可以归纳出一个识别内外因的流程,但是否构成指导准则还需要进一步探究.流程的第 1 步是先找出系统的自适应行为,然后尽可能地识别出所有的因素.第 2 步需要探索哪些因素是属于系统本身的(无论部署环境是什么它们都必定存在);哪些因素是属于环境的,这些因素独立于系统存在.所谓内因,是系统所固有的东西,系统的行为可以完全由内因决定,所以在分析影响系统行为的因素时,那些属于系统本身的因素可以被划分为内因,当然,这很可能需要领域专家的进一步校对和帮忙.此外,我们并不声称内因外因的区分能在所有现实复杂系统中实现——这超出了本文的范畴,相反地,本文仅试图提供一种新的方法来设计自适应系统,通过适应逻辑的分层,以适应多种不

同的部署环境.

部署环境的改变或者扩展,都可能伴随着新一轮的环境因素与内因关系(即 IAP)的学习,我们认为,这个成本与传统方法中自适应系统的整体规则更换或者更新相比,相对较低(尚未验证),我们将在未来工作中对由于部署环境的更换所带来的 IAP 或者整体规则的学习或模拟成本进行测算和比较.基于内外因的方法其优点还在于:能够让系统设计者着重于系统相对核心部分的设计,简化了设计者在早期阶段的工作,这也是软件工程方法论原则——关注点分离的又一次应用;通过溯源自适应行为的内在原因,更容易帮助设计者发掘出影响内因的环境因素(即外因).另外,由于环境中存在各种具有非确定性的环境因素以及系统内部有各种不同的变量,自适应逻辑,特别是关于环境的自适应规则,不可避免地就会非常复杂,且环境因素对不同内因变量的影响也是不同的,所以,获取这些规则和关系不是一件容易的事,这也是未来工作的一个重要方向.

5 相关工作

在本文中,我们基于归因理论,新提出了一个设计自适应系统的概念模型,在设计时通过将系统和环境解耦,让自适应系统不仅能够适应具有非确定性的某个部署环境,还能够很好地适应不同的部署环境.我们将相关工作分为 3 类.首先,探讨自适应系统的建模和决策以定位本文的工作;其次,自适应系统中的可复用性也是本文的一个关注点;最后,讨论不同学科与自适应系统之间的交叉方法,这些方法也为推动自适应系统的发展起到了至关重要的作用.

自适应系统的建模分为 4 个维度[31]:目标,系统需要达到的目标;变化,自适应的原因;机制,系统应该如何应对变化;效果,自适应对系统的影响.本文关注的是变化和机制两个维度.在 MORPH 框架中,自适应行为被划分为两个独立的关注点:一个是配置自适应,研究如何改变构件的结构和操作参数;另一个是行为自适应,研究如何引导系统的行为.MORPH 通过明确区分这两类不同的自适应且协调二者,从而使得系统能够适应复杂的自适应场景[32].本文不对自适应行为作进一步划分,而是将引发自适应行为的原因划分为两个独立的关注点——内因和外因,且认为内因才是引发自适应行为的根本原因.此外,在多智能体设计中,常用 BDI(即信念 belief、愿望 desire 和意图 intention)模型来刻画智能体的结构,信念代表一个智能体关于自身、外部世界和其他智能体的信息状态,愿望或者目标是它的动机状态,而意图是实现当前意愿的动作序列[33].本文的定义与 BDI 有相似之处,例如用数据状态来刻画关于自身的信息状态,目标和偏好反映系统需要达到和最好达到的数据状态.不同之处在于,我们将外部信息状态用外因表示,外因只有通过影响内因,才能影响系统需要执行的动作序列,即自适应行为.

自适应系统中的决策方法可以分为基于规则的方法、基于目标的方法、基于效用的方法以及基于控制论的方法.在基于规则的方法中,适应规则规定了当某一环境变化发生时应进行的适应性动作[34,35],其形式有以下几种:CA(条件和动作)规则[36]、ECA 规则(事件、条件和动作)[37]、状态转换图[11].在基于目标的方法中会指定一个目标状态,规划的目的是弄清楚从当前状态到目标状态的一系列操作[38,39].基于效用的方法通过效用函数将系统模型中的每个状态映射成数值,并选择使效用值最大的状态.基于控制论方法的系统由两部分组成,一个是需要被适应的目标系统,另一个是实现特定控制算法或策略从而让目标系统去适应的控制器,设定值是适应目标的期望值,控制器通过系统输出和设定点之间的差别来调整目标系统,这类方法为系统的数学分析和设计提供了理论和工具[19].本文主要探究基于规则的自适应系统,采用 CA 形式,将自适应规则分为特定系统部分和特定环境部分,分别代表了内部归因和外部归因.内外归因能否运用于基于目标或者控制论的方法还有待进一步探究.

可复用性一直是自适应系统领域的关注点.一般来说,研究主要集中在如何为自适应提供框架,如 Rainbow[34]通过架构层的抽象模型来监控系统层的执行情况,该模型通过翻译层与系统层进行交互;Hogna 是一个在云上部署自管理 Web 应用的平台[40],允许开发人员定制反馈循环的每个阶段,而不必自己实现整个层.可以复用的不同模式也促进了动态自适应系统的开发[11].Weyns 等人提出了一个全面的参考模型——FORMS,潜在地支持对常见问题的可重用的架构解决方案或架构模式的文档[41,42].自适应软件产品线(autonomic

software product line,简称 ASPL)是将一个独立于领域的管理系统集成到特定领域的软件产品线中,开发具有系统复用性的自适应软件系统[43].其他技术,如双向变换、同步化机制,已被应用于确保自适应系统中可复用的正确性[13].虽然我们的方法与大多数可复用的方法类似,将系统框架分为多个层次,但我们不是基于相似功能或结构的划分以复用,我们的划分是基于自适应行为的归因(归因于环境或系统本身),以便于内归因可以在不同的部署环境中复用.

自适应系统是一个跨学科的研究领域.自适应的概念来源于生物学,是指生物为了适应新的环境而改变其行为[44].计算机科学中的生物学方法是随着 Parunak[45]对自然多代理系统中集体行为的研究而出现的,生物学中的其他机制,如蜂群、筑巢、成型[46]、人体免疫系统[47]等在自组织系统中已被采用,并可移植到自适应系统中.除此之外,学习和借鉴其他领域的知识很重要,这些知识正在用于或已经用于开发和研究相似的系统,或者已经对自适应系统贡献了合适的解决方案:对于设计自适应系统,由自然界启发的策略已经受到广泛的关注[48];化学方面的研究也已逐渐被应用,例如 Viroli 等人[49]提出了基于生化元组空间和化学反应的自组织系统协调模型;在物理领域,Weyns 等人采用基于场的机制在多智能体系统中进行自适应任务分配;社会领域则集中研究了市场和拍卖机制,例如在多智能体系统中的协调就是基于社会约定的[50].跨学科角度可以帮助我们构建新颖的自适应系统模型和架构.本文的方法受认知科学研究成果的启发,而认知科学在自适应系统这一领域中从未被探讨过.我们的框架强调自适应行为的原因,来自于外部环境和内部系统两个方面,将系统与特定环境解耦,从而为自适应系统的构建带来了一个全新的视角.

6 结 论

系统自适应已经变得越来越重要,虽然在这一领域已经投入了大量优秀的研究工作,但自适应作为一个领域仍处于起步阶段,现有的知识和方法不足以应对当今不断扩展和复杂的环境.本文主要研究基于规则的自适应系统,并在归因理论的启发下,提供一个概念模型和实现框架.本文将自适应逻辑按照内归因和外归因分为两层,分别对应固定的独立设计和可改变的部署绑定,这种新的概念模型可以帮助自适应系统适应不同的部署环境.

在今后的研究中,我们计划将该方法应用到更多的实际场景中,以增强其适用性.由于环境复杂,环境因素与内因之间的映射关系也复杂多变,研究如何获取影响性适应规则会是一个潜在的方向.此外,采用机器学习中的强化学习方法,以应对开放的部署环境,也是未来值得探究的关注点.

References:

[1] de Lemos R, Giese H, Etc HAM. Software engineering for self-adaptive systems: A second research roadmap. In: Proc. of the Software Engineering for Self-adaptive Systems II, 2010 Revised Selected and Invited Papers. Dagstuhl Castle, 2010. 1–32.

[2] Cheng BHC, Lemos R de, Giese H, Inverardi P, Etc JM. Software engineering for self-adaptive systems: A research roadmap. In: Proc. of the Software Engineering for Self-adaptive Systems. 2009. 1–26.

[3] Sawyer P, Bencomo N, Whittle J, Letier E, Finkelstein A. Requirements-aware systems: A research agenda for re for self-adaptive systems. In: Proc. of the 18th IEEE Int'l Requirements Engineering Conf. Sydney, 2010. 95–103.

[4] Salehie M, Tahvildari L. Self-adaptive software: Landscape and research challenges. ACM Trans. on Autonomous and Adaptive Systems, 2009,4(2):1–42.

[5] Shevtsov S, Berekmeri M, Weyns D, Maggio M. Control-theoretical software adaptation: A systematic literature review. IEEE Trans. on Software Engineering, 2018,44(8):784–810.

[6] Modoni GE, Trombetta A, Veniero M, Sacco M, Mourtzis D. An event-driven integrative framework enabling information notification among manufacturing resources. Int'l Journal of Computer Integrated Manufacturingint, 2019,32(3):241–252.

[7] Li N, Bai D, Peng Y, Yang Z, Jiao W. Verifying stochastic behaviors of decentralized self-adaptive systems: A formal modeling and simulation based approach. In: Proc. of the 2018 IEEE Int'l Conf. on Software Quality, Reliability and Security (QRS). Lisbon: IEEE, 2018. 67–74.

[8] Aldrich J, Garlan D, Kästner C, Goues CL, Mohseni-Kabir A, Ruchkin I, Samuel S, Schmerl BR, Timperley CS, Veloso M, Voysey I, Biswas J, Guha A, Holtz J, Cámara J, Jamshidi P. Model-based adaptation for robotics software. IEEE Software, 2019,36(2): 83–90.

[9] Dijkstra EW. On the Role of Scientific Thought. Selected Writings on Computing: A Personal Perspective. Springer-Verlag, 1982. 60–66.

[10] Keeney J, Cahill V. Chisel: A policy-driven, context-aware, dynamic adaptation framework. In: Proc. of the 4th IEEE Int'l Workshop on Policies for Distributed Systems and Networks (POLICY 2003). Lake Como, 2003. 3–14.

[11] Ramirez AJ, Cheng BHC. Design patterns for developing dynamically adaptive systems. In: Proc. of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-managing Systems, SEAMS 2010. Cape Town, 2010. 49–58.

[12] Bozhinoski D, Garlan D, Malavolta I, Pelliccione P. Managing safety and mission completion via collective run-time adaptation. Journal of Systems Architecture—Embedded Systems Design, 2019,95:19–35.

[13] Colson K, Dupuis R, Montrieux L, Hu Z, Uchitel S, Schobbens P-Y. Reusable self-adaptation through bidirectional programming. In: Proc. of the 11th Int'l Workshop on Software Engineering for Adaptive and Self-Managing Systems—SEAMS 2016. Texas: ACM Press, 2016. 4–15.

[14] Yang QL, Ma XX, Xing JC, Hu H, Wang P, Han DS. Software self-adaptation: Control theory based approach. Chinese Journal of Computers, 2016,39(11):2189–2215 (in Chinese with English abstract).

[15] Kephart JO, Chess DM. The vision of autonomic computing. IEEE Computer, 2003,36(1):41–50.

[16] Heider F. The Psychology of Interpersonal Relations. New York: Wiley, 1958.

[17] Kelley HH. Attribution theory in social psychology. In: Proc. of the Nebraska Symp. on Motivation. Lincoln: University of Nebraska Press, 1967,15:192–238.

[18] Krupitzer C, Roth FM, VanSyckel S, Schiele G, Becker C. A survey on engineering approaches for self-adaptive systems. Pervasive and Mobile Computing, 2015,17:184–206.

[19] Jiao W, Sun Y. Self-adaptation of multi-agent systems in dynamic environments based on experience exchanges. Journal of Systems and Software, 2016,122:165–179.

[20] Chen Y, Iyer S, Liu X, Milojicic DS, Sahai A. SLA decomposition: translating service level objectives to system level thresholds. In: Proc. of the 4th Int'l Conf. on Autonomic Computing (ICAC 2007). Jacksonville, 2007. 3.

[21] Longueville B, Gardoni M, Others. A survey of context modeling: Approaches, theories and use for engineering design researches. In: Proc. of the ICED 2003, the 14th Int'l Conf. on Engineering Design. Stockholm, 2003. 437–438.

[22] Filieri A, Maggio M, Angelopoulos K, Etc ND. Control strategies for self-adaptive software systems. TAAS, 2017,11(4): 24:1–24:31.

[23] Shen Z, Subbiah S, Gu X, Wilkes J. CloudScale: Elastic resource scaling for multi-tenant cloud systems. In: Proc. of the ACM Symp. on Cloud Computing in Conjunction with SOSP 2011, SOCC '11. Cascais, 2011. 5.

[24] Zhao T, Zhang W, Zhao H, Jin Z. A reinforcement learning-based framework for the generation and evolution of adaptation rules. In: Proc. of the 2017 IEEE Int'l Conf. on Autonomic Computing, ICAC 2017. Columbus, 2017. 103–112.

[25] Amza C, Chanda A, Cox AL, Elnikety S, Gil R, Rajamani K, Zwaenepoel W, Cecchet E. Specification and implementation of dynamic Web site benchmarks. In: Proc. of the IEEE Int'l Workshop on Workload Characterization, WWC-5. 2002.

[26] Klein C, Maggio M, Karl-Erik A, Hernández-Rodriguez F. Brownout: Building more robust cloud applications. In: Proc. of the 36th Int'l Conf. on Software Engineering, ICSE 2014. Hyderabad, 2014. 700–711.

[27] Almasri MM, Alajlan AM, Elleithy KM. Trajectory planning and collision avoidance algorithm for mobile robotics system. IEEE Sensors Journal, 2016,16(12):5021–5028.

[28] Lee JH, Park JB, Lee BH. Turnover prevention of a mobile robot on uneven terrain using the concept of stability space. Robotica, 2009,27(5):641–652.

[29] Webots. Webots reference manual.

[30] Michel O. Webots: Professional mobile robot simulation. Journal of Advanced Robot System, 2004,1:39–42.

[31] Andersson J, de Lemos R, Malek S, Weyns D. Modeling dimensions of self-adaptive software systems. In: Proc. of the Software Engineering for Self-Adaptive Systems. 2009. 27–47.

[32] Braberman VA, D'Ippolito N, Kramer J, Sykes D, Uchitel S. An extended description of MORPH: A reference architecture for configuration and behaviour self-adaptation. In: Proc. of the Software Engineering for Self-Adaptive Systems. 2013. 377–408.

[33] Anand S. Rao, Michael P. Georgeff. {BDI} agents: From theory to practice. In: Proc. of the 1st Int'l Conf. on Multiagent Systems San Francisco, 1995. 312–319.

[34] Garlan D, Cheng S-W, Huang A-C, Schmerl BR, Steenkiste P. Rainbow: Architecture-based self-adaptation with reusable infrastructure. IEEE Computer, 2004,37(10):46–54.

[35] Garlan D, Schmerl BR, Cheng S-W. Software architecture-based self-adaptation. In: Proc. of the Autonomic Computing and Networking. 2009. 31–55.

[36] Fernandes P, Werner C, Murta LGP. Feature modeling for context-aware software product lines. In: Proc. of the 20th Int'l Conf. on Software Engineering & Knowledge Engineering (SEKE 2008), San Francisco, 2008. 758–763.

[37] Cheng BHC, Sawyer P, Bencomo N, Whittle J. A goal-based modeling approach to develop requirements of an adaptive system with environmental uncertainty. In: Proc. of the 12th Int'l Conf. on Model Driven Engineering Languages and Systems, MODELS 2009. Denver, 2009. 468–483.

[38] Qian W, Peng X, Chen B, Mylopoulos J, Wang H, Zhao W. Rationalism with a dose of empiricism: Case-based reasoning for requirements-driven self-adaptation. In: Proc. of the 22nd IEEE Int'l Requirements Engineering Conf. Karlskrona, 2014. 113–122.

[39] Tsigkanos C, Pasquale L, Ghezzi C, Nuseibeh B. On the interplay between cyber and physical spaces for adaptive security. IEEE Trans. on Dependable and Secure Computing, 2018,15(3):466–480.

[40] Barna C, Ghanbari H, Litoiu M, Shtern M. Hogna: A platform for self-adaptive applications in cloud environments. In: Proc. of the 10th IEEE/ACM Int'l Symp. on Software Engineering for Adaptive and Self-managing Systems, SEAMS 2015. Florence, 2015. 83–87.

[41] Weyns D, Malek S, Andersson J. FORMS: Unifying reference model for formal specification of distributed self-adaptive systems. TAAS, 2012,7(1):8:1–8:61.

[42] Weyns D. Software engineering of self-adaptive systems. In: Handbook of Software Engineering. 2019. 399–443.

[43] Abbas N. Designing Self-adaptive Software Systems with Reuse. Linnaeus University Press, 2018.

[44] Longman A-W. Adaptive Control. Boston: Publishing Co., Inc., 1994.

[45] Parunak HVD. 《Go to the ant》: Engineering principles from natural multi-agent systems. Annals OR, 1997,75:69–101.

[46] Mamei M, Menezes R, Tolksdorf R, Zambonelli F. Case studies for self-organization in computer science. Journal of Systems Architecture, 2006,52(8–9):443–460.

[47] Hart E, McEwan C, Timmis J, Hone A. Advances in artificial immune systems. Evolutionary Intelligence, 2011,4(2):67–68.

[48] Macías-Escrivá FD, Haber RE, Toro RM del, Hernández V. Self-adaptive systems: A survey of current approaches, research challenges and applications. Expert Systems With Applications, 2013,40(18):7267–7279.

[49] Viroli M, Casadei M. Biochemical tuple spaces for self-organising coordination. In: Field J, Vasconcelos VT, eds. Coordination Models and Languages. Berlin, Heidelberg: Springer-Verlag, 2009. 143–162.

[50] Salazar N, Rodríguez-Aguilar JA, Arcos JL. Robust coordination in large convention spaces. AI Communications, 2010,23(4): 357–372.

附中文参考文献:

[14] 杨启亮,马晓星,邢建春,胡昊,王平,韩德帅.软件自适应:基于控制理论的方法.计算机学报,2016,39(11):2189–2215.

李念语(1995－),女,博士生,CCF 学生会员,主要研究领域为(人机协同的)自适应软件系统,建模与验证.

刘坤(1996－),男,硕士生,CCF 学生会员,主要研究领域为软件自适应,多智能体学习.

陈正胤(1997－),男,博士生,CCF 学生会员,主要研究领域为自适应软件,前瞻自适应.

焦文品(1969－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为软件工程,智能软件,软件方法学.