DOI：10. 13471/j. cnki. acta. snus. ZR20240030

自主式交通系统架构自适应演进方法*

方明辉[1,2]，由林麟[1,2]，郝迈[1,2]，张稷[3]，梁晨[4]，陈耿祥[1,2]

1. 中山大学智能工程学院，广东 深圳 518107
2. 广东省智能交通系统重点实验室，广东 广州 510006
3. 深圳市城市交通规划设计研究中心股份有限公司，广东 深圳 518057
4. 深圳市市政设计研究院有限公司，广东 深圳 518029

摘　要：基于本体技术和语义逻辑表达方法，以知识图谱的形式建立了自主式交通系统标准化知识库。进一步提出基于演进分析的架构自适应转换方案，通过推断系统在代际间的演进过程，实现具体架构随代际演进的自动转变。最后，以开源自主式交通系统知识图谱为例，评估了提出方法的可行性。相比对照方案，本文方法对高效分析系统演进以及实现架构合理自组织具有较大优势，能满足交通系统向自主化发展的需求。

关键词：自主式交通系统；演进分析；知识图谱；本体；语义网

中图分类号：U11　　**文献标志码：**A　　**文章编号：**2097－0137（2024）04－0115－09

The adaptive evolution mechanism of autonomous transportation system architecture

FANG Minghui[1,2], *YOU Linlin*[1,2], *HAO Mai*[1,2], *ZHANG Ji*[3], *LIANG Chen*[4], *CHEN Gengxiang*[1,2]

1. School of Intelligent Systems Engineering, Sun Yat-sen University, Shenzhen 518107, China
2. Guangdong Key Laboratory of Intelligent Transportation Systems, Guangzhou 510006, China
3. Shenzhen Urban Transportation Planning and Design Research Center Company Limited, Shenzhen 518057, China
4. Shenzhen Municipal Design & Research Institute Company Limited, Shenzhen 518029, China

Abstract: This paper proposes a novel approach that utilizes ontology technology and semantic logic representation, which can construct a standardized knowledge repository in the form of a knowledge graph (KG) for autonomous transportation system (ATS). In addition, an architecture-adaptive transformation schema, to facilitate the automatic evolution of specific architectures across generations, grounded in evolutionary analysis is designed in accordance with inferring the system's evolutionary trajectory. To validate the schema, a fair and comprehensive assessment involving rational, efficient, and effective is achieved based on an open-source KG of ATS. The summarized results can, experimentally, showcase the potential of the proposed approach to meet the evolving requirements of transportation system has got, and continuously get transformed to ATS.

Key words: autonomous transportation system; evolution analysis; knowledge graph; ontology; semantic web

* **收稿日期：**2024－01－20　　**录用日期：**2024－01－31　　**网络首发日期：**2024－04－01
基金项目：国家重点研发计划（2020YFB1600400）；国家自然科学基金（62002398）
作者简介：方明辉（1999年生），男；**研究方向：**自主式交通系统；E-mail：fangmh7@mail2.sysu.edu.cn
通信作者：梁晨（1986年生），男；**研究方向：**自主式交通系统；E-mail：liangch1@szmedi.com.cn

交通运输系统正朝更高程度的自主化发展(关积珍，2022)。现有智能交通系统体系框架，如“中国智能运输体系框架”、美国“ARC-IT”以及欧盟“FRAME”等展现出明显的短板：在描述TS向自主化发展的逐步变化时，缺乏对系统发展阶段即代际的标准化划分(赵娜等,2014;魏伟等,2022)。因而难以处理不同代际间的兼容性和交互性问题，也无法确保系统运行时的一致性和稳定性(裴建中，2018)。此外，现有ITS体系将所有交通行为都交由中央控制的系统架构来管理和调度(严新平等,2021)，无法根据交通场景规模和自主水平的变化调整架构，以支撑面向不同交通场景的个性化服务(You et al.，2022a)，指导实现交通各类组分间的协同化运作(Zhou et al.，2022)。

面对不同的TS自主化发展阶段，亟需一种高效的架构自适应转换方法，规范不同代际系统架构设计，实现架构的跨代际扩展性和兼容性(You et al.，2022b)。针对架构的扩展性，需要解决的首要技术问题是定义标准化的架构描述框架，为架构的转换提供标准化接口。在此基础上，利用不同代际TS组分及其关联作为背景知识，为面向不同代际架构的标准化设计与转换提供依据(唐进君等,2022)。因此，一种高效的架构与TS知识表示方案成为了研究热点。知识图谱在知识库搭建、知识推理等领域的落地经验表明：图结构更善于处理多元复杂数据，以表达机器可推理的结构化语义信息，减少要素多样性、指向性和权重引起的建模限制(王淑营等,2022;张栋豪等,2021;邱凌等,2022)。针对架构的跨代际兼容性，关键是构造合理的跨代际架构转换算法，实现架构自适应转换。例如，党引弟等(2019)基于演进决策规则，依赖反馈循环，分析与调整网络空间安全系统运行架构。李青山等(2021)提出一种基于策略和强化学习的双重自适应机制，对系统感知到的局部和全局变化进行决策，保障复杂环境下飞行器任务的可靠执行。

然而，现有研究忽略了组分演进过程中的语义变化、更新或集成，无法提供一个知识上完备的自适应模型(张明悦等,2020)，以满足不同代际下系统架构应用需求。由于TS知识库是指导架构设计的依据，本文提出基于知识图谱演进分析(Osborne et al.,2018)的ATS架构自适应转换方法。ATS知识图谱中储存了各代际下系统包含的各类实体。通过分析待转换架构中包含的知识，基于ATS的演进关系进行架构重构。

在此过程中，如何分析TS知识的演进，即从图结构的知识中准确识别本体信息的演进细节成为研究的重点。有学者提出基于启发式算法的通用本体知识演进分析方案(Noy et al.,2002)，但存在准确度和语义集成度较低的问题。Hartung et al.(2013)提出了一种生物知识数据库的演进分析方案，但所处理的医学本体结构较为简单，无法有效应对ATS实体间更为错综复杂的关系。Dyn-Diff在Hartung等的基础上，设计了一种支持演进关系类别更多的方案，但存在难以验证和算法复杂度较高的问题(Benavides et al.,2022)。根据Santos et al.(2020)的研究，现有演进分析方案普遍存在演进关系分析不准确、演进关系语义集成度较低的问题，需要更加高效的实体演进分析算法，以支撑场景架构自适应转换。面向ATS架构，本文提出了一种系统架构自适应演进方法，解决了ATS系统发展无演进分析、多代际架构无法相互转换的难题。

1 问题描述

假设ATS知识库定义为集合$K=\{G_1,G_2,\cdots,G_n\}$。其中，G_i是代际i的ATS知识图谱，包括实体集合、关系集合和属性集合。若G_i包含的实体集合表示为$C_i=\{c_1,\cdots,c_p\}$，G_i包含的有向关系三元组集合和属性三元组集合表示为

$$R_i=\{(c_{\text{head}},r,c_{\text{tail}})\},\ A_i=\{(c_{\text{main}},k,v)\},$$

其中r为有向关系名称，k为属性名称，v为属性值，且$\forall\{c_{\text{head}},c_{\text{tail}},c_{\text{main}}\}\in C_i$，则$G_i$为$G_i=(C_i,R_i,A_i)$。定义$E_{ij}$为$G_i$和$G_j$的差异表，储存$G_i$和$G_j$之间的演进关系。架构$O_{si}$被定义为面向场景$s$，在代际$i$下的架构知识图谱，$O_{si}\in G_i$。

基于以上定义，将本文的研究问题定义为：给定ATS知识库、待转换架构O_{si}和目标代际j；利用知识图谱演进分析方法，根据G_i和G_j的差异生成E_{ij}；之后从E_{ij}中提取与O_{si}中实体有关的演进关系，根据演进关系对O_{si}进行重构，最终得到O_{si}。

2　基于知识图谱的 ATS 架构自适应演进方法

如图 1 所示，方法主要包含知识储存、演进关系分析和架构自适应转换模块：知识储存模块以知识图谱形式储存不同代际下的 ATS 知识，实现知识的结构化组织；演进关系分析模块对比相邻代际知识，生成演进表，表达系统随代际有序演进的语义关系。架构自适应转换模块根据架构及需要转换到的代际，自适应提取相关演进关系，重构得到目标代际架构。

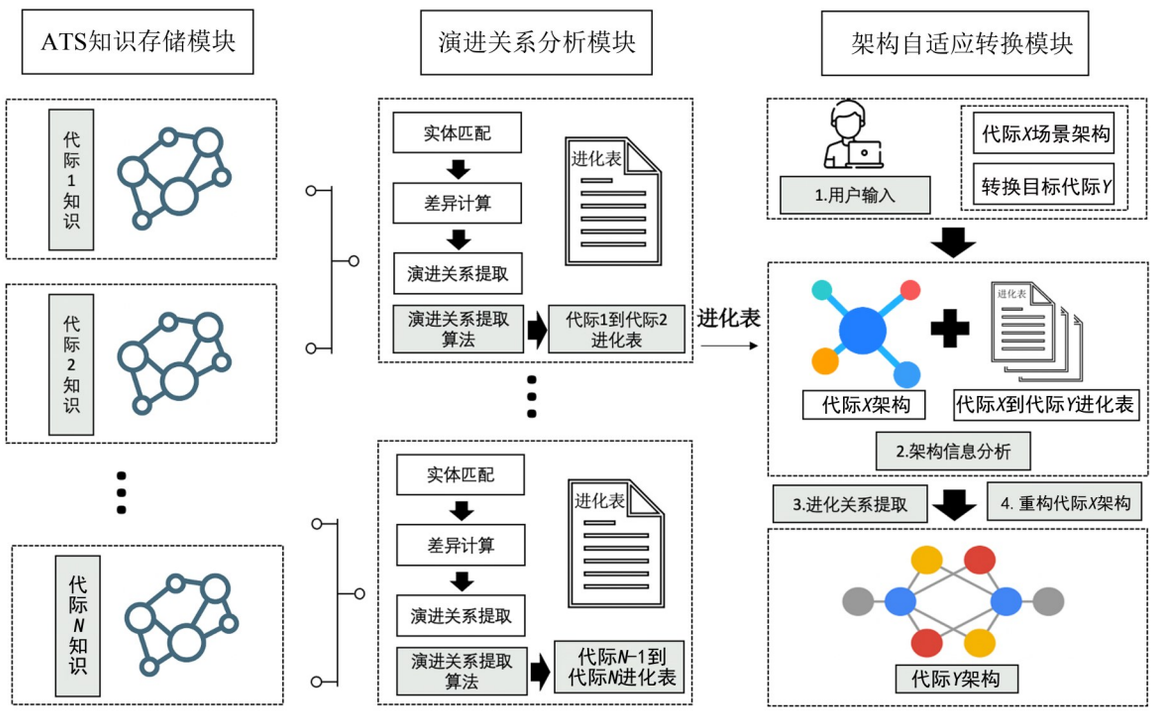

图 1　基于知识图谱的自主式交通系统架构自适应演进方法架构图

Fig. 1　Schematic diagram of adaptive evolution mechanism of ATS architecture based on knowledge graph

2.1　ATS 知识图谱构建

ATS 知识图谱在使用斯坦福法构建领域本体的基础上，自顶向下进行构建（Buchgeher et al.，2021）。以 ATS 驾驶自动化子系统为例，该系统描述了不同级别自动驾驶车辆的软硬件结构、功能实现与信息交互逻辑、条件状态适应性等方面的知识。其中，外部技术和内部需求驱动驾驶自动化系统从“应急辅助”向“完全自动驾驶”的体系化转变，具有明显的代际演进特征。通过对领域术语进行认知和抽象，提炼普适性的约束，构建如图 2 所示的本体模型，对基本概念如系统、硬件、功能及其之间的关联进行本体层面的定义。

在此之上，根据 SAE 定义的驾驶自动化分级标准，采集分析主流汽车工业产品技术标准信息，结合专家咨询，将梳理的领域知识按本体规定转化为基于关系、属性三元组模式的知识图谱。构建 3 个代际开源驾驶自动化知识图谱，其中实体、

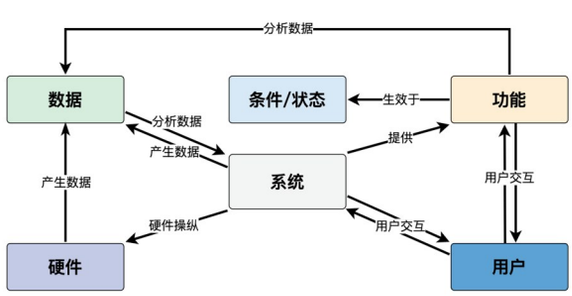

图 2　ATS 驾驶自动化子系统的本体模型

Fig. 2　Ontology model of ATS driving automation subsystem

关系和属性的数量如表 1 所示。其中，实体知识是领域概念的实例；属性知识从属于实体，表达实体的特征；关系知识定义实体之间的关联，是本体约束的实例。系统复杂度随代际的演进快速增长。

2.2　ATS 知识图谱演进分析方案

为梳理 ATS 的演进关系，演进分析模块遍历相邻代际知识图谱，通过实体对齐、差异分析和

表1 ATS驾驶自动化知识图谱各代际包含的知识

Table 1 Knowledge contained in each generation of ATS driving automation knowledge graph

代际	知识类型	知识数量
1级（部分驾驶辅助）	实体	141
	关系	238
	属性	90
2级（组合驾驶辅助）	实体	190
	关系	368
	属性	123
3级（有条件自动驾驶）	实体	187
	关系	378
	属性	127

应用演进规则实现实体演进的分析，生成知识在代际间的演进路径。

2.2.1 实体匹配 为识别ATS中不同代际下表达语义信息相同的实体，提出基于文本相似度的匹配算法，按照粗粒度到细粒度的顺序，进行实体对齐。构建相似性矩阵**SM**，如式(1)所示。

$$\mathbf{SM}=\begin{pmatrix} S_{11} & \cdots & S_{1j} \\ \vdots & \ddots & \vdots \\ S_{i1} & \cdots & S_{ij} \end{pmatrix}, \tag{1}$$

式中S_{ij}为实体i和实体j间的相似度。每一行由原代际实体构成，第一行为原代际粗粒度实体，剩余各行为原代际细粒度实体；每一列由演进代际实体构成，第一列为演进代际粗粒度实体，剩余各列为演进代际细粒度实体。实体间相似度由式（2）确定。

$$S_{ij}=\frac{2M}{T_i+T_j}, \tag{2}$$

式中M为两实体名称包含重复字符串的长度，T_i和T_j代表两个实体名称的字符串长度。

通过对相同或相似的交通实体自动化匹配，可以揭示演进前后系统中存在交叉关联的实体，为演进实体的演进分析提供基础。

2.2.2 差异分析 差异分析旨在发现系统演进过程中发生变化的知识，如技术的引入、迭代或弃用、系统交互关系的更迭。根据知识图谱对中的匹配关系，对于每种差异判定规则，遍历新旧代际实体、关系或属性是否存在增加或删除。如对于i和j两个代际，增加和删除实体c的逻辑判断语句如式(3)和(4)所示。

$$\exists c\notin C_i\cap c\in C_j\rightarrow \mathrm{addEntity}(c), \tag{3}$$

$$\exists c\in C_i\cap c\notin C_j\rightarrow \mathrm{delEntity}(c). \tag{4}$$

差异分析算法支持的差异类别，如表2所示。

表2 本方案可以识别的差异类型

Table 2 Types of evolutions supported by proposed mechanism

差异类型	描述
addEntity(e)	增加实体e
delEntity(e)	删除实体e
modEntity(e_1,e_2)	实体e_1被修改为实体e_2
addRelation(e_1,r,e_2)	增加关系三元组(e_1,r,e_2)
delRelation(e_1,r,e_2)	删除关系三元组(e_1,r,e_2)
modRelation(e_1,r_1,e_2,r_2)	关系(e_1,r_1,e_2)修改为(e_1,r_2,e_2)
addAttribute(e,n,v_1)	实体增加属性三元组(e,n,v_1)
delAttribute(e,n,v_1)	实体删除属性三元组(e,n,v_1)
modAttribute(e,n,v_1,v_2)	实体的属性(e,n,v_1)修改为(e,n,v_2)

其中，修改类差异的判定基于知识的增删以及实体的匹配关系。识别修改实体、关系、属性的规则分别如式(5)-(7)所示：

$$\exists c_1\in C_i\bigcap\exists c_2\in C_j\cap\mathrm{delEntity}(c_1)\cap\mathrm{addEntity}(c_2)\cap\mathrm{match}(c_1,c_2)\rightarrow\mathrm{modEntity}(c_1,c_2), \tag{5}$$

$$\begin{aligned}&\exists c_1,c_2\in C_i\cap\exists c_3,c_4\in C_j\cap\mathrm{match}(c_1,c_3)\\&\cap\mathrm{match}(c_2,c_4)\cap\mathrm{delRelation}(c_1,r_1,c_2)\\&\cap\mathrm{addRelation}(c_3,r_2,c_4)\\&\rightarrow\mathrm{modRelation}(c_3,r_1,c_4,r_2),\end{aligned} \tag{6}$$

$$\begin{aligned}&\exists c_1\in C_i\cap\exists c_2\in C_j\cap\mathrm{match}(c_1,c_2)\\&\cap\mathrm{delAttribute}(c_1,n,v_1)\cap\mathrm{addAttribute}(c_2,n,v_2)\\&\rightarrow\mathrm{modAttribute}(c_2,n,v_1,v_2).\end{aligned} \tag{7}$$

值得注意的是：1）每个差异都有其逆差异，以确保差异分析的可逆性，即可以从代际V_1到V_2的变化逆推代际V_2到V_1的变化。2）基于特性1，通过整合V_1到V_2以及V_2到V_3的差异集，可以形成代际V_1经V_2演进到V_3的差异集，反之亦然。这些特性支撑TS演进的可逆化分析。

2.2.3 演进关系分析 为减少架构演进的计算成本，本文提出一种基于规则的算法，生成高集成度的演进关系，以支持差异的语义化分析，如系统功能的细化、融合和系统功能子集的增删。算法共支持10种演进分析类型，如表3所示。分析算法如算法1所示。

表3　支持的演进关系类型

Table 3　Types of evolutions supported by proposed mechanism

演进类型	描述
entitySubstitute(e_1, e_2)	实体e_1被实体e_2替代
merge(e_1, e_2)	实体e_2融合到实体e_1
split(e_1, e_2)	实体e_2从实体中分割
delLeafEntity(e_1, e_2)	为实体删除叶子实体e_2
addLeafEntity(e_1, e_2)	为实体添加叶子实体e_2
entityMove(e_1, e_2, e_3)	将的父实体从e_2变为e_3
addInnerEntity(e_1, e_2)	继承了e_2的层级关系并成为e_2的父实体
delInnerEntity(e_1, e_2)	e_2继承了的层级关系并不再是的子实体
addSubgraph(e_1, e_2)	e_2添加到e_1的子图中
delSubgraph（e_1，e_2）	e_2从e_1的子图中删除

算法1：演进关系分析算法

Input： 演进规则 Rules，差异分析结果 Diffs
Output： 演进表 ResEvo

```
1.  if rules ！ = null && Diffs ！ = null then
2.      forall rule ∈ Rules do
3.          if isApplicable(rule,Diffs)
4.              ResEvo. append(extractEvo(rule,Diffs))
5.          endif
6.      endfor
7.      if isMergeable(ResEvo) then
8.          ResEvo ← evoMerge(ResEvo)
9.      endif
10.  endif
11.  return ResEvo
```

若找到满足e_1规则的差异，则提炼为演进操作。如对于i和j两个代际，合并和分隔实体的逻辑判断语句如(8)-(9)所示：

$$\exists c_1, c_2 \in C_i \cap c_3 \in C_j \cap \text{mapEntity}(c_1, c_3) \cap \text{mapEntity}(c_2, c_3) - ((\exists c_4 \in C_j \cap \text{mapEntity}(c_1, c_4)) \cup (\exists c_5 \in C_j \cap \text{mapEntity}(c_2, c_5)) \rightarrow \text{merge}(c_3, c_1), \text{merge}(c_3, c_2), \tag{8}$$

$$\exists c_1 \in C_i \cap c_2, c_3 \in C_j \cap \text{mapEntity}(c_1, c_2) \cap \text{mapEntity}(c_1, c_3) - ((\exists c_4 \in C_i \cap \text{mapEntity}(c_4, c_2)) \cup (\exists c_5 \in C_i \cap \text{mapEntity}(c_5, c_3)) \rightarrow \text{split}(c_1, c_2), \text{split}(c_1, c_3). \tag{9}$$

用户可从演进表中提炼交通系统演进过程的语义表达。如新代际下技术的引入使系统功能进一步细分或融合、系统中信息交互对的概念移动以及支撑系统运行的硬件子图的增删改等高度集成化的语义信息。

2.3　架构自适应转换方案

如图3所示，输入某代际下的架构X以及期望转换到的代际Y后，一方面模块读取架构X包含的信息，并解析为实体、关系和属性集合。另一方面，读取并整合代际X到代际Y的演进表。根据架构包含的实体，从演进表中提取有关的演进关系。最终，根据演进关系对架构X包含的相关关系和属性信息进行增删改，依次实现架构从X到Y代际的逐步演进。对应的伪代码如算法2所示。

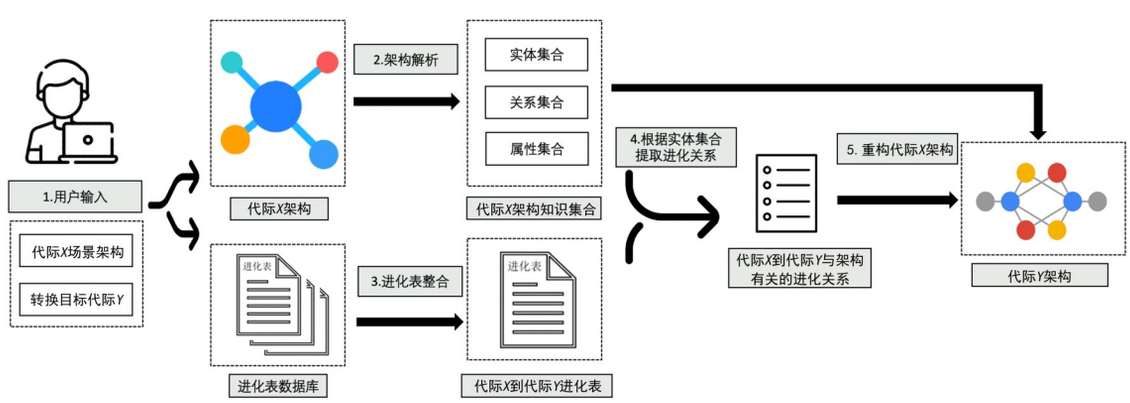

图3　架构自适应转换方案示意图

Fig. 3　Schematic diagram of the architecture adaptive conversion solution

算法 2：架构自适应转换算法

Input： 演进表集合 resEvoList， 代际 X 架构 oldArch，目标代际 Y

Output： 代际 Y 架构 newArch

```
newArch = oldArch
while X < Y do
    oldArch = newArch
    resEvo = resEvoList. get（X， ++X）
    relaventDiff = null
    forall oldEntity ∈ oldArch. getEntities（） do
        if exists（resEvo. hasMapEntity（oldEntity， newArch. getEntities（））） then
            relaventDiff. add（map（oldEntity））
        endif
        if exists（resEvo. hasRelaventDelEntity
            （resEvo. getRelationDiff（）， oldEntity， newArch. getEntities（））） then
            relaventDiff. add（del（oldEntity））
        endif
    end for
    forall newAddEntity ∈ resEvo. getAddEntities（） do
        if exists（resEvo. hasRelaventAddEntity
            （resEvo. getRelationDiff（），newAddEntity，oldArch. getEntities（））） then
            relaventDiff. add（add（newEntity））
        endif
    end for
    newArch. setEntities（resEvo. handleEntityEvo（relaventDiff））
    newArch. setRelations（resEvo. handleRelationEvo
    （oldArch. getEntities（）， newArch. getEntities（），resEvo. getRelationDiff（））
    newArch. setAttributes（resEvo. handleAttributeEvo
    （newArch. getEntities（），resEvo. getAttributeDiff（））
end while
return newArch
```

架构自适应演进方法以系统知识图谱为基础，以系统在代际间的演进关系为媒介，实现架构在不同代际间的自适应转换。相较于传统框架，基于 ATS 知识库，不仅能针对不同场景、自主化代际自适应生成架构，也能通过架构信息的结构化描述，提高系统运行效率，减少系统跨代际运行所需干预。

3 实验与评估

3.1 数据集与建模

根据前面的讨论，基于表征 ATS 跨代际演进的驾驶自动化子系统知识库，对提出方法进行实验评估。

3.2 评价指标与实验设计

以演进分析的准确度、可逆性、语义集成度以及运行时间为评估指标进行评价。

（1）演进准确度：基于代际 X 到代际 Y 的演进关系，可将代际 X 下的知识库重构为代际 Y 下的知识库，若得到的重构知识库与代际 Y 下的知识库全等，则代际 X 到代际 Y 知识库的演进关系准确。对应的表达式为

$$\mathrm{evo}_{XY}(K_X) \cap K_Y = \mathrm{evo}_{XY}(K_X) \cup K_Y, \tag{10}$$

式中 $\mathrm{evo}_{XY}(K_X)$ 为基于代际 X 到代际 Y 知识库的演进关系重构后的代际 X 知识库。K_Y 为代际 Y 下的知识库。

（2）演进可逆性：在上述基础上，基于代际 X 到代际 Y 知识库的演进关系，逆推代际 Y 到代际 X 知识库的演进关系，对代际 Y 下知识库进行重构，若得到的知识库与代际 X 下的知识库全等，则演进关系可逆性。对应的表达式为：

$$\mathrm{evo}'_{XY}(K_Y) \cap K_X = \mathrm{evo}'_{XY}(K_Y) \cup K_X, \tag{11}$$

式中 $\mathrm{evo}'_{XY}(K_Y)$ 为逆推得到的代际 Y 到代际 X 知识库的演进关系。K_X 为代际 X 下的知识库。

（3）演进关系语义集成度：定义语义集成度如式(12)所示。

$$I = 1 - \frac{C_{\mathrm{evo}}}{C_{\mathrm{basic}}}, \tag{12}$$

式中 I 为语义集成度指标，C_{basic} 为基本演进关系，包括实体、关系和属性的添加、删除，代表演进分析中语义集成度最低的演进关系。C_{evo} 为算法计算出的演进关系。

（4）运行时间：定义运行时间指标为从程序读入两个代际的知识图谱到完成演进分析耗费的计算时间(wall-clock time)。

以通用的知识图谱差异分析算法 OWL-Diff（Kremen et al.，2011）、Conto-Diff 以及可支持演进规则数量更多的 Dyn-Diff 算法作为基准进行对比，评估本文提出的演进分析方法的语义集成度和算法复杂度。

3.3　实验结果与分析

3.3.1　准确度与可逆性结果分析　以ATS驾驶自动化知识库由L1经L2演进到L3为例，本文方法识别出的演进关系类型及数量如图4所示。其中，add与delete操作分别包括实体、关系和属性的增添和删除差异。所提出方法能够从差异分析结果中提取出多项演进操作。

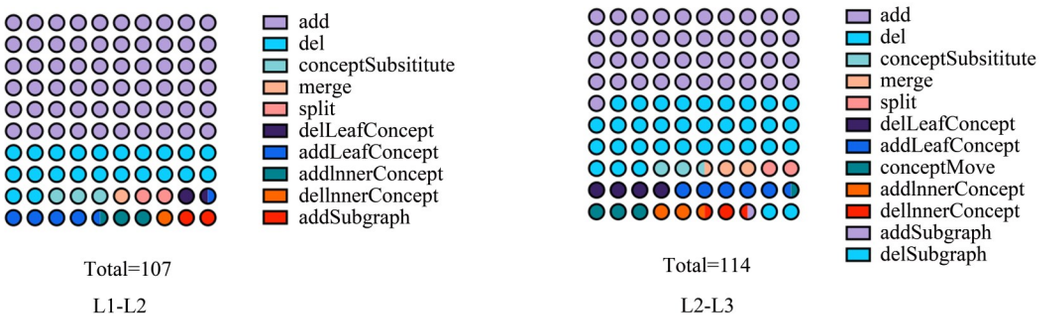

图4　分析出的演进操作类型及数量

Fig. 4　Type and number of evolution operations analyzed

应用演进关系对L1和L2代际知识图谱进行重构。重构前后各类知识数量、重构后知识库$\mathrm{evo}_{XY}(K_X)$与目标代际知识库K_Y的关系，如表4所示。在两个测试结果中，$\mathrm{evo}_{XY}(K_X)$包含的各类知识数目与K_Y保持一致，且$\mathrm{evo}_{XY}(K_X)\cap K_Y=\mathrm{evo}_{XY}(K_X)\cup K_Y$，即$\mathrm{evo}_{XY}(K_X)$与$K_Y$全等。基于评价指标(1)，本文方法可准确输出知识在各代际间的演进关系。

同理，应用逆向演进关系对K_Y重构，重构前后各类知识数量、重构后知识图谱$\mathrm{evo}'_{XY}(K_Y)$与初始代际知识图谱K_X的关系，如表5所示。基于评价指标(2)，即演进的可逆性，本文方法输出的演进关系可实现多代际的可逆转换，支撑变化的精准溯源。

3.3.2　语义集成度分析　如图5所示，在语义表达解释性上，通过集成演进关系语义信息，本文方法在两次试验中的语义集成度均显著高于对比方案，比最好水平分别提升了49%和42%。如图6所示，实体和关系知识的演进关系数量相比对照方案的最佳水平平均减少了28.1%和15.2%。由于属性知识从属于实体，表现形式较为单一，与对比方案的差异不大。

表4　重构前后知识图谱包含的各类知识数量及关系

Table 4　Quantitative and the relationship of origin KG before and after the reconstruction

代际	L1-L2			L2-L3		
	实体	关系	属性	实体	关系	属性
K_X	141	238	90	190	368	123
K_Y	190	368	123	187	378	127
$\mathrm{evo}_{XY}(K_X)$	141	238	90	190	368	123
$\mathrm{evo}_{XY}(K_X)\cap K_Y$	141	238	90	190	368	123
$\mathrm{evo}_{XY}(K_X)\cup K_Y$	141	238	90	190	368	123

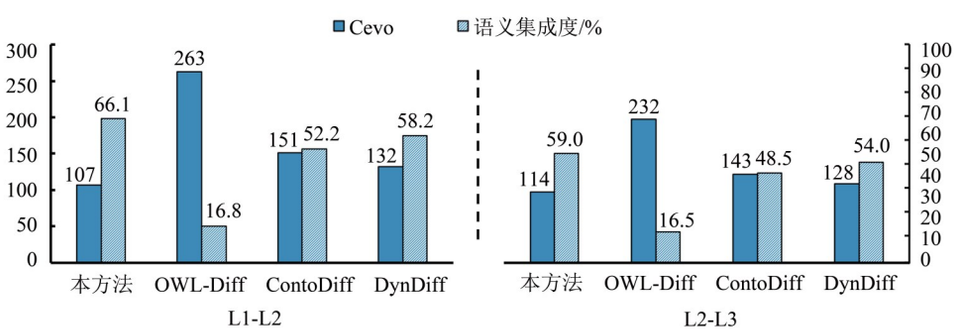

图5　语义集成度评估结果

Fig. 5　Semantic integration evaluation results

表5 逆向演进表重构前后知识图谱包含的各类知识数量及重构前后知识图谱的关系

Table 5 Quantitative and relationship between various types knowledge contained in the KG before and after the reconstruction of destination KG

代际	L1-L2			L2-L3		
	实体	关系	属性	实体	关系	属性
K_X	141	238	90	190	368	123
K_Y	190	368	123	187	378	127
$\mathrm{evo}_{XY}(K_X)$	190	368	123	187	378	127
$\mathrm{evo}_{XY}(K_X) \cap K_Y$	190	368	123	187	378	127
$\mathrm{evo}_{XY}(K_X) \cup K_Y$	190	368	123	187	378	127

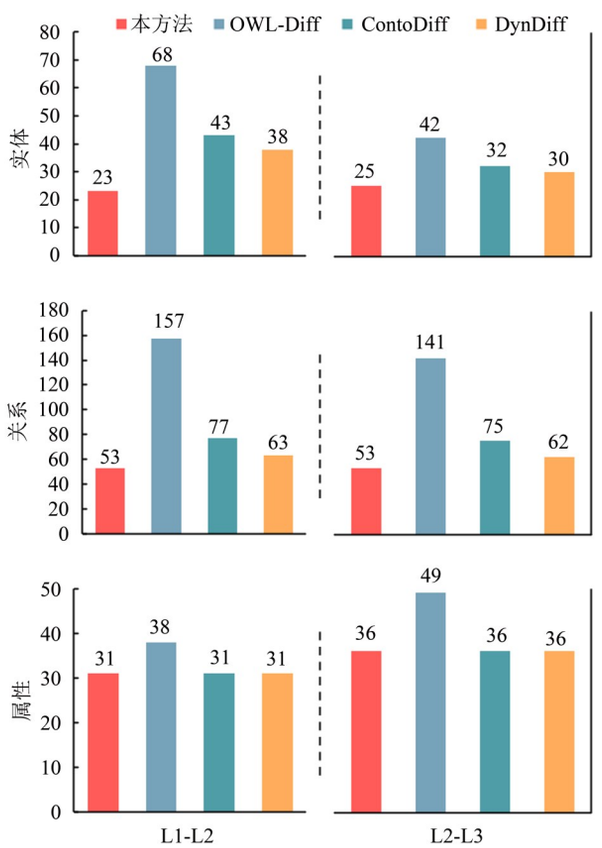

图6 各方案分析结果的演进关系数量

Fig. 6 Number of evolution relationships for each mechanism

3.3.3 运行时间分析 对比演进分析方案对数据集进行演进分析时所耗费的计算时间，评估运行时间指标。如表6所示，本文方法运行时间接近DynDiff，略高于ContoDiff。OWL-Diff方案的运行时间最少，实现了近10倍的运行时间提升。可能因OWL-Diff侧重于对基础差异的分析，对语义集成部分处理较少。

表6 运行时间评估结果

Table 6 Wall-clock time assessment result

方法	代际	运行时间/s
本文方法	L1−L2	1. 618
	L2−L3	1. 591
OWL-Diff	L1−L2	0. 118
	L2−L3	0. 295
ContoDiff	L1−L2	1. 532
	L2−L3	1. 692
DynDiff	L1−L2	1. 648
	L2−L3	1. 797

综上所述，基于多代际的ATS自动驾驶知识库，从演进关系的准确度、可逆性、语义集成度以及运行时间四个维度对本文方法进行验证评估。本文方法具有更高的准确性和可逆性，保证了自适应转换的精度和可复现性。与传统算法相比，本文方法在演进分析中取得了更高的语义集成度，提供了更为准确且集成的架构代际转换操作，算法复杂度合理，运行效率能得到保证。

4 结 论

本文首次提出基于知识图谱演进分析的ATS架构自适应演进方法，利用本体理论构建ATS知识图谱，对代际数据进行高效建模，之后利用知识库的演进表作为先验知识，对需要进行演进的架构进行重构，生成目标代际下的架构。通过对演进表的复用，在提升架构自适应计算效率的同时，也保证了架构重构的可靠性。此外，本文还提供了一种基于规则的演进分析方法，通过3步式分析，实现面向不同代际ATS的演进分析。对比实验表明，本文方案具有演进分析准确、结果具有可逆性的特点。同时，所提出的演进分析方法在语义集成度上也有较大提升。

随着深度学习的进一步成熟，后续研究将进一步结合知识图谱与深度学习模型，对架构自适应设计方案进行完善，减少人工干预及先验知识的输入，进一步提高架构自适应转换的可靠性。

参考文献：

党引弟，宋宁宁，2019. 动态自适应演进安全架构研究[J]. 信息技术与网络安全，38(10)：18-23.

关积珍，2022. 智能交通系统发展演进及其代际特征[J]. 人工智能，9(4)：40-49.

李青山，廉宗民，王璐，等，2021. 空间飞行器控制软件的动态自适应演化方法[J]. 空间控制技术与应用，47(2)：63-72.

裴建中，2018. 道路工程学科前沿进展与道路交通系统的代际转换[J]. 中国公路学报，31(11)：1.

邱凌，张安思，李少波，等，2022. 航空制造知识图谱构建研究综述[J]. 计算应用研究，39(4)：968-977.

唐进君，庹昊南，刘佑，等，2022. 基于 BERT-Bi-LSTM-CRF 模型的自主式交通系统参与主体识别方法[J]. 交通信息与安全，40(5)：80-90.

王淑营，李雪，黎荣，等，2022. 基于知识图谱的高速列车知识融合方法[J]. 西南交通大学学报：1-11.https://kns.cnki.net/kems/detail/51.1277.U.2022711.1502.002.html.

魏伟，郑来，蔡铭，2022. 面向自主式交通的智能交通系统用户需求研究[J]. 交通科技与经济，24(2)：1-7.

严新平，褚端峰，刘佳仑，等，2021. 智能交通发展的现状，挑战与展望[J]. 交通运输研究，7(6)：2-10+22.

张明悦，金芝，赵海燕，等，2020. 机器学习赋能的软件自适应性综述[J]. 软件学报，31(8)：2404-2431.

张栋豪，刘振宇，郑维强，等，2021. 知识图谱在智能制造领域的研究现状及其应用前景综述[J]. 机械工程学报，57(5)：90-113.

赵娜，袁家斌，徐晗，2014. 智能交通系统综述[J]. 计算机科学，41(11)：7-11+45.

BENAVIDES S D，CARDOSO S D，SILVEIRA M D A，et al，2022. Dyndiff：A tool for comparing versions of large ontologies[C]//5th Workshop on Semantic Web Solutions for Large-Scale Biomedical Data Analytics. CEUR：29-30.

BUCHGEHER G，GABAUER D，MARTINEZ-GIL J，et al，2021. Knowledge graphs in manufacturing and production：A systematic literature review [J]. IEEE Access，9：55537-55554.

HARTUNG M，GROSS A，RAHM E，2013. Conto-Diff：Generation of complex evolution mappings for life science ontologies[J]. J Biomed Inform，46(1)：15-32.

KREMEN P，SMID M，KOUBA Z，2011. OWLDiff：A practical tool for comparison and merge of OWL ontologies [C]//22nd International Workshop on Database and Expert Systems Applications. Toulouse，France：IEEE：229-233.

NOY N F，MUSEN M A，2002. Promptdiff：A fixed-point algorithm for comparing ontology versions [J]. AAAI/IAAI：744-750.

OSBORNE F，MOTTA E，2018. Pragmatic ontology evolution：Reconciling user requirements and application performance [C]// 17th International Semantic Web Conference. Springer：495-512.

SANTOS J S，SILVA V T，AZEVEDO L G，et al，2020. An experimental analysis of tools for ontology evolution management [C]// 22nd International Conference on Enterprise Information Systems. SciTePress：111-121.

WILLIAMS B，2021. Automated driving levels[EB/OL]. Automated Vehicles and MassS：Removing the Barriers. https://ieeexplore.ieee.org/abstract/document/9466297.

YOU L，HE J，ZHAO J，et al，2022a. A federated mixed logit model for personal mobility service in autonomous transportation systems[J]. Systems，10(4)：117.

YOU L，HE J，WANG W，et al，2022b. Autonomous transportation systems and services enabled by the next-generation network[J]. IEEE Netw，36(3)：66-72.

ZHOU Z S，CAI M，XIONG C，et al，2022. Construction of autonomous transportation system architecture based on system engineering methodology[C]// 25th International Conference on Intelligent Transportation Systems (ITSC). Macau，China：IEEE：3348-3353.

（责任编辑　王海蓉）