软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, 2023,34(7):3002−3021 [doi: 10.13328/j.cnki.jos.006855]

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

自动驾驶交叉路口测试场景建模及验证方法*

夏春艳[1,2], 黄 松[1], 郑长友[1], 张清睿[1], 王 宇[1], 魏瑀皓[1]

[1](中国人民解放军陆军工程大学 指挥控制工程学院, 江苏 南京 210007)

[2](牡丹江师范学院 计算机与信息技术学院, 黑龙江 牡丹江 157012)

通信作者: 黄松, huangsong@aeu.edu.cn

摘 要: 自动驾驶汽车在缓解交通拥堵和消除交通事故方面发挥着重要作用. 为了保证自动驾驶系统的安全性和可靠性, 在自动驾驶汽车部署到公共道路之前, 必须进行全面的测试. 现有的测试场景数据大多来源于交通事故和交通违法场景, 而且自动驾驶系统最基本的安全需求就是遵守交通法规, 这充分体现了自动驾驶汽车遵守交通规则的重要性. 然而, 目前严重缺少针对交通法规构建的自动驾驶测试场景. 因此, 从交通法规出发, 根据自动驾驶系统的安全需求, 提出了交叉路口测试场景的 Petri 网建模及形式化验证方法. 首先, 依据自动驾驶测试场景对交规进行分类, 提取适合自动驾驶汽车的文本交规, 并进行半形式化表征; 其次, 以覆盖道路交通安全法规以及测试场景功能测试规程为目标, 融合交叉路口场景要素的交互行为, 合理选择并组合测试场景要素, 布设交叉路口测试场景; 然后, 基于交规的测试场景被建模为一个 Petri 网, 其中, 库所描述自动驾驶汽车的状态, 变迁表示状态的触发条件, 并选择时钟约束规范语言(CCSL)作为中间语义语言, 将Petri网转换为一个可进行形式化验证的中间语义模型, 提出了具体的转换方法; 最后, 通过 Tina 软件分析验证交规场景模型的活性、有界性和可达性, 结果表明了所建模型的正确性, 并基于 SMT 的分析工具 MyCCSL 来分析 CCSL 约束, 采用 LTL 公式以形式化方法验证交规场景模型的一致性.

关键词: 自动驾驶; 测试场景; 交规模型; 形式化验证

中图法分类号: TP311

中文引用格式: 夏春艳, 黄松, 郑长友, 张清睿, 王宇, 魏瑀皓. 自动驾驶交叉路口测试场景建模及验证方法. 软件学报, 2023, 34(7): 3002–3021. http://www.jos.org.cn/1000-9825/6855.htm

英文引用格式: Xia CY, Huang S, Zheng CY, Zhang QR, Wang Y, Wei YH. Modeling and Verification Method of Intersection Test Scenario for Automated Driving. Ruan Jian Xue Bao/Journal of Software, 2023, 34(7): 3002−3021 (in Chinese). http://www.jos.org.cn/1000-9825/6855.htm

Modeling and Verification Method of Intersection Test Scenario for Automated Driving

XIA Chun-Yan[1,2], HUANG Song[1], ZHENG Chang-You[1], ZHANG Qing-Rui[1], WANG Yu[1], WEI Yu-Hao[1]

[1](College of Command and Control Engineering, Army Engineering University of PLA, Nanjing 210007, China)

[2](College of Computer and Information Technology, Mudanjiang Normal University, Mudanjiang 157012, China)

Abstract: Autonomous vehicles play an important role in easing traffic congestion and eliminating traffic accidents. In order to ensure the safety and reliability of the autonomous vehicle, there must be an all-around test before they are deployed on public roads. Most of the existing test scenario data come from traffic accidents and traffic violations. Furthermore, the most fundamental safety requirement of an autopilot system is that autonomous vehicles should comply with traffic law, which fully reflects the importance of autonomous vehicles complying with traffic rules. Nevertheless, there is a severe lack of test scenarios built for the traffic law. Therefore, in this study, the

* 基金项目: 牡丹江师范学院国家级课题培育项目(GP2022008); 牡丹江师范学院学科建设揭榜挂帅项目(MSYSYL2022008); 黑龙江省省属高等学校基本科研业务费(1452ZD010); 黑龙江省高等教育教学改革重点委托项目(SJGZ20200175)

本文由“形式化方法与应用”专题特约编辑董云卫教授、刘关俊教授、毛晓光教授推荐.

收稿时间: 2022-09-04; 修改时间: 2022-10-08; 采用时间: 2022-12-05; jos 在线出版时间: 2022-12-30

safety requirements of the autopilot system are extracted from the traffic law perspective, and a Petri net modeling and formal verification method for intersection test scenarios is proposed. Firstly, the traffic rules are classified according to the test scenarios of automated driving, the rule text suitable for the autonomous vehicle is extracted and semi-formalized. Secondly, with the aim of covering traffic law and function testing procedure of the test scenario, the interactive behavior of the intersection scene elements is integrated, the typical test scene elements are selected and combined to deploy the intersection test scenarios. Then, the test scenario based on traffic rules is modeled as a Petri net, in which places describe the state of the autonomous vehicle and transitions represent the trigger condition of the state. Moreover, the Clock Constraint Specification Language (CCSL) is chosen as the intermediate semantic language to convert the Petri net into an intermediate semantic model which can be formally verified. A specific conversion method is proposed. Finally, Tina is used to verify the activity, boundedness, and accessibility of the traffic-law scenario model, and the experimental results prove the validity of the model. Besides, the CCSL constraints are analyzed through the analysis tool MyCCSL which is based on the SMT and the consistency of the model is verified by the LTL formula.

Key words: automated driving; test scenario; traffic law model; formal verification

随着人工智能的快速发展和物联网技术的显著进步, 近 10 年来, 自动驾驶技术取得了长足的稳定发展. 在可预见的未来, 自动驾驶汽车被广泛认为是人们日常生活中最受欢迎的人工智能应用之一. 尽管前景光明, 但自动驾驶技术的发展也面临着新的挑战, 其中, 安全是最令人担忧的问题. 例如, 2018 年, 一辆 Uber 的自动驾驶汽车在公共道路上行驶时, 未能避让前方行人而发生车祸事故, 导致行人死亡[1]. 这也警示我们: 由于开放、动态和不可预测的复杂交通环境, 自动驾驶车辆必须经过严格且全面的测试, 才能确保车辆不会发生不必要的行为.

自动驾驶汽车测试方法主要分为基于里程的测试方法和基于场景的测试方法: 基于里程的测试方法是指测试车辆在一定的交通环境下连续行驶, 不需要预先设定测试任务和目标; 基于场景的测试方法是指测试车辆通过预先设定的场景, 完成某项特定任务和目标. 里程测试主要针对车辆的综合性能, 一般在车辆开发的最后阶段进行, 要求车辆具备较为完整的功能和较为可靠的性能, 其缺点是测试周期长、效率低、成本高昂; 而且在公共道路上进行测试, 还面临交通安全风险. 相比于里程测试, 场景测试的应用灵活, 测试效率高, 针对性强, 且具有可重复性和可扩展性, 有助于自动驾驶汽车功能的快速迭代和完善. 目前, 针对场景测试基础理论、应用方法、技术的研究越来越受到产业界和学术界的重视, 但场景测试方法仍存在一些亟待突破的技术难点, 包括场景提取、筛选以及测试场景的构建等关键技术.

目前, 研究人员针对自动驾驶测试提出了许多关键场景生成的不同技术[2−10]. 测试场景可以由测试人员根据经验和理论方法进行构建, 也可以从相关数据中筛选和提取. 由此形成的场景, 可以部署到虚拟环境或者真实的受控测试场地, 从而实现对自动驾驶汽车的测试和验证. 然而, 现有的大多数自动驾驶测试方法都没有充分考虑交通安全法律法规的限制, 例如“自动驾驶汽车不可以违反交通法规”, 也没有相关的标准来实现和验证[11], 这导致了自动驾驶汽车在公共道路行驶中存在严重的交通安全隐患. 此外, 在已有的测试场景中, 很多数据来源于交通事故场景和交通违法场景. 受此启发, 如果构建测试场景时就能够充分考虑交通法规的约束, 那么很大程度上能够提高自动驾驶系统的安全性. 鉴于此, 本文提出一种基于交通法规构建自动驾驶交叉路口测试场景模型及形式化验证方法, 旨在从交通法规出发, 提取自动驾驶系统的安全需求, 建模交叉路口测试场景, 并对场景模型进行形式化验证, 具体过程如下.

(1) 基于交通安全法律法规, 按照自动驾驶测试场景对交通法规进行分类, 提取适合自动驾驶测试场景的交通规则, 并将文本形式的交通规则进行半形式化的表征, 为构建符合交规约束的复杂测试场景提供理论依据.

(2) 根据自动驾驶分级测试项目要求以及提取的半形式化交通法规, 结合实际交通路况, 融合相应测试场景要素的交互行为, 合理选择测试场景要素, 以灵活手段对各个测试场景要素进行布设, 组合复杂的测试场景.

(3) 基于 Petri 网理论构建交规测试场景模型, 模型中的库所描述的是自动驾驶汽车的行驶状态, 变迁表示各个状态之间转变的触发条件; 然后选择时钟约束规范语言(clock constraint specification

language, CCSL)作为中间语义语言, 将 Petri 网转换为一个可进行形式化验证的中间语义模型, 并给出具体的 Petri 网到 CCSL 约束的转换方法.

(4) 通过 Tina 软件分析验证交规场景模型的活性、有界性和可达性, 结果表明了所提方法构建交规场景模型的正确性, 并基于 SMT 的分析工具 MyCCSL 来分析 CCSL 约束, 采用 LTL 公式以形式化方法验证了交规场景模型的一致性.

本文第 1 节介绍相关工作. 第 2 节介绍基础知识. 第 3 节说明场景建模方法. 第 4 节给出模型验证方法. 第 5 节验证所提方法的有效性. 第 6 节讨论场景模型的相关问题. 第 7 节总结全文和展望下一步工作.

1 相关工作

在智能交通系统中, 自动驾驶汽车在缓解交通拥堵和消除交通事故方面起着至关重要的作用. 许多公司都致力于这个领域, 如谷歌 Waymo、百度 Apollo 和 Autoware.

目前, 自动驾驶测试方法可分为里程测试和场景测试. 对自动驾驶汽车进行严格的里程测试是非常必要的, 但其缺点是风险较高、成本昂贵. 因此, 自动驾驶汽车在执行里程测试之前, 往往会进行广泛的基于场景的测试. 当前较为流行的测试场景生成方法可分为基于数据驱动的方法和基于模型驱动的方法: 在数据驱动的方法中, 场景是由现有的数据集生成的, 如碰撞报告[2−4]和真实世界的驾驶记录[5−7]; 在模型驱动的方法中, 基于系统模型的评价结果生成具体的场景[8−10]. 上述诸多测试场景生成方法, 通过实验验证均取得了较好的效果, 但这些方法都没有充分考虑交通安全法律法规的约束, 使得生成的测试场景存在忽视交通法规的潜在可能性. 为此, 本文提出一种基于交规建模的测试场景生成方法, 目的是充分考虑交通法规在生成测试场景过程中的重要性.

Petri 网理论在构建自动驾驶测试场景时也得到了广泛的应用. Kumar[12]设计了基于 Petri 网模型的交叉路口避碰系统和巡航控制系统, 包括两个速度和距离的内部控制器与 3 个避免碰撞的外部控制器, 分析验证了模型的正确性与优越性. Tang 等人[13]提出了一种通过地图建模进行自动驾驶汽车路线覆盖的测试方法, 该方法首先将地图建模为一个 Petri 网模型, 然后基于 Petri 网模型定义了交叉路口的拓扑特征和路线特征用于交叉路口的分类. 实验结果表明, 该方法实现了更高的路线覆盖率. 由于场景可被认为是场面的时间序列, 其转换由动作或者时间触发, 而 Petri 网正是为离散事件建模, 特别是处理并发性和非确定性问题能够体现出强大的功能, 这就使得 Petri 网理论在构建场景模型的方法中取得了较好的效果.

安全关键软件失效引起的事故比例越来越高, 而软件安全性特别强调需求的重要性. 目前, 形式化方法已经被理论和实践证明是一种提高软件安全性的有效途径. 但对 Petri 网进行有针对性的扩展, 以支持安全关键软件的形式化验证方法还缺少深入的研究. Lima 等人[14]从序列图表示的 UML 交互中创建一个基于 PROMELA 的模型, 并使用 SPIN 模型检查器来模拟执行, 验证了用线性时间逻辑(LTL)编写的属性. Han 等人[15]实现了从序列图到时间自动机网络的全映射, 使用 UPPAAL 模型检查器实现了形式化的验证. 基于序列图和 Petri 网之间关系的定义和解释, Soares 等人[16]提出了一种将序列图自动转换为 Petri 网的方法, 使用 CPN 工具来执行, 并在建模方面考虑形式化变换规则. 上述方法适用于基本 Petri 网的有限扩展, 并没有给出 Petri 网的形式化定义以及从模型到验证语言的转换方法. 为了解决 Petri 网对安全关键软件进行形式化验证时在安全性描述、自动化程度和验证效率方面的不足, 本文提出了一种基于 Petri 网的安全需求的一致性验证方法, 依赖时钟约束规范语言(CCSL)作为中间语义语言, 将安全关键系统的安全需求与实际执行调度进行比较, 以形式化方法有效地验证了软件安全需求的一致性.

2 基础知识

本文所提方法主要基于 Petri 网建模和 CCSL 约束验证, 下面就相关概念和基本知识予以介绍.

2.1 Petri网基本理论

1962 年, Petri 网概念首次被提出并用于描述通信结构模型[17]. 这种网状的系统结构模型非常直观, 可以

通过描述事件的状态转移关系实现动态分析系统的功能[18]. 目前, Petri 网作为一种强大的建模和分析工具, 在通信和计算机等领域得到了广泛的应用, 已经发展成为系统建模分析领域中非常重要的一种工具[19]. 因此, 本文选用 Petri 网对复杂的测试场景进行建模, 不仅可以直观地反映系统状态转移关系, 还能对系统进行动态分析.

Petri 网的基本结构元素包括库所、变迁和弧: 库所代表的是模拟系统中可能出现的状态, 用圆形节点表示; 变迁描述了系统两个状态之间转移的动作, 用矩形节点表示; 弧是有方向的连接线(有向弧), 用于连接库所和变迁, 表示状态和动作之间的转换, 任意两个变迁或者库所之间是不存在弧的. 圆形节点内的黑点为库所内的标识, 代表系统的资源, 变迁的发生使得标识在库所之间转移, 系统内标识的转移代表了资源的流动. 当变迁描述的动作发生后, 标识会转移到下一个满足条件的库所中, 这个过程描述了系统状态的动态变化. 库所中存在标识, 表明库所描述的条件为真; 反之, 条件为假.

定义 1. 将一个库所/变迁系统(place/transition system)表示为一个六元组: $\Sigma=(P,T,F,H,W,M)$, 其中: P 为有穷、非空库所集合, 图形表示为一个圆圈; T 为有穷、非空变迁集合, 图形表示为一个矩形; F 为库所与变迁之间的有向弧集合; $H:p\to\{1,2,3\ldots\}$称为库所容量函数; $W:f\to\{1,2,3\ldots\}$称为权函数; $M:p\to\{0,1,2,\ldots\}$是Σ的一个标识, 图形上在库所的圆圈里使用小黑点进行表示, 有几个标识就对应几个小黑点, 同时满足条件:

$$\forall p\in P:M(p)\leqslant H(p) \tag{1}$$

当Σ满足条件:

(a) $\forall p\in P:H(p)=\infty$,

(b) $\forall f\in F:W(f)=1$,

即每个库所的容量都是无穷大, 每条有向弧的权函数都是 1 时, 就称Σ为一个原型 Petri 网. 其中, 原型 Petri 网可以表示为一个四元组: $PN=(P,T,F,M)$.

原型 Petri 网变迁的发生规则为:

(1) 对于变迁 $t\in T$, 如果

$$\forall p\in P:p\in {}^\bullet t\to M(p)\geqslant 1 \tag{2}$$

即如果变迁 t 的输入库所 p 中, 标识 $M(p)$大于等于 1, 则此时变迁 t 具有发生权, 记为 $M[t\rangle$.

(2) 当在标识 M 下变迁 t 可以发生, 即 $M[t\rangle$, 变迁 t 发生过后将得到一个新的标识 M'(记为 $M[t\rangle M'$), 对于$\forall p\in P$,

$$M'(p)=\begin{cases}M(p)-1, & 若p\in {}^\bullet t-t^\bullet\\ M(p)+1, & 若p\in t^\bullet-{}^\bullet t\end{cases} \tag{3}$$

其中, ${}^\bullet t$ 表示变迁 t 的前置库所, 或称输入库所; $t^\bullet$表示变迁 t 的后置库所, 或称输出库所.

2.2 时钟约束语言(CCSL)

时钟约束规范语言最初被设计为一种简单的语言, 用于表达 MARTE (modeling and analysis of real-time and embedded systems)模型时钟之间的约束, 现在已经逐步发展为独立于 UML 的具有形式化验证语义的语言, 在汽车领域得到广泛应用[20]. CCSL 使用逻辑时钟来描述事件之间的因果关系和时间关系. 逻辑时钟可以形式化地定义为一个二元组, 即 $C:=\langle I,\prec\rangle$, 其中: I 是一组时刻; $\prec$称为严格优先, 是 I 上的严格顺序关系; $C[k]$表示 I 中的第 k 个时刻, $k\in\mathbf{N}^+$. 每个逻辑时钟被定义为一个事件发生的序列, 事件的每一次发生都称为时刻, 也称为滴答.

在 CCSL 中, 逻辑时钟的依赖关系被分为两类: 时钟约束和时钟定义. 时钟约束描述的是两个时钟之间的依赖关系, 如优先关系、因果关系、从属关系和互斥关系这 4 个原始约束. 时钟定义, 即定义新的时钟,如并关系、交关系、下确界关系、上确界关系、相对延迟关系和周期关系等, 并提供基于现有时钟创建新时钟的规则. 为方便处理, 本文将时钟约束和时钟定义统称为 CCSL 约束. 表 1 给出了常用的 CCSL 约束, 并描述了其直观含义.

表 1 时钟约束的语法和语义

语法	语义	描述
$C_1 \prec C_2$	$\forall i \in \mathbf{N}^+, C_1[i] \prec C_2[i]$	时钟 C_1 比时钟 C_2 早滴答
$C_1 \preccurlyeq C_2$	$\forall i \in \mathbf{N}^+, C_1[i] \preccurlyeq C_2[i]$	时钟 C_1 不会慢于时钟 C_2 滴答
$C_1 \subseteq C_2$	$\forall i \in \mathbf{N}^+, \exists j \in \mathbf{N}^+, C_1[i] \equiv C_2[j]$	如果时钟 C_1 滴答，则时钟 C_2 滴答
$C_1 \# C_2$	$\forall i, j \in \mathbf{N}^+, C_1[i] \neq C_2[j]$	时钟 C_1 和时钟 C_2 不会同时滴答
$C_1 = C_2 + C_3$	$\forall i \in \mathbf{N}^+, \exists j,k \in \mathbf{N}^+, (C_1[i] \equiv C_2[j]) \vee (C_1[i] \equiv C_3[k])$	定义时钟 C_1, C_1 滴答当且仅当 C_2 或 C_3 滴答
$C_1 = C_2 * C_3$	$\forall i \in \mathbf{N}^+, \exists j,k \in \mathbf{N}^+, (C_1[i] \equiv C_2[j]) \wedge (C_1[i] \equiv C_3[k])$	定义时钟 C_1, C_1 滴答当且仅当 C_2 与 C_3 同时滴答
$C_1 = C_2 \wedge C_3$	$\forall i \in \mathbf{N}^+, (C_3[i] \preccurlyeq C_2[i] \rightarrow C_1[i] \equiv C_3[i]) \vee (C_2[i] \preccurlyeq C_3[i] \rightarrow C_1[i] \equiv C_2[i])$	定义时钟 C_1, C_1 不会慢于 C_2 和 C_3 滴答
$C_1 = C_2 \vee C_3$	$\forall i \in \mathbf{N}^+, (C_3[i] \preccurlyeq C_2[i] \rightarrow C_1[i] \equiv C_2[i]) \vee (C_2[i] \preccurlyeq C_3[i] \rightarrow C_1[i] \equiv C_3[i])$	定义时钟 C_1, C_1 不会快于 C_2 和 C_3 滴答
$C_1 = C_2$ \$ d on C_3	$\forall m \in \mathbf{N}^+, \exists i,j,k \in \mathbf{N}^+, (C_1[i] \equiv C_2[k])) \wedge (C_3[i] - C_3[m]) = d$	定义时钟 C_1, C_3 每累计滴答 d 次, C_1 与 C_2 滴答情况一致; 特别地, 当 C_3 为一个一直滴答的时钟, 表示 C_1 与 C_2 滴答间隔 d 步

3 场景建模方法

鉴于自动驾驶汽车遵守交规对于道路安全行驶的重要性, 本文提出一种基于交规建模测试场景的生成方法, 该方法由 3 个部分组成.

3.1 交规提取

现有的交通法规是面向人类驾驶员制定的, 具有较高的抽象性, 且部分内容存在较大弹性. 对自动驾驶系统而言, 难以理解抽象的法规条文. 因此, 需将交通法规转化为明确化、定量化、符号化、模型化的数字化交规. 同时, 对于交规条款进行清晰化, 以便于机器理解. 例如, 道路交通法实施条例第四十七条: “机动车超车时, 应当提前开启左转向灯、变换使用远、近光灯或者鸣喇叭. 后车应当在确认有充足的安全距离后, 从前车的左侧超越, 在与被超车辆拉开必要的安全距离后, 开启右转向灯, 驶回原车道”. 对于这些法律条文, 人类易于理解, 但对于自动驾驶系统而言, 并不清楚“提前多少时间开启”“充足的安全距离是多少”“必要的安全距离是多少”. 因此, 对于抽象的文本交规, 进行模糊语义的明确化、参数取值清晰化, 方便转为机器执行语言是首要解决的问题.

依据《中华人民共和国道路交通安全法》《中华人民共和国道路交通安全法实施条例》《智能网联汽车道路测试与示范应用管理规范(试行)》《智能网联汽车运行安全测试技术要求(征求意见稿)》和《智能网联汽车自动驾驶功能测试规程(试行)》等国内现有的交通法律法规和标准, 提取适合自动驾驶系统的交通法规, 并根据基本测试场景进行分类, 为构建复杂场景模型提供理论依据.

例如, 依据《中华人民共和国道路交通安全法实施条例》第三十八条的规定, 可提取适合自动驾驶车辆通过交叉路口安全行驶的半形式化交通规则, 如图 1 所示.

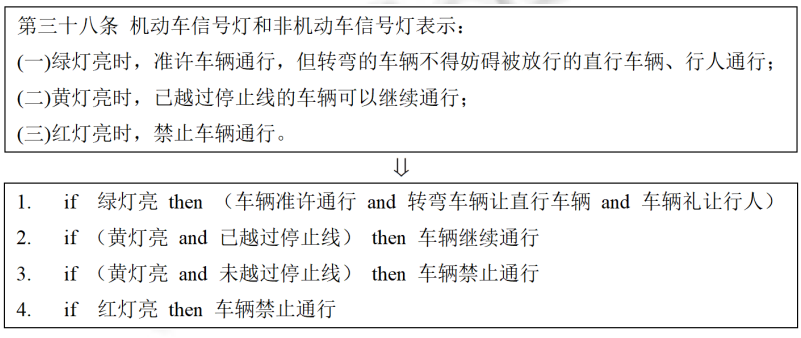


图 1 半形式化交规的提取

3.2 场景布设

研究表明, 测试场景的复杂性越大, 探测出自动驾驶系统性能缺陷的可能性越大, 对自动驾驶系统的挑战性也就越大, 进而可以认为, 场景复杂程度是影响场景质量的关键因素之一. 但是场景作为一个高度复杂的多源异构系统, 绝对不是一堆元素的简单堆叠, 因为场景生成过程中, 元素随机组合会导致场景状态空间爆炸问题. 由于场景具有不可预测、极其复杂、重复性差、无法穷举等特点, 选择一个通用的、被业界认可的场景要素整合方式, 是建立测试场景的关键.

本文根据自动驾驶分级的测试项目要求, 以及依据交规提取的半形式化表征约束条件, 融合交叉路口测试场景要素的交互行为, 合理选择组合并布设测试场景要素, 重构验证自动驾驶安全性的功能测试场景, 以便保障测试场景建模的完备性. 由于本文的目标是基于交规建模测试场景, 并验证场景模型的正确性和一致性, 目的是为自动驾驶生成测试用例提供满足交规的场景模型, 因此, 这里假设交通设施完备, 行人和障碍车辆都遵守交通规则, 没有异常行为. 下面以交叉路口左转测试场景为例, 说明基于交规的交叉路口场景的布设过程.

第 1 步. 提取交规.

依据交通安全法律法规提取适合自动驾驶测试场景的交通规则, 并将文本形式的交规进行半形式化的表征, 按照基本测试场景对交规分类, 获得交叉路口的交通约束规则, 进一步提取路口左转场景相关的半形式化规则, 如图 2 所示. 采用相同的方法, 可以提取交叉路口的直行测试场景、右转测试场景和掉头测试场景的交规, 这里不再赘述.

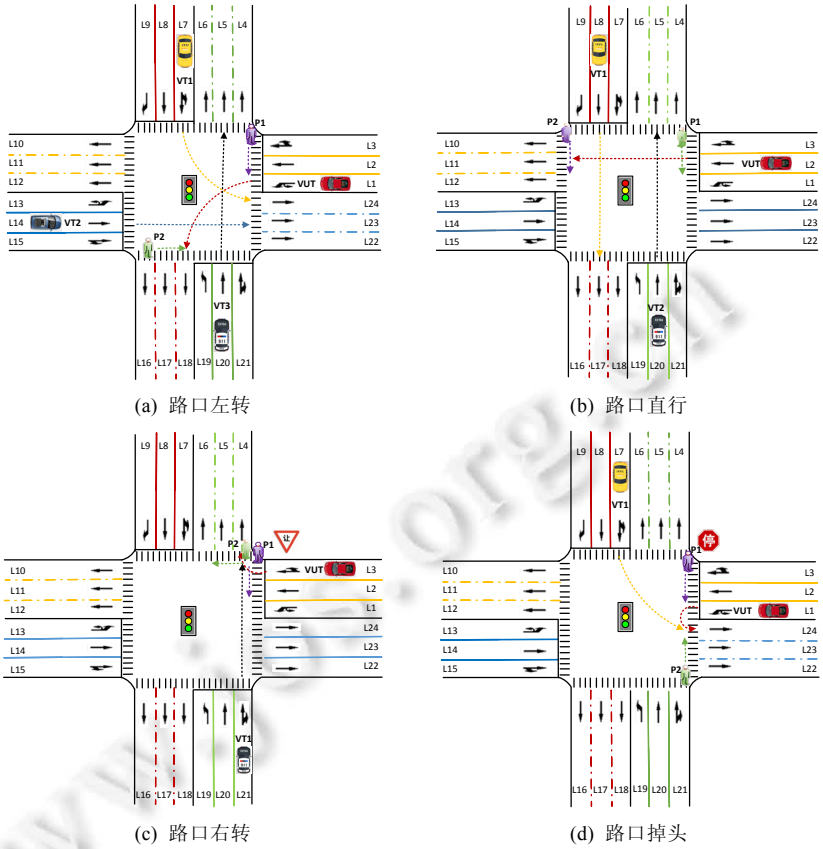
```
if 绿灯亮 then （车辆准许通行 and 转弯车辆让直行车辆 and 车辆礼让行人）
if （黄灯亮 and 车辆已越过停止线） then 车辆继续通行
if （黄灯亮 and 车辆未越过停止线） then 车辆禁止通行
if 红灯亮 then 车辆禁止通行
if 有导向车道 then 车辆按所需行进方向驶入导向车道
if 车辆向左转弯 then 提前开启左转向灯
if 相对方向行驶车辆 then 右转弯车辆让行左转弯车辆
if （有交通标志控制 or 有交通标线控制） then 优先通行的车辆先行
if 绿色箭头灯亮 then 本车道车辆按指示方向通行
if 红色交叉灯亮 then 禁止本车道车辆通行
if 车辆向左转弯 then （靠路口中心线左侧转弯 and 提前开启左转向灯）
if（车辆向左转弯 and 夜间行车）then（靠路口中心线左侧转弯 and 开启近光灯 and 开启左转向灯）
if 车辆遇放行信号 then 依次通过
if 车辆遇停止信号 then 依次停在停止线以外
if （车辆遇停止信号 and 没有停止线） then 在路口以外依次停车
if 交叉路口交通阻塞 then 在路口外依次停车等候
if （前方机动车停车排队 or 缓慢行驶） then 在路口外依次停车排队等候
if 没有交通信号灯 then （转弯车辆让行直行车辆 and 礼让行人）
if （没有交通标志控制 or 没有交通标线控制） then （车辆减速慢行 and 让行右方车辆）
if 遇有灯光信号、交通标志或交通标线与交通警察的指挥不一致 then 服从交通警察的指挥
```

图 2 向左转弯场景交通法规半形式化表征

第 2 步. 场景布设.

自动驾驶过程中, 行驶状态受人、车、路、环境等多种因素影响, 测试场景的设定应根据实际测试需求, 确定测试场景要素. 简单的场景设计会造成场景覆盖能力的不足; 而过于复杂的场景设计又会破坏物体相对关系, 损失场景真实性. 此外, 测试场景的设置还应符合交通规则的约束, 也就是各测试场景要素之间不应产生矛盾, 如车道与指示标志不一致、车道方向与车辆行驶方向不一致等. 因此, 以覆盖道路交通安全法规以及测试场景功能测试规程为目标, 结合实际交通路况, 融合交叉路口场景要素的交互行为, 合理选择并组合测试场景要素, 布设测试车辆(红色)路口左转测试场景, 如图 3(a)所示. 类似的方法可以布设交叉路口的直行测

试场景、右转测试场景和掉头测试场景，如图 3(b)、图 3(c)和图 3(d)所示.



(a) 路口左转　(b) 路口直行

(c) 路口右转　(d) 路口掉头

图 3　交叉路口场景布设

3.3 构建模型

Petri 网是一个可对复杂系统进行建模的图形化、数学化的建模语言，尤其适用于对并发系统、分布系统的建模. 可用 Petri 网来建模的典型场景有：同步过程、序列化过程、并发过程以及冲突选择. 因此，这里选择 Petri 网来建模基于交规的自动驾驶复杂场景. 下面以交叉路口左转场景为例，介绍场景建模的过程.

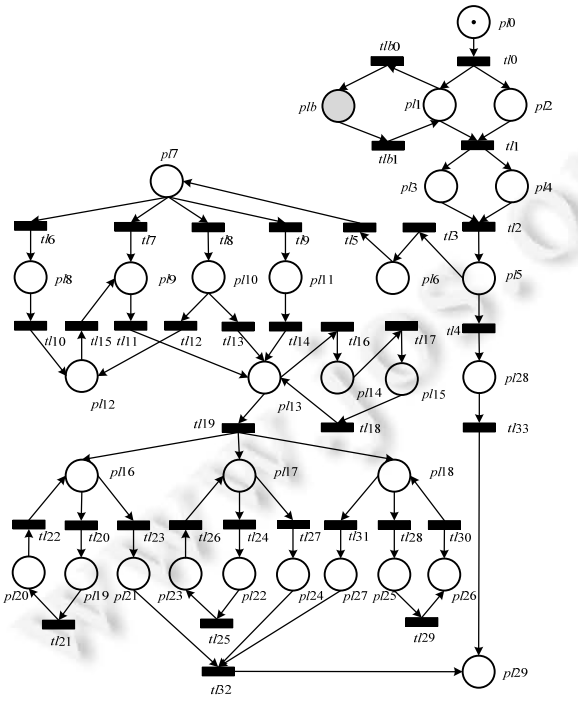
典型的交叉路口场景，基于区域的划分，可以分为接近路口、到达路口、驶入路口、经过路口和驶出路口这 5 个阶段，它们与交通参与者等场景要素的交互行为过程描述如下.

- 接近路口：当测试车辆在接近路口路段行驶并接收指令向左转弯时，识别响应交通标线和交通标志，实现变更车道，驶入左转导向车道. 当测试车辆到达距离路口[100,30] m 路段行驶时，开启左转向灯，并调整车速至 30 km/h 以下. 当测试车辆到达距离路口[30,0] m 路段行驶时，检测有无交警指挥，如果有交警指挥，则按照交警指挥手势通过路口；否则，按照下述过程建模场景.
- 到达路口：测试车辆减速慢行并检测交通信号灯. 如果路口红灯，则测试车辆在停止线前停车等候；如果路口黄灯且测试车辆未驶过停止线，则测试车辆在停止线前停车等候；如果路口黄灯且测试车辆已经驶过停止线，则测试车辆减速慢行；如果路口绿灯或者无交通信号灯，则测试车辆减速慢行；但是，当测试车辆减速慢行预计驶入路口时，如果人行横道有行人通过，则测试车辆停车礼让行人.
- 驶入路口：当测试车辆到达路口时，如果路口是绿灯，且人行横道上没有行人通过，则测试车辆减速慢行驶入路口；如果路口黄灯且测试车辆已经越过停止线，且人行横道上没有行人通过，则测试车辆减速慢行驶入路口；如果路口无交通信号灯，且人行横道上没有行人通过，则测试车辆减速慢行驶入

路口.

- 经过路口: 测试车辆驶入路口后, 如果有右侧车辆通过路口, 则测试车辆让行右侧车辆; 如果有直行车辆通过路口, 则测试车辆让行直行车辆; 如果有行人通过路口, 则测试车辆礼让行人; 否则, 测试车辆在路口行驶.
- 驶出路口: 测试车辆经过路口后, 到达驶离车道, 则测试车辆驶出路口, 在离开路口路段行驶.

图 4 显示了交叉路口左转测试场景模型, 依据早期的道路交通管理条例, 将接近路口阶段划分为距离路口大于 100 m、[100,30] m 和[30,0] m 的这 3 个路段. 库所描述的是测试车辆的状态, 其中, *p*l0 是左转场景模型的初始状态, 含有一个标识. 变迁表示由标识触发的从初始状态到最终状态的多个事件, 描述的是两个状态转换的动作, 变迁的发生使得标识在库所之间转移.



库所	含义
pl0	接近路口路段行驶
pl1	识别响应交通标线
pl2	识别响应交通标志
pl3	开启左转向灯
pl4	调整车速 30km/h 以下
pl5	检测有无交警指挥
pl6	慢行检测交通信号灯
pl7	停止线前减速慢行
pl8	路口红灯
pl9	路口绿灯
pl10	路口黄灯
pl11	路口无信号灯
pl12	停车等待
pl13	减速慢行
pl14	行人通过人行横道
pl15	礼让行人
pl16	慢行检测右侧车辆
pl17	慢行检测直行车辆
pl18	慢行检测路口行人
pl19	右侧有车
pl20	让行右侧车辆
pl21	路口行驶
pl22	直行有车
pl23	避让直行车辆
pl24	路口行驶
pl25	路口有人
pl26	礼让路口行人
pl27	路口行驶
pl28	按照交警指挥手势行驶
pl29	离开路口路段行驶
plb	向左变更车道
tlb0	发送指令向左变更车道
tlb1	接收指令完成车道变更

变迁	含义
tl0	接收指令向左转弯
tl1	到达[100,30]米路段
tl2	到达[30,0]米路段
tl3	无交警指挥
tl4	有交警指挥
tl5	到达路口
tl6	识别红灯
tl7	识别绿灯
tl8	识别黄灯
tl9	识别无灯
tl10	响应红灯
tl11	响应绿灯
tl12	响应未过停止线
tl13	响应已过停止线
tl14	响应无灯
tl15	变为绿灯
tl16	识别行人通过人行横道
tl17	响应行人通过人行横道
tl18	自适应调整车速
tl19	驶入路口
tl20	识别右侧车辆
tl21	响应右侧车辆
tl22	自适应调整车速
tl23	识别响应右侧无车
tl24	识别直行车辆
tl25	响应直行车辆
tl26	自适应调整车速
tl27	识别响应直行无车
tl28	识别路口行人
tl29	响应路口行人
tl30	自适应调整车速
tl31	识别响应路口无人
tl32	驶出路口
tl33	交警指挥驶出路口

图 4 路口左转测试场景模型

测试车辆在收到指令交叉路口向左转弯时, 需要识别响应交通标线是否为左转车道: 如果不是左转车道, 则需要变更至左转车道. 而变更车道属于另外一个交通场景, 所以在模型中要设置相应的调用变更车道场景的接口. 在路口左转场景模型中, *tlb*0 表示发送向左变更车道的指令, *tlb*1 表示接收完成向左变更车道的指令, *plb* 表示测试车辆向左变更车道场景模型. 在这里, 只需把 *plb* 转换为向左变更车道场景即可.

4 模型验证方法

随着系统规模的扩大和复杂度的增加, 模型的安全性直接关系到软件的质量, 亟需自动化的精确分析方法对其验证. 如何验证描述系统需求和模型设计的一致性, 已经成为研究人员关注的一个重要问题, 绝大多数安全标准推荐使用形式化分析验证方法. 形式化方法是基于严谨的逻辑语言和精确的数学语义的方法学, 主要应用于对系统软件进行建模分析, 从而把验证一个系统是否满足它的设计需求问题等价为验证模型是否满足相应的逻辑公式的问题, 而验证模型是否满足逻辑公式则可以通过模型检测器自动化完成[21]. 本文通过将 Petri 网模型的语义转换为 CCSL 约束, 一个可进行形式化验证的中间语义模型, 基于 CCSL 约束的可满足性和 LTL 公式来验证系统需求和模型设计的一致性.

4.1 从Petri网到CCSL约束的转换

通过对场景模型中体现交规属性的库所和变迁的时序逻辑关系描述, 映射基于交规场景模型的 CCSL 语义约束. 实质上, CCSL 约束描述的是状态变化的时序逻辑关系, 这里的状态由 Petri 网中的库所描述. 依据 Petri 网的结构关系, 首先给出状态持续时间的概念, 然后从以下情况来考虑转换 CCSL 约束的规则, 包括顺序关系、同步关系、并发关系、选择关系和循环关系.

(1) 状态持续时间

在模型中, 状态是一个持续的过程, 需要的时间相对较长, 这里假设状态持续时间是有限的. 事实上, 每个状态都可以表示为开始事件、结束事件和持续时间. 例如, 在交叉路口场景中, 当交通信号灯响应为红灯, 测试车辆需在停止线前停车等待, 直到交通信号灯响应为绿灯时, 测试车辆才可以通行. 依据本条交规构建交叉路口场景模型, 如果开始事件 *event*1 的交规属性为响应红灯, 结束事件 *event*2 的交规属性为响应绿灯, 则状态 *state* 的交规属性为停车等待. 容易理解, 停车等待这个状态需要一定的持续时间. 在这种情况下, 状态持续时间定义为:

$$state.t=event2.t_2-event1.t_1,$$
$$\text{s.t.}\quad t_1<t_2.$$

状态的持续时间是通过状态的开始事件和结束事件之间的时序逻辑关系进行定义的, 在 Petri 网模型中, 状态持续时间的描述如图 5 所示.

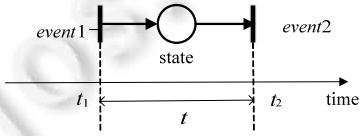
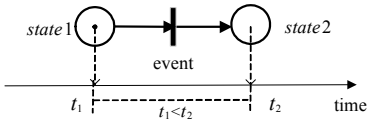


图 5 状态持续时间

(2) 顺序关系

顺序关系指的是状态的一个线性的时序逻辑关系. 例如, 在交叉路口场景中, 当交通信号灯为红灯时, 测试车辆应该能够识别红灯, 并且在停止线前停车等待. 根据交规进行语义分析, 基本事件 *event* 的交规属性为识别红灯, *event* 的前置条件状态 *state*1 的交规属性为交通信号灯红灯, *event* 的后置条件状态 *state*2 的交规属性为停止线前停车等待. 在这种情况下, *state*1 和 *state*2 为顺序关系.

在 Petri 网模型中, 顺序关系是通过事件的前置条件状态和后置条件状态之间的时序逻辑关系进行定义的, 如图 6(a)所示. 顺序关系描述的时序逻辑是两个状态的先后关系, 其对应的 CCSL 约束如图 6(b)所示.



(a) 顺序关系图

CCSL:

1. *state*1_*attribute*≺*state*2_*attribute*

(b) CCSL 约束

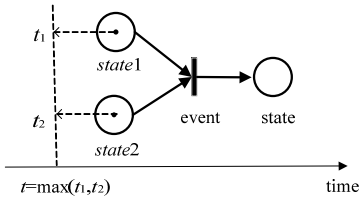
图 6 状态顺序关系转换

(3) 同步关系

同步关系指的是两个或两个以上的状态必须同步出现才能引发后续事件. 例如, 在交叉路口场景中, 当测试车辆在达到路口[100,30] m 路段前行驶时, 需要同步完成交通标线和交通标志的识别响应. 根据交规进行语义分析, 状态 *state*1 交规属性为识别响应交通标线, 状态 *state*2 交规属性为识别响应交通标志, 只有 *state*1 和 *state*2 同步满足, 才会引发后续事件 *event* 交规属性为达到路口[100,30] m 路段. 在这种情况下, *state*1 和 *state*2 为同步关系.

在 Petri 网模型中, 同步关系是通过同一事件的两个前置条件状态之间的时序逻辑关系进行定义的, 如图

7(a)所示. 同步关系描述的时序逻辑是两个状态的逻辑与关系, 最迟出现的状态应该先于下一个状态, 其对应的 CCSL 约束如图 7(b)所示.



(a) 同步关系图

CCSL:

1. *state*1_*attribute*≺*state*_*attribute*
2. *state*2_*attribute*≺*state*_*attribute*
3. *temp*=*state*1_*attribute*∨*state*2_*attribute*
4. *temp*≺*state*_*attribute*

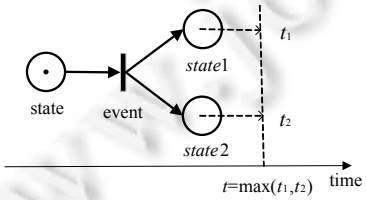
(b) CCSL 约束

图 7 状态同步关系转换

(4) 并发关系

并发关系是指两个状态的独立出现, 相互之间没有影响. 例如, 在交叉路口场景中, 当测试车辆接收指令向左转弯时, 测试车辆需要识别响应交通标线和交通标志, 而这两个状态可以独立存在. 根据交规进行语义分析, 事件 *event* 的交规属性为接收指令向左转弯, 状态 *state*1 交规属性为识别响应交通标线, 状态 *state*2 交规属性为识别响应交通标志. 在这种情况下, *state*1 和 *state*2 为并发关系.

在 Petri 网模型中, 并发关系是通过同一事件的两个后置条件状态之间的时序逻辑关系进行定义的, 如图 8(a)所示. 并发关系描述的时序逻辑也是两个状态的逻辑与关系, 但最早出现的状态应该后于上一个状态, 其对应的 CCSL 约束如图 8(b)所示.



(a) 并发关系图

CCSL:

1. *state*_*attribute*≺*state*1_*attribute*
2. *state*_*attribute*≺*state*2_*attribute*
3. *temp*=*state*1_*attribute*∧*state*2_*attribute*
4. *state*_*attribute*≺*temp*

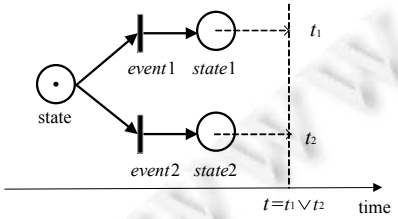
(b) CCSL 约束

图 8 状态并发关系转换

(5) 选择关系

选择关系指的是两个或者两个以上状态的触发事件共享一个状态资源, 它们可以发生, 但有且仅能有一个事件发生. 例如, 在交叉路口场景中, 当交通信号灯为黄灯时, 测试车辆要么已过停止线, 要么未过停止线, 只有一个事件发生. 根据交规进行语义分析, 状态 *state* 交规属性为交通信号灯为黄灯, 事件 *event*1 交规属性为已过停止线, 则状态 *state*1 交规属性为减速慢行; 而事件 *event*2 交规属性为未过停止线, 则状态 *state*2 交规属性为停车等待. 在这种情况下, *event*1 和 *event*2 为冲突事件, 即 *state*1 和 *state*2 为选择关系.

在 Petri 网模型中, 选择关系是通过冲突事件的两个后置条件状态的时序逻辑关系进行定义的, 如图 9(a)所示. 选择关系描述的时序逻辑是两个状态的排斥或关系, 其对应的 CCSL 约束如图 9(b)所示.



(a) 选择关系图

CCSL:

1. *state*_*attribute*≺*state*1_*attribute*
2. *state*_*attribute*≺*state*2_*attribute*
3. *state*_*attribute*≺*temp*
4. *temp*=*state*1_*attribute*+*state*2_*attribute*
5. *state*1_*attribute*#*state*2_*attribute*

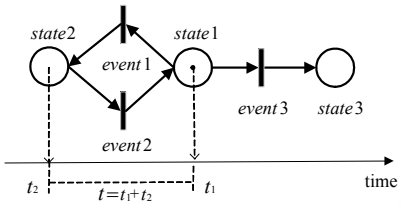
(b) CCSL 约束

图 9 状态选择关系转换

(6) 循环关系

循环关系指的是两个或两个以上状态的触发事件依次交替发生, 引发其后置条件状态的交替转换. 例如, 在交叉路口场景中, 当测试车辆路口行驶时, 如果检测到有行人横穿路口, 则测试车辆就要礼让行人, 直到路口无人, 则测试车辆继续路口行驶. 根据交规进行语义分析, 状态 *state*1 交规属性为慢行检测路口行人, 事件 *event*1 交规属性为识别响应路口行人, 状态 *state*2 交规属性为路口礼让行人, 事件 *event*2 交规属性为自适应调整车速, 直到事件 *event*3 交规属性为路口无人, 则 *state*3 交规属性为路口行驶. 在这种情况下, *state*1 和 *state*2 为循环关系.

在 Petri 网模型中, 循环关系是通过两个不断转换的状态之间的时序逻辑关系进行定义的, 如图 10(a)所示. 循环关系描述的时序逻辑是两个或者两个以上状态依次出现的交替关系, 为了降低场景模型复杂度以及防止状态空间爆炸, 这里设置循环的限制次数为 *d*, 其对应的 CCSL 约束如图 10(b)所示.



(a) 循环关系图

CCSL:

1. *state*1_*attribute*≺*state*2_*attribute*
2. *state*1_*attribute*≺*state*3_*attribute*
3. *state*3_*attribute*=*state*1_*attribute* \$ *d* on *state*2_*attribute*

(b) CCSL 约束

图 10 状态循环关系转换

说明: 开始状态和结束状态除外, 其余状态均有触发事件和终止事件, 所有状态和事件之间都存在优先关系. 由于优先关系是传递的, 所以没有必要把它们都写下来. 我们可以使用深度优先搜索算法来遍历所有的状态和事件, 并将其转换为 CCSL 约束.

4.2 基于LTL公式描述模型的属性

线性时序逻辑(linear temporal logic, LTL)模型检测就是验证由 CCSL 约束集描述的模型是否满足某些系统属性, 这些属性可以通过 LTL 公式进行描述, 本文将其扩展到 CCSL 的调度问题上, 进而实现模型验证. LTL 是命题逻辑的延伸, 适用于推理无限状态序列. LTL 公式是由原子命题、布尔操作符和时态操作符构成.

LTL 公式的语法定义如下:

$$\left.\begin{array}{l}\varphi ::= \top \mid \bot \mid ap \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \to \varphi \\ X\varphi \mid F\varphi \mid G\varphi \mid \varphi U\varphi \mid \varphi W\varphi \mid \varphi R\varphi\end{array}\right\} \tag{4}$$

其中, *ap* 为原子命题, ⊤表示 true, ⊥表示 false, 时态操作符 *X*, *F*, *G*, *U*, *W* 和 *R* 分别表示下一个状态(next)、未来某个状态(future)、所有将来的状态(globally)、直到(until)、弱直到(weak-until)和释放(release).

使用 LTL 公式描述系统行为具有的属性, 其原子命题集合对应时钟集合, 即一个原子命题 *ap* 在状态序列的第 *i* 步为真当且仅当时钟 *c* 在调度的第 *i* 步滴答. 因此, LTL 公式的语义可以表述在 CCSL 的调度上, 描述的就是时钟行为具有的一些属性.

考虑一个简单的例子, 使用 CCSL 约束指定交叉路口绿灯和红灯的交替闪烁. 假设交通信号灯首先亮起的是绿灯, 则可以通过下面的 CCSL 约束来进行形式化表示.

(1) *green*≺*red*.

(2) *temp*=*green* \$ 1.

(3) *red*≺*temp*.

其中, 两个时钟 *green* 和 *red* 分别代表绿灯和红灯, *green*(*red*)滴答表示绿灯(红灯)亮; 时钟 *temp* 是一个虚拟时钟, 它不代表任何真实事件, 仅用于帮助指定时钟的约束. 那么, 可以使用 LTL 模型检测的方法验证系统是否具有下面的属性.

属性 I. 某个方向上的交通信号灯不能同时为绿灯和红灯, 可以用 LTL 公式表示:

$$G(\neg(green \wedge red)).$$

属性 II. 当交通信号灯为绿灯时, 其下一个时刻不再是绿灯, 可以用 LTL 公式表示:

$$G(green \rightarrow X(\neg green)).$$

5 实验评估

在本节中, 对基于交规构建的自动驾驶测试场景模型进行评估, 研究的问题主要考虑两个方面.

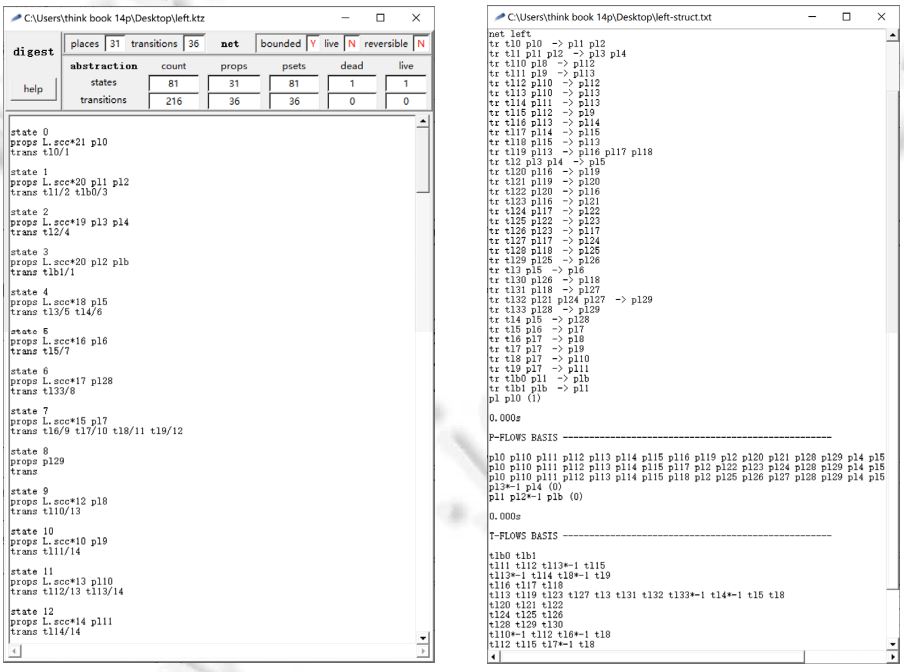
- RQ1: 模型的正确性如何?
- RQ2: 模型的一致性如何?

实验环境及配置, CPU AMD Ryzen 7 5800H CPU@3.20 GHz 和 Windows 10 (64 位).

5.1 模型的正确性

为了验证模型的正确性, 根据 Petri 网理论中对模型动态特性进行分析的方法, 对路口左转场景模型的活性、有界性、可达性进行验证. 这样, 一方面可以科学地反映出模型的特性; 另一方面, 又可以在分析验证的过程中发现潜在的问题并及时修改, 以保证模型的正确性. 简单来说, 活性是指检查所有的活动是否都能被执行; 有界性是指检查一个库所中是否发生了标识堆积的现象; 可达性是指标识能否从一个库所到达另一个库所.

基于 Petri 网的交规场景模型, 采用 Tina 软件进行动态分析, 下载地址为 https://projects.laas.fr/tina/home.php. Tina 是一款功能强大的对 Petri 网支持得比较好的分析工具, 它可以对模型的有界性、活性和可达性进行分析, 给出相关的分析报告. 此外, Tina 还有专门的模拟模块, 它可以切换到运行模拟器中, 模拟完成后, 还可以返回到初始状态进行单步模拟分析. 因此, 本文采用 Tina 软件, 对路口左转场景的 Petri 网模型仿真, 模型分析验证结果如图 11 所示.



(a) 状态空间分析 (b) 结构分析

图 11 路口左转场景模型分析验证结果

(1) 有界性分析

对于一个 Petri 网$\Sigma=(P,T;F,M_0)$, 如果存在一个非负实数 B, 使得对于库所 $p_i\in P$, 能够满足$\forall M\in R(M_0)$, $M(p_i)\leqslant B$, 则称库所 p_i 是有界的, 且称满足此条件的最小 B 值为库所 p_i 的界, 记为 $B(p_i)$, 即:

$$B(p_i)=\min\{B|\forall M\in R(M_0):M(p_i)\leqslant B\} \tag{5}$$

如果对模型中每一个库所 p 都存在一个非负实数 B, 使得$\forall M\in R(M_0)$, $M(p)\leqslant B$, 则称该模型为有界 Petri 网, 且称最大的那个界值为 Petri 网的界. Petri 网的有界性从理论层面确保了系统不会出现缓冲区溢出的问题, 从而保证了系统不会在某个操作阶段发生堆积.

本文通过 Tina 软件的“state space analysis”模块对基于 Petri 网的左转场景模型进行仿真, 得到图 11(a)的状态空间分析结果. 由此可以看出, 路口左转场景模型包含 31 个库所和 36 个变迁, bounded 属性为“Y”, 也就是说, 该模型通过了仿真软件的有界性检测. 通过 Tina 软件的“structural analysis”模块对 Petri 网模型进行仿真, 得到图 11(b)的结构分析结果. 由此可以看出每个变迁的前置库所和后置库所, 以及模型中流的相关信息.

(2) 活性分析

对于一个 Petri 网$\Sigma=(P,T;F,M_0)$, M_0 为初始标识, $t\in T$. 如果$\forall M\in R(M_0)$, $\exists M'\in R(M)$,使得 $M'[t\rangle$, 则称变迁 t 是活的. 如果对所有 $t\in T$ 都是活的, 则称Σ具有活性. Petri 网的活性从理论层面保证了系统不会发生死锁, 从而保证了系统可以连续正常地运行.

本文为测试车辆通过交叉路口向左转弯测试场景建立系统模型, 该模型属于单向结构, 而非循环结构, 所以 Petri 网不具有活性. 从图 11 的结果可知, 模型具有一个“live”状态和一个“dead”状态, 分别对应模型的初始状态和结束状态. 如果在这两个状态之间添加一个变迁, 使得模型从单向结构转化为循环结构, 则此时的模型不仅具有活性, 而且具有可逆性. 由此可以得出结论: 向左转弯场景模型不会发生死锁,系统可以连续正常地运行.

(3) 可达性分析

对于一个 Petri 网$\Sigma=(P,T;F,M_0)$, $t\in T$. 对于初始标识 M_0, 如果 $M'\in R(M_0)$, 使 $M_0[t\rangle M'$, 则称标识 M'为从 M_0 直接可达. 如果存在变迁序列 $t_1,t_2,\ldots,t_k$ 和标识序列 $M_1,M_2,\ldots,M_k$, 使得 $M_0[t_1\rangle M_1[t_2\rangle\ldots M_{k1}[t_k\rangle M_k$, 则称 M_k 为从 M_0 可达. 即模型从初始标识 M_0 经过有限次变迁变为 M_k, 则称标识 M_k 是可达的. Petri 网的可达性从理论层面确保了系统的正确性, 通过分析模型的可达状态, 可以找到模型逻辑上的错误.

本文通过 Tina 软件的“reachability checking”模块对模型进行仿真, 得到的可达性检测结果与状态空间分析结果一致. 图 11 结果显示: Petri 网模型的所有库所均具有可达性, 所有变迁均具有可触发性. 由此可知, 路口左转场景的 Petri 网模型具有可达性.

根据对模型的有界性、活性和可达性的动态属性分析, 验证了路口左转场景模型在逻辑上和结构上的正确性, 表明了所建模型是有效的.

5.2 模型的一致性

为了验证模型的一致性, 基于 CCSL 约束的形式化分析方法, 通过 LTL 公式对模型设计和系统需求的一致性进行分析和验证. 本文以交叉路口左转场景模型为例, 将其分为 3 个独立的子模块进行形式化的分析和验证, 即接近路口模块、交通信号灯模块和路口行驶模块. 最后, 将 3 个子模块的 CCSL 约束组合, 即可得到路口左转场景模型的完整 CCSL 描述.

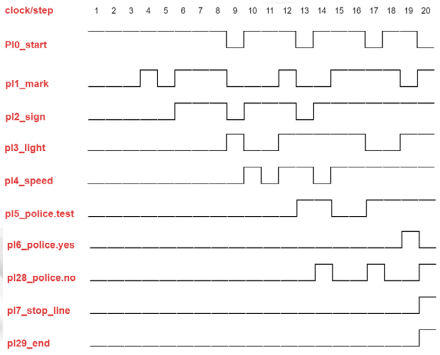
基于 Petri 网的交规场景模型采用 MyCCSL 工具进行分析验证, 下载地址为 https://github.com/northcity0406/CCSLSMT. MyCCSL 为 CCSL 的形式化分析工具, 通过将 CCSL 约束及其相应方法转换为 SMT 公式, 然后依托 SMT 求解器进行求解, 可以有效地避免状态爆炸问题. 对于图 4 的路口左转场景的 Petri 网模型, 将模型中的每个状态映射为一个逻辑时钟, 从而用 CCSL 约束描述系统需求. 由于路口左转整网模型为非循环结构, 时钟的滴答即代表状态的发生, 而时钟的持续滴答可以被认为是自动驾驶汽车多次通过路口的行为. 下面通过对 3 个子模块的描述, 得到完整的路口左转场景模型的 CCSL 描述, 从而对 Petri 网模型进行 LTL 模型检验, 并给出分析验证结果.

(1) 接近路口模块

在接近路口模块中, 当测试车辆收到指令向左转弯时, 同步识别响应交通标线和交通标志. 当测试车辆到达距离路口[100,30] m 路段时, 同步开启左转向灯和调整车速至 30 km/h 以下. 当测试车辆到达距离路口[30,0] m 路段时, 检测有无交警指挥: 如果有交警指挥, 则按照交警指挥手势行驶, 直到离开路口; 如果无交警指挥, 则停止线前慢行. 基于上述过程的 Petri 网模型以及从 Petri 网到 CCSL 约束的转换规则, 得到接近路口模块的 CCSL 描述如图 12(a)所示. 对于一个 CCSL 约束集, 如果存在一个调度满足其所有约束, 才能说明该约束集是可满足的. 利用 MyCCSL 工具对接近路口模块的 CCSL 约束集进行分析, 返回结果为 sat, 说明该约束集是可满足的, 其时钟图结果如图 12(b)所示.

1. *pl0_start* < *pl1_mark* ∧ *pl2_sign*
2. *pl1_mark* ∨ *pl2_sign* < *pl3_light* ∧ *pl4_speed*
3. *pl3_light* ∨ *pl4_speed* < *pl5_police.test*
4. *pl5_police.test* < *pl6_police.yes* + *pl28_police.no*
5. *pl6_police.yes* # *pl28_police.no*
6. *pl6_police.yes* < *pl7_stop_line*
7. *pl28_police.no* < *pl29_end*

(a) CCSL 约束集



(b) 时钟图

图 12 接近路口模块的 CCSL 约束集及其时钟图

根据交通法规提取接近路口模块的系统需求, 结合 CCSL 约束集确定系统行为应该满足的交规属性, 并将系统属性形式化的表示为 LTL 公式进行模型验证. 在接近路口模块中, 系统行为满足的典型交规属性可描述为“有交警指挥和无交警指挥不能同时出现”, LTL 公式表示如下:

$$G(\neg((pl6\_police.yes)\wedge(pl28\_police.no))).$$

为了检验模型设计与系统需求的一致性, 将图 12 所示的 CCSL 约束以及描述属性的 LTL 公式放入 MyCCSL 工具, 并使用 Z3 来检验生成的 SMT 公式的可满足性. 在不同边界范围内重复模型检查, Z3 总是返回具有属性的 unsat 结果. 这意味着模型设计满足了本条属性, 通过了系统需求的一致性验证.

表 2 显示了对不同边界进行模型属性验证的结果和时间. 结果表明, 随着边界的增加, 模型检验的时间会增加. 当边界不大于 100 时, 验证在几分钟内都可以完成, 具有较好的效率.

表 2 接近路口模块 LTL 模型属性检测结果

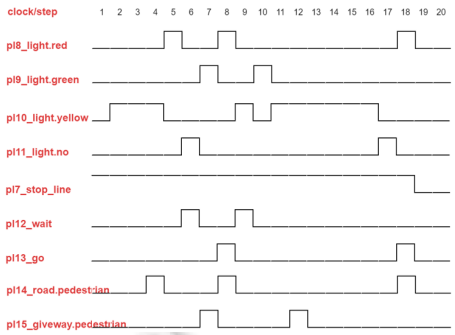
边界	$G(\neg((pl6\_police.yes)\wedge(pl28\_police.no)))$	
	结果	时间(s)
20	unsat	1.38
40	unsat	6.16
60	unsat	21.60
80	unsat	47.36
100	unsat	95.62

(2) 交通信号灯模块

在交通信号灯模块中, 测试车辆在停止线前减速慢行, 如果无交通信号灯, 则减速慢行; 如果红灯, 则停车等待, 直到变为绿灯; 如果绿灯, 则减速慢行; 如果黄灯且已越过停止线, 则减速慢行; 如果黄灯且未越过停止线, 则停车等待. 当测试车辆减速慢行时, 如果行人横穿人行横道, 则礼让行人后再驶入路口. 基于 Petri 网模型以及从 Petri 网到 CCSL 约束的转换规则, 得到交通信号灯模块的 CCSL 描述如图 13(a)所示. 利用 MyCCSL 工具对交通信号灯模块的 CCSL 约束集进行分析, 返回结果为 sat, 说明该约束集是可满足的, 其时钟图结果如图 13(b)所示.

1. *pl*7_*stop_line* < *pl*8_*light.red* + *pl*9_*light.green* + *pl*10_*light.yellow* + *pl*11_*light.no*
2. *pl*8_*light.red* # *pl*9_*light.green* # *pl*10_*light.yellow* # *pl*11_*light.no*
3. *pl*8_*light.red* < *pl*12_*wait*
4. *pl*12_*wait* < *pl*9_*light.green*
5. *pl*9_*light.green* < *pl*13_*go*
6. *pl*10_*light.yellow* < *pl*12_*wait* + *pl*13_*go*
7. *pl*12_*wait* # *pl*13_*go*
8. *pl*11_*light.no* < *pl*13_*go*
9. *pl*14_*road.pedestrian* < *pl*15_*giveway.pedestrian*
10. *pl*13_*go* = *pl*14_*road.pedestrian* $ 1
11. *pl*15_*giveway.pedestrian* < *pl*13_*go*

(a) CCSL 约束集



(b) 时钟图

图 13 交通信号灯模块的 CCSL 约束集及其时钟图

在交通信号灯模块中, 通过 LTL 公式验证系统行为满足的交规属性. 例如, 红灯和绿灯两种信号灯状态不能同时出现; 红灯亮时测试车辆应该在停止线前停车等待, 而不是减速慢行驶入路口. 基于 MyCCSL 软件中用来验证模型属性的 LTL 公式定义, 本文将形式化的 LTL 公式描述为否定表达式. 表 3 显示了交通信号灯模型应该具有的属性描述、LTL 公式表示和验证结果, 其验证过程与接近路口模块相同, 不再赘述.

表 3 交通信号灯模块的 LTL 模型检测结果

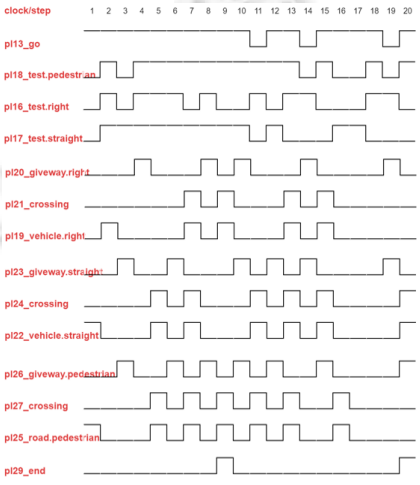
	属性描述	LTL 公式	结果
1	红灯和绿灯不能同时出现	$G(\neg((pl8\_light.red)\wedge(pl9\_light.green)))$	unsat
2	红灯和黄灯不能同时出现	$G(\neg((pl8\_light.red)\wedge(pl10\_light.yellow)))$	unsat
3	绿灯和黄灯不能同时出现	$G(\neg((pl9\_light.green)\wedge(pl10\_light.yellow)))$	unsat
4	如果红灯, 则停车等待(非驶入路口)	$G((pl8\_light.red)\rightarrow(X(\neg(pl13\_go))))$	unsat
5	如果行人通过人行横道, 则礼让行人(非驶入路口)	$G((pl14\_road.pedestrian)\rightarrow(X(\neg(pl13\_go))))$	unsat

(3) 路口行驶模块

在路口行驶模块中, 当测试车辆驶入路口时, 同步检测右侧车辆、直行车辆和路口行人. 如果右侧有车通行, 则让行右侧车辆; 如果直行有车通行, 则避让直行车辆; 如果路口有人, 则礼让行人; 否则, 路口行驶, 直到离开路口. 基于上述过程的 Petri 网模型以及从 Petri 网到 CCSL 约束的转换规则, 得到路口行驶模块的 CCSL 描述如图 14(a)所示. 利用 MyCCSL 工具对路口行驶模块的 CCSL 约束集进行分析, 返回结果为 sat, 说明该约束集是可满足的, 其时钟图结果如图 14(b)所示.

1. *pl*13_*go* < *pl*16_*test.right* ∧ *pl*17_*test.straight* ∧ *pl*18_*test.pedestrian*
2. *pl*16_*test.right* < *pl*20_*giveway.right* + *pl*21_*crossing*
3. *Pl*20_*giveway.right* # *pl*21_*crossing*
4. *pl*19_*vehicle.right* < *pl*20_*giveway.right*
5. *pl*21_*crossing* = *pl*19_*vehicle.right* $ 1
6. *Pl*20_*giveway.right* < *pl*21_*crossing*
7. *pl*17_*test.straight* < *pl*23_*giveway.straight* + *pl*24_*crossing*
8. *pl*23_*giveway.straight* # *pl*24_*crossing*
9. *pl*22_*vehicle.straight* < *pl*23_*giveway.straight*
10. *pl*24_*crossing*= *pl*22_*vehicle.straight* $ 1
11. *pl*23_*giveway.straight* < *pl*24_*crossing*
12. *pl*18_*test.pedestrian* < *pl*26_*giveway.pedestrian* + *pl*27_*crossing*
13. *pl*26_*giveway.pedestrian* # *pl*27_*crossing*
14. *pl*25_*road.pedestrian* < *pl*26_*giveway.pedestrian*
15. *pl*27_*crossing* = *pl*25_*road.pedestrian* $ 1
16. *pl*26_*giveway.pedestrian* < *pl*27_*crossing*
17. *pl*21_*crossing* ∨ *pl*24_*crossing* ∨ *pl*27_*crossing* < *pl*29_*end*

(a) CCSL 约束集



(b) 时钟图

图 14 路口行驶模块的 CCSL 约束集及其时钟图

在路口行驶模块中, 基于 CCSL 约束集和 LTL 公式验证系统行为满足的交规属性. 表 4 显示了路口行驶模型应该具有的属性描述、LTL 公式表示和验证结果, 其验证过程与接近路口模块相同, 不再赘述.

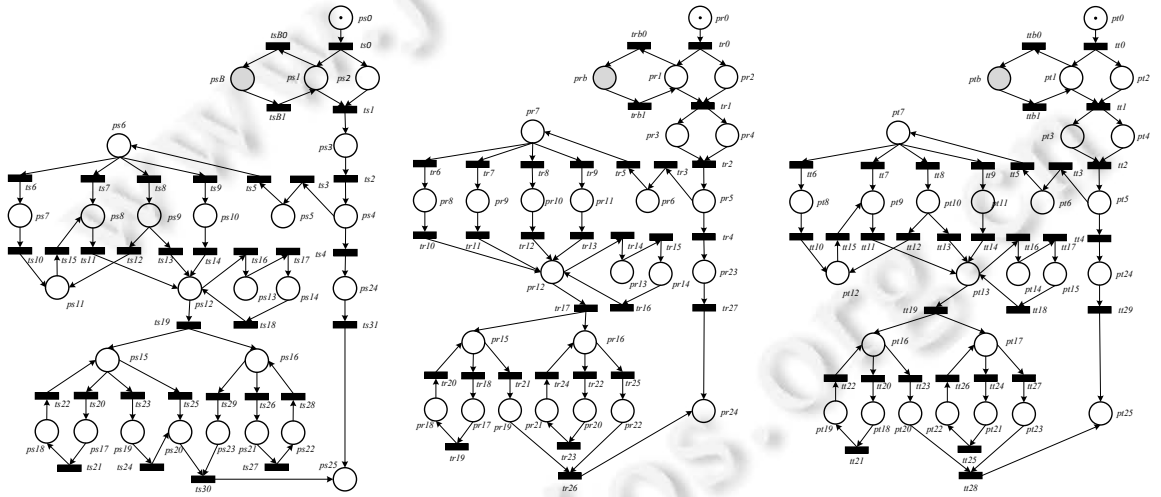
表 4 路口行驶模块的 LTL 模型检测结果

	属性描述	LTL 公式	结果
1	让行右侧车辆和路口行驶不能同时出现	$G(\neg((pl20\_giveway.right)\wedge(pl21\_crossing)))$	unsat
2	避让直行车辆和路口行驶不能同时出现	$G(\neg((pl23\_giveway.straight)\wedge(pl24\_crossing)))$	unsat
3	礼让行人和路口行驶不能同时出现	$G(\neg((pl26\_giveway.pedestrian)\wedge(pl27\_crossing)))$	unsat
4	如果右侧有车, 则让行右侧车辆(非路口行驶)	$G((pl19\_vehicle.right)\rightarrow(X(\neg(pl21\_crossing))))$	unsat
5	如果直行有车, 则避让直行车辆(非路口行驶)	$G((pl22\_vehicle.straight)\rightarrow(X(\neg(pl24\_crossing))))$	unsat
6	如果路口有人, 则礼让行人(非路口行驶)	$G((pl25\_road.pedestrian)\rightarrow(X(\neg(pl27\_crossing))))$	unsat

通过上述子模块 CCSL 约束集组合得到路口左转场景模型的完整 CCSL 约束集, 并在完整模型系统上进行死锁验证和可满足性验证, 验证结果在后文表5所示的交叉路口模型验证结果中进行了描述. 同时, 选择代表性属性的 LTL 公式在左转场景整网模型上进行系统需求和模型设计的一致性验证, 结果与路口行驶模块的 LTL 模型检测结果相同. 左转场景综合结果表明, 路口左转场景模型的 CCSL 约束集在不同边界下均具有可满足性, 并且无死锁. 此外, 通过 LTL 公式的形式化方法验证了模型设计与系统需求的一致性.

5.3 交叉路口场景建模与分析

实际上, 交叉路口测试场景模型可分为左转场景、直行场景、右转场景和掉头场景这 4 个子场景模型. 基于上述左转场景建模方法以及第 3.2 节的场景布设, 容易构建如图 15 所示的直行、右转和掉头这 3 个子场景模型. 由图 15 可见, 交叉路口各个子场景模型之间有许多相同的交叉子模块. 对于模型中的库所和变迁的含义, 这里也不再赘述.



(a) 直行 (b) 右转 (c) 掉头

图 15 交叉路口测试场景模型

基于第 5.1 节所述方法, 采用 Tina 软件对交叉路口各个子场景模型进行仿真分析, 所得结果显示, 每个子模型均具有活性、有界性和可达性, 说明所构建的交叉路口模型是正确的. 基于第 5.2 节所述方法, 使用 MyCCSL 工具在不同边界上对上述模型进行形式化分析验证, 可知每个模型都无死锁且都是可调度的, 其分析验证结果如表 5 所示. 从表 5 数据还可以看出, 随着边界的增大, 求解时间明显增加. 原因是系统模型复杂度的增大, 导致了模型验证时间随之增大, 但在有效时间内, 均能得到模型的正确验证结果, 说明本文方法是可行而有效的.

表 5 交叉路口模型验证结果

	活性	有界性	可达性	边界		30		40		50	
				场景		结果	时间(s)	结果	时间(s)	结果	时间(s)
交叉路口场景模型	√	√	√	左转场景	可满足性	sat	51.65	sat	247.78	sat	820.61
					死锁	unsat	77.67	unsat	293.31	unsat	1 096.57
				直行场景	可满足性	sat	14.57	sat	94.08	sat	305.15
					死锁	unsat	25.13	unsat	173.52	unsat	243.1
				右转场景	可满足性	sat	42.12	sat	193.00	sat	509.73
					死锁	unsat	52.15	unsat	205.80	unsat	428.66
				掉头场景	可满足性	sat	22.50	sat	121.70	sat	213.81
					死锁	unsat	48.82	unsat	165.35	unsat	347.40

通过 LTL 公式对模型检测, 验证模型设计与系统需求的一致性. 由于第 5.2 节已经详细介绍了左转场景的相关属性验证过程, 而且每个场景交通信号灯模块系统属性相同, 因此, 这里只给出直行、右转和掉头这 3 个场景的验证属性及其相关结果. 表 6 显示了系统模型具有的典型属性描述、LTL 公式表示和验证结果, 实验结果进一步说明了交叉路口场景生成模型能够通过系统需求的一致性验证.

表 6 交叉路口 LTL 模型检测结果

		属性描述	LTL 公式	结果
1	直行	少数让行和路口行驶不能同时出现	$G(\neg((ps18\_giveway.vehicle)\wedge(ps20\_crossing)))$	unsat
2		礼让行人和路口行驶不能同时出现	$G(\neg((ps22\_giveway.pedestrian)\wedge(ps23\_crossing)))$	unsat
3		如果本道车辆少数, 则少数让行	$G((ps17\_vehicle.less)\rightarrow(X(\neg(ps20\_crossing))))$	unsat
4		如果路口有人, 则礼让行人	$G((ps21\_road.pedestrian)\rightarrow(X(\neg(ps23\_crossing))))$	unsat
5	右转	让行车辆和路口行驶不能同时出现	$G(\neg((pr18\_giveway.vehicle)\wedge(pr19\_crossing)))$	unsat
6		礼让行人和路口行驶不能同时出现	$G(\neg((pr21\_giveway.pedestrian)\wedge(pr22\_crossing)))$	unsat
7		如果路口有车, 则让行路口车辆	$G((pr17\_road.vehicle)\rightarrow(X(\neg(pr19\_crossing))))$	unsat
8		如果路口有人, 则礼让行人	$G((pr20\_road.pedestrian)\rightarrow(X(\neg(pr22\_crossing))))$	unsat
9	掉头	让行车辆和路口行驶不能同时出现	$G(\neg((pt19\_giveway.vehicle)\wedge(pt20\_crossing)))$	unsat
10		礼让行人和路口行驶不能同时出现	$G(\neg((pt22\_giveway.pedestrian)\wedge(pt23\_crossing)))$	unsat
11		如果路口有车, 则让行路口车辆	$G(pt18\_road.vehicle)\rightarrow(X(\neg(pt20\_crossing))))$	unsat
12		如果路口有人, 则礼让行人	$G((pt21\_road.pedestrian)\rightarrow(X(\neg(pt23\_crossing))))$	unsat

5.4 场景模型的应用性

在自动驾驶汽车测试中, 由于场景用例的测试输入、测试条件和测试结果明确可控、可重复性强且效率高, 所以针对特定场景生成可执行测试用例来验证自动驾驶车辆的功能是一种有效的方法. 目前, 对于场景用例的测试输入和测试条件的确定, 主要还是依赖于实际交通数据的理论分析, 但是前期的数据采集和分析成本相对较高, 并且获得的数据在交通规则方面存在局限性. 鉴于此, 首先, 研究了基于交通规则的交叉路口测试场景的建模方法, 构建了满足系统安全需求一致性验证的模型; 然后, 研究了基于场景模型的自动驾驶汽车测试用例的进化方法, 自动生成能够在仿真环境中测试自动驾驶汽车可靠性的场景用例.

基于交规构建的测试场景模型, 通过了系统安全需求的一致性验证, 表明该模型能够表征相应的交规属性. 例如, 交叉路口场景模型通过了“如果路口有人, 则礼让行人”的交规属性验证, 意味着该模型能够表征“路口礼让行人”这条交通规则. 后续工作将依据构建的场景模型生成大量的覆盖模型路径的测试用例, 由于生成的测试用例能够覆盖场景模型的所有路径, 也就能够表征场景模型所有的交规属性, 从而实现通过仿真环境中的测试来验证自动驾驶汽车可靠性的目的. 也就是说, 如果自动驾驶汽车通过了基于模型生成的场景用例的测试, 则说明自动驾驶汽车遵守了场景模型表征的交通规则; 相反, 如果自动驾驶汽车发生了安全事故, 则说明自动驾驶汽车存在违反交通规则的可能性, 那么要从故障分析树的角度回溯产生的具体原因.

从自动驾驶汽车是否遵守交通规则的角度出发, 以交叉路口左转场景为例, 阐述本研究所构建的场景模型对后续工作的实际应用价值. 例如, 在如图 4 所示的交叉路口左转场景中, 生成覆盖目标路径的测试数据时, 如果选取的目标路径包含子路径 P_1: $pl18\rightarrow tl28\rightarrow pl25\rightarrow tl29\rightarrow pl26\rightarrow tl30\rightarrow pl18$, 则生成的覆盖该目标路径的测试数据可以验证自动驾驶汽车是否遵守了“路口礼让行人”的交通规则; 如果选取的目标路径包含子路径

P_2: $pl16 \rightarrow tl20 \rightarrow pl19 \rightarrow tl21 \rightarrow pl20 \rightarrow tl22 \rightarrow pl16$, 则生成的测试数据可以验证自动驾驶汽车是否遵守了“路口让行右侧车辆”的交通规则; 如果选取的目标路径包含子路径 P_3: $pl17 \rightarrow tl24 \rightarrow pl252 \rightarrow tl25 \rightarrow pl23 \rightarrow tl26 \rightarrow pl17$, 则生成的测试数据可以验证自动驾驶汽车是否遵守了“路口让行直行车辆”的交通规则. 后续工作采用遗传算法进化生成覆盖场景模型目标路径的大规模测试数据, 并在仿真环境中测试自动驾驶汽车的可靠性. 进一步地, 基于场景模型生成的测试用例集, 采用多目标优化方法搜索最优解集, 对生成的测试用例进行优先级排序, 删除冗余测试用例, 从而实现对测试用例集的优化, 在仿真环境中能够更有效地测试自动驾驶系统的可靠性, 这也是本研究的后续工作.

为了更好地应用基于交规的场景模型在自动驾驶领域开展测试工作, 本工作的研究模型(不局限于交叉路口场景模型, 还包括巡航、跟车、变道、超车等复杂测试场景模型)均可开源供读者参考使用, 相关场景模型的下载链接如下: https://github.com/xiachunyan/test1.

5.5 场景模型的完整性

当前, 虽然自动驾驶的发展引发广泛关注, 但与其相关的技术、实验、政策和法规等尚处于发展阶段, 针对自动驾驶测试场景标准法规的研究并不多. 因此, 本研究可能存在不完整性问题, 下面从两个方面讨论测试场景模型的完整性.

- 一方面, 构建的场景模型交规表征的完整性, 本研究依据中华人民共和国道路交通安全法、智能网联汽车道路测试管理规范和智能网联汽车自动驾驶功能测试规程等国内现有的交通法律法规, 采用人工方式逐条提取适合自动驾驶系统的交通法规, 并融合交叉路口测试场景要素的交互行为, 针对交规重构验证自动驾驶安全性的功能测试场景, 从而尽量保证所构建的场景模型能够较为全面地表征交规属性.
- 另一方面, 验证的场景模型交规检测的完整性, 基于交规构建的交叉路口场景模型, 通过了模型安全需求的一致性验证, 即交叉路口场景模型表征的交规属性均已得到验证. 由于场景模型的每条路径都对应一条交规属性, 所以采用进化方法生成覆盖场景模型全部路径的测试用例, 能够表征模型验证的每条交规属性, 再通过场景用例对自动驾驶系统行为进行测试, 也就能够验证自动驾驶汽车是否违反了模型表征的每条交通规则, 即通过验证的场景模型生成的场景用例能够测试自动驾驶汽车是否遵守了模型表征的所有交通规则. 从上述两方面的分析来看, 基于交规构建的测试场景模型具有较好的完整性.

本文的后续工作研究基于交规建模测试场景的用例生成和优化方法, 进一步在仿真环境中对自动驾驶汽车进行交规属性的完整性验证, 从测试充分性和故障分析的角度考虑, 分析模型可能产生的误报原因是否为模型不完备引发的错误, 以及更深入地考虑异常突发场景的组合和生成.

6 总 结

基于交规建模的自动驾驶系统测试场景, 充分考虑了道路交通安全法律法规的约束, 几乎涵盖了智能网联汽车自动驾驶功能测试规程的全部内容, 利用 Petri 网建模交叉路口测试场景, 通过 Tina 软件验证了所建模型的正确性, 并将 Petri 网模型转为形式化的 CCSL 约束集, 依赖 LTL 公式描述系统的属性, 通过 MyCCSL 工具验证了模型设计和系统需求的一致性. 实验结果表明了本文提出的基于交规建模自动驾驶交叉路口测试场景生成方法的有效性.

后续工作: 在交规建模的自动驾驶复杂测试场景背景下, 进一步研究基于搜索的测试用例生成方法和优化方法, 从而更好地解决自动驾驶系统安全性和可靠性的测试问题.

References:

[1] An DD, Liu J, Chen XH, *et al*. Formal modeling and dynamic verification for human cyber physical systems under uncertain environment. Ruan Jian Xue Bao/Journal of Software, 2021, 32(7): 1999−2015 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6272.htm [doi: 10.13328/j.cnki.jos.006272]

[2] Najm WG, Toma S, Brewer J. Depiction of priority light-vehicle pre-crash scenarios for safety applications based on vehicle-to-vehicle communications. In: Proc. of the Crash Causes. 2013. 1−33.

[3] Nitsche P, Thomas P, Stuetz R, *et al*. Pre-crash scenarios at road junctions: A clustering method for car crash data. Accident Analysis & Prevention, 2017, 107(10): 137−151.

[4] Hauer F, Schmidt T, Holzmuller B, *et al*. Did we test all scenarios for automated and autonomous driving systems? In: Proc. of the IEEE Intelligent Transportation Systems Conf. (ITSC). 2019. 2950−2955.

[5] Ding W, Xu M, Zhao D. CMTS: A conditional multiple trajectory synthesizer for generating safety-critical driving scenarios. In: Proc. of the 2020 IEEE Int'l Conf. on Robotics and Automation (ICRA). 2020. 4314−4321.

[6] Paardekooper J, Montfort S, Bracqumond A. Automatic identification of critical scenarios in a public dataset of 6000 km of public-road driving. In: Proc. of the 26th Int'l Technical Conf. on the Enhanced Safety of Vehicles (ESV). 2019. 1−8.

[7] Zhao D, Lam H, Peng H, *et al*. Accelerated evaluation of automated vehicles safety in lane-change scenarios based on importance sampling techniques. IEEE Trans. on Intelligent Transportation Systems, 2017, 18(3): 595−607.

[8] Beglerovic H, Stolz M, Horn M. Testing of autonomous vehicles using surrogate models and stochastic optimization. In: Proc. of the 20th Int'l Conf. on Intelligent Transportation Systems (ITSC). IEEE, 2017. 1−6.

[9] Althoff M, Lutz S. Automatic generation of safety-critical test scenarios for collision avoidance of road vehicles. In: Proc. of the IEEE Intelligent Vehicles Symp. (IV). 2018. 1326−1333.

[10] Klischat M, Althoff M. Generating critical test scenarios for automated vehicles with evolutionary algorithms. In: Proc. of the IEEE Intelligent Vehicles Symp. (IV). 2019. 2352−2358.

[11] Zhu XL, Wang HC, You HM, *et al*. Survey on testing of intelligent systems in autonomous vehicles. Ruan Jian Xue Bao/Journal of Software, 2021, 32(7): 2056−2077 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6266.htm [doi: 10.13328/j.cnki.jos.006266]

[12] Kumar SSV. Intersection collision avoidance for autonomous vehicles using Petri nets [MS. Thesis]. Indianapolis: Purdue University, 2019.

[13] Tang Y, Zhou Y, Wu F, *et al*. Route coverage testing for autonomous vehicles via map modeling. In: Proc. of the IEEE Int'l Conf. on Robotics Automation (ICRA). 2021. 11450−11456.

[14] Lima V, Talhi C, Mouheb D, *et al*. Formal verification and validation of UML 2.0 sequence diagrams using source and destination of messages. Electronic Notes in Theoretical Computer Science, 2009, 254(1): 143−160.

[15] Han D, Xing J, Yang Q, *et al*. Formal sequence: Extending UML sequence diagram for behavior description and formal verification. In: Proc. of the 40th IEEE Annual Computer Software and Applications Conf. 2016. 474−481.

[16] Soares JAC, Lima B, Faria JP. Automatic model transformation from UML sequence diagrams to coloured Petri nets. In: Proc. of the 6th Int'l Conf. on Model-driven Engineering and Software Development. 2018. 668−679.

[17] Petri C, Reisig W. Petri net. Scholarpedia, 2008, 3(1): 133−136.

[18] Liu G. Petri Nets: Theoretical Models and Analysis Methods for Concurrent Systems. Springer, 2022.

[19] He L, Liu G, He LF, *et al*. Prioritized time-point-interval Petri nets modeling multi-processor real-time systems and TCTLx. IEEE Trans. on Industrial Informatics, 2022. [doi: 10.1109/TII.2022.3222342]

[20] Gascon R, Mallet F, Deantoni J. Logical time and temporal logics: Comparing UML MARTE/CCSL and PSL. In: Proc. of the 18th Int'l Symp. on Temporal Representation and Reasoning. IEEE, 2011.

[21] He LF, Liu GJ. Time-point-interval prioritized time Petri nets modelling real-time systems and TCTL checking. Ruan Jian Xue Bao/Journal of Software, 2022, 33(8): 2947−2963 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6607.htm [doi: 10.13328/j.cnki.jos.006607]

附中文参考文献:

[1] 安冬冬, 刘静, 陈小红, 等. 不确定环境下 hCPS 系统的形式化建模与动态验证. 软件学报, 2021, 32(7): 1999−2015. http://www.jos.org.cn/1000-9825/6272.htm [doi: 10.13328/j.cnki.jos.006272]

[11] 朱向雷, 王海弛, 尤翰墨. 自动驾驶智能系统测试研究综述. 软件学报, 2021, 32(7): 2056−2077. http://www.jos.org.cn/1000-9825/6266.htm [doi: 10.13328/j.cnki.jos.006266]

[21] 何雷锋, 刘关俊. 模拟实时系统的点区间优先级时间 Petri 网与 TCTL 验证. 软件学报, 2022, 33(8): 2947−2963. http://www.jos.org.cn/1000-9825/6607.htm [doi: 10.13328/j.cnki.jos.006607]

夏春艳(1980－), 女, 博士生, 副教授, 主要研究领域为软件测试, 数据挖掘, 形式化方法.

黄松(1970－), 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, 软件安全性, 软件测试与质量评估.

郑长友(1986－), 男, 博士, 副教授, 主要研究领域为软件工程, 软件测试.

张清睿(1998－), 男, 硕士生, 主要研究领域为软件工程, 软件测试.

王宇(1998－), 女, 硕士生, 主要研究领域为软件工程, 智能软件测试.

魏瑀皓(1996－), 男, 硕士生, 主要研究领域为软件测试与质量评估, 智能软件测试.