软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2018,29(6):1699−1715 [doi: 10.13328/j.cnki.jos.005468]

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

机器人关节通信总线系统的建模与验证*

孟 瑶[1,2], 李晓娟[1,2], 关 永[1,2], 王 瑞[1,2], 张 杰[3]

[1](首都师范大学 信息工程学院,北京 100048)

[2](轻型工业机器人与安全验证北京市重点实验室(首都师范大学),北京 100048)

[3](北京化工大学 信息科学与技术学院,北京 100029)

通信作者: 李晓娟, E-mail: lixj@cnu.edu.cn

摘 要: 高速串行现场总线(controller area network,简称 CAN)被广泛部署到机器人通信系统中.而服务机器人任务具有并发性和高实时性的特点,因此,如何根据总线协议规范和应用需求精化设计模型,保证系统设计的正确性和实时性要求,避免设计阶段的漏洞十分必要.针对传统方法的局限性,提出使用形式化方法对基于 CAN 现场总线型控制系统进行建模分析.首先,对系统进行模型抽象和形式表达;其次进行形式建模和自动验证,在 UPPAAL 中实现主控制器、关节控制器、收发器、仲裁器和 CAN 总线的时间自动机模型;最后对机器人通信系统进行正确性验证和实时性分析.实时性分析发现:随着总线上关节节点数的增多,低优先级节点的最坏仲裁时延的增长速率加大.针对这个问题,在形式模型中加入了改进的动态优先级策略.实验结果表明:部署动态优先级策略后不仅减小了低优先级节点的仲裁时延,而且还可以加大 CAN 总线的节点负载量,为系统设计提供有效的指导和参考.

关键词: 形式化验证; 实时性;时间自动机;CAN;动态优先级

中图法分类号: TP311

中文引用格式: 孟瑶,李晓娟,关永,王瑞,张杰.机器人关节通信总线系统的建模与验证.软件学报,2018,29(6):1699−1715. http://www.jos.org.cn/1000-9825/5468.htm

英文引用格式: Meng Y, Li XJ, Guan Y, Wang R, Zhang J. Modeling and verification for robot joint bus communication system. Ruan Jian Xue Bao/Journal of Software, 2018,29(6):1699−1715 (in Chinese). http://www.jos.org.cn/1000-9825/5468.htm

Modeling and Verification for Robot Joint Bus Communication System

MENG Yao[1,2], LI Xiao-Juan[1,2], GUAN Yong[1,2], WANG Rui[1,2], ZHANG Jie[3]

[1](College of Information Engineering, Capital Normal University, Beijing 100048, China)

[2](Beijing Key Laboratory of Light Industrial Robot and Safety Verification (Capital Normal University), Beijing 100048, China)

[3](College of Information Science & Technology, Beijing University of Chemical Technology, Beijing 100029, China)

Abstract: Controller area network (CAN) is a high-speed serial fieldbus, widely deployed in the robot communication system. Due to the concurrency of service-oriented robot tasks and tightness requirement for real-time, it is necessary to explore how to refine the design

* 基金项目: 国家自然科学基金(61373034, 61572331, 61472468, 61602325); 国家科技支撑计划(2015BAF13B01); 国际科技合作计划(2011DFG13000); 北京市科委项目(LJ201607); 北京市教委科研基地建设项目(TJSHG201510028010); 北京市属高等学校创新团队建设与教师职业发展计划(IDHT20150507)

Foundation item: National Natural Science Foundation of China (61373034, 61572331, 61472468, 61602325); the National Key Technology Research and Development Program (2015BAF13B01); the International Cooperation Program on Science and Technology (2011DFG13000); the Project of Beijing Municipal Science & Technology Commission (LJ201607); the Project of Construction of Innovative Teams and Teacher Career Development for Universities and Colleges Under Beijing Municipality (IDHT20150507); the Scientific Research Base Development Program of the Beijing Municipal Commission of Education (TJSHG201510028010)

本文由形式化方法的理论基础专题特约编辑傅育熙教授、李国强副教授、田聪教授推荐.

收稿时间: 2017-07-01; 修改时间: 2017-09-01, 2017-11-06; 采用时间: 2017-12-05; jos 在线出版时间: 2017-12-28

CNKI 网络优先出版: 2017-12-29 13:19:27, http://kns.cnki.net/kcms/detail/11.2560.TP.20171229.1318.009.html

model according to the bus protocol specifications and application system, in order to ensure the correctness and real-time requirements of the system and avoid bugs in design process. However the traditional methods are limited. This paper proposes a formal method to verify and analyze the correctness of CAN based fieldbus real-time control system. The model abstraction, formal modeling and automatic verification are presented for the system including the time automata model of master, joint controller, transceiver, arbitration and CAN bus. The formal models show that the worst arbitration delay of the low priority node increases rabidly with the increasing number of joints on the CAN bus. Furthermore, an improved dynamic priority strategy is designed and added into the formal models in order to improve the worst arbitration delay problem. The experimental results show that the deployment of the dynamic priority strategy not only reduces the arbitration delay of the low priority nodes, but also increases the capacity of nodes on CAN bus. The result provides guidance and effective reference for the system design.

Key words: formal verification; real-time; timed automata; CAN; dynamic priority

CAN(controller area network)总线是一种高速串行总线,也是目前支持分布式控制且应用最为广泛的现场总线之一.与其他现场总线相比,CAN 总线具有高实时性、高可靠性、易开发且易于扩展等优点.目前,CAN 总线不仅仅应用于汽车行业[1,2],而且越来越被广泛地应用于船舶业、铁路轨道、导航等,近年来则被广泛应用于机器人通信系统中[3−5].由于机器人通信系统的复杂性、高并发性和对实时性的高要求,因此,保证机器人精确和实时地完成预期动作非常重要.特别是在机器人分布式节点间的信息传输中,通信时延必须满足规定的响应时间需求,否则将影响系统的稳定性或导致系统出错、崩溃.CAN 总线协议规范中,用信息静态优先级策略进行通信实时性能保证.但随着 CAN 总线上节点的增多和对实时性要求的提高,静态优先级策略不能满足节点间传输的时延需求,而工业中则通过改用总线协议,如 EtherCAT,或增加 CAN 的个数来解决实时性问题.本文在形式化模型中部署动态优先级策略,并通过实验分析了其对通信实时性的影响.实验结果表明,设计人员可以根据应用在设计中进行优先级调整来满足关键应用的实时性要求.

目前,对 CAN 总线协议的验证主要集中于电动汽车的控制系统中,鲜有文献针对机器人通信环境对 CAN 总线的正确性和实时性进行验证分析.而在这些 CAN 协议的验证方法中,主要采用的还是传统的验证方法,如测试、模拟和仿真,然而这 3 种方法都具有一定的局限性,不能覆盖所有的执行路径,所以验证结果不完备.

- 测试方法往往在设计的最后阶段借助实体机搭建测试平台进行参数设定和测试,这种方法需要等到最后阶段才能进行,不仅费时而且可复用度不高,在测试过程中参数设定有限,使得测试结果不精确;
- 模拟和仿真的方法主要是利用计算机软件进行环境模拟和动作仿真,进而对通信进行分析.例如,文献[6]采用 CANoe 软件对重型货车的 CAN 网络进行建模和仿真,验证了其 CAN 网络的可行性.CANoe 是一个对车载网络进行仿真分析的软件,它提供编程功能,信号也可以实时分析.但是它依赖硬件,提供的参数设置有限,使实验结果不完备.

为了能够在设计初期保证其系统设计的严谨性、避免设计阶段的 BUG、严密而系统地验证并分析 CAN 总线在机器人通信中的正确性和实时性,本文针对机器人关节节点等分布式系统的应用,通过形式建模,系统分析随着节点和传输消息的增加对系统实时性的影响,为 CAN 总线在机器人节点间通信的设计提供参考.近几十年来,随着形式化技术的不断推进和验证工具的不断完善,形式化方法已经逐步成为一种成熟的高可靠验证技术,尤其是在设计初期细化设计模型方面,成为不可或缺的补充工具.形式化方法已经在众多领域取得了巨大的进步[7].模型检测[8]是形式化方法的一种,与定理证明[9]相比,能实现系统的自动化证明,减少人为参与.Alur[10]提出的时间自动机理论是模型检测技术的重要理论基础,在实时系统的验证和自动验证工具的发展中起到关键的作用.基于时间自动机理论的模型检测技术主要是借助时间自动机的形式对系统建模,通过搜索模型的状态空间,验证属性是否成立.目前,模型检测方法已经应用到协议验证方面,文献[11]使用模型检测工具 UPPAAL 对 Zeroconf 协议进行了建模和分析,提供了一种对协议进行建模的方法,并在文中介绍了对模型抽象简化的思路.文献[12]采用时间自动机的方法对 CAN 协议规范进行了建模,并对 CAN 规范的几个重要属性进行了验证,提出由于饥饿现象和 bus-off 节点的存在导致有些属性不满足需求.为解决这个问题,作者在文中引入了错误恢复机制,并对该机制的有效性进行了验证.

机器人应用领域越来越广泛已经不仅仅用于工业,由于市场的需求,大量的服务型机器人开始应用于医疗

和家庭,因此,机器人控制系统对实时性要求越来越高.为保证机器人系统的实时性,设计者从多个方面对机器人控制系统进行了完善:操作系统方面采用了具有实时性的 RGMP-ROS 操作系统[13],硬件方面采用执行速度快的控制器,节点间通信方面采用具有实时性的 CAN 总线.但是 CAN 总线采用的是静态优先级策略,其通信的实时性仍然存在一些问题,对 CAN 总线通信的性能进行分析优化十分必要.本文采用模型检测的方法对 CAN 总线型控制在机器人通信中的功能正确性进行了验证,并对 CAN 的实时性进行分析,在静态优先级基础上部署了动态优先级策略,对两种策略对实时性的影响做了对比.

模型检测方法的过程主要包括 3 部分.

- 根据要验证的性质对系统进行属性抽取用计算树逻辑(CTL)表示出来;
- 建立形式化框架对系统进行抽象和简化,建立时间自动机模型;
- 在模型检测验证工具中对属性进行验证:如果通过,则说明系统满足需求;不通过,则说明系统不满足需求或系统模型建立不严密完善.

UPPAAL[14]是应用比较广泛的模型检测工具之一,它不仅可以检验模型,还包含了检测死锁的机制,特别是其提供的时钟变量,能够准确地表达模型需要满足的时钟约束,对系统的实时性分析提供保证.因此,UPPAAL 在强调实时性的系统验证方面应用广泛,例如:用 UPPAAL 验证工具对 Zeroconf 协议进行验证和分析[11];用 UPPAAL 模型对多处理器实时系统可调度性进行分析[15].本文采用模型检测技术借助模型检测工具 UPPAAL 对机器人通信系统的功能正确性进行验证,同时对其实时性进行了分析,主要包括:

- 主控制器部分、关节控制器部分、收发器部分、仲裁器部分、总线部分的形式化表示和在 UPPAAL 中的实现;
- 关键属性的抽取和验证;
- 分析实时性,提出一种动态优先级策略并在模型中部署,进行分析验证.

本文提出的 CAN 总线型模型可以作为其他分布式控制系统的总线型验证模板,根据具体运行环境,可以对模型进行配置和扩展,快速建立总线型模型.

本文第 1 节描述基于 CAN 的机器人控制系统,并说明 CAN 总线的仲裁机制,给出系统形式化框架.第 2 节给出构建时间自动机的步骤和模型的形式化描述,并详细介绍使用 UPPAAL 实现系统的建模,此外还分享了一些建模经验.第 3 节阐述用 CTL 公式描述属性.第 4 节对系统进行功能正确性验证,分析 CAN 实时性并提出一种动态优先级策略,改进模型,并验证系统实时性.第 5 节综述相关工作.第 6 节总结本文工作并提出下一步的研究方向.

1 系统描述

1.1 基于CAN的机器人控制系统

控制系统分为集中式控制系统、分布式控制系统和现场总线分布式控制系统.

- 集中式控制系统采用的是集中式控制可靠性较低且不易扩展;
- 分布式控制系统的核心思想是“信息集中,控制分散”,但是它的通信由专用网络的封闭系统来实现,具有一定的缺陷;
- 现场总线控制系统是一种全分布式结构,具有良好的开放性、互操作性和互用性.它把控制功能彻底下放到工业生产过程的现场,依靠现场智能设备本身便可实现基本控制功能.

机器人是一个多自由度系统,机器人控制本质上是对各个关节的运动进行控制,使其协调运动,从而完成一些相对复杂的动作.基于这些特点,机器人控制系统采用的是现场总线分布式控制系统.主要由上位主控计算机模块、通信模块和下位关节控制器模块组成:上位主控计算机负责整个系统的调度管理、在线运动规划、故障诊断和人机交互等功能;通信模块负责上位计算机与下位关节控制器之间的实时信息交换;各关节控制器和驱动直流无刷电机集成在一起,各个关节的运动由各关节控制器发出 PWM 信号驱动直流电机实现.

图 1 是一个简单的机器人分布式控制系统,其中,PC 机是上位主控计算机,通信模块采用的是能保证低负载

量通信实时性的 CAN 协议,与总线相连的是下位关节节点.在实际应用中,这些节点可以是关节伺服控制系统或其他智能设备.

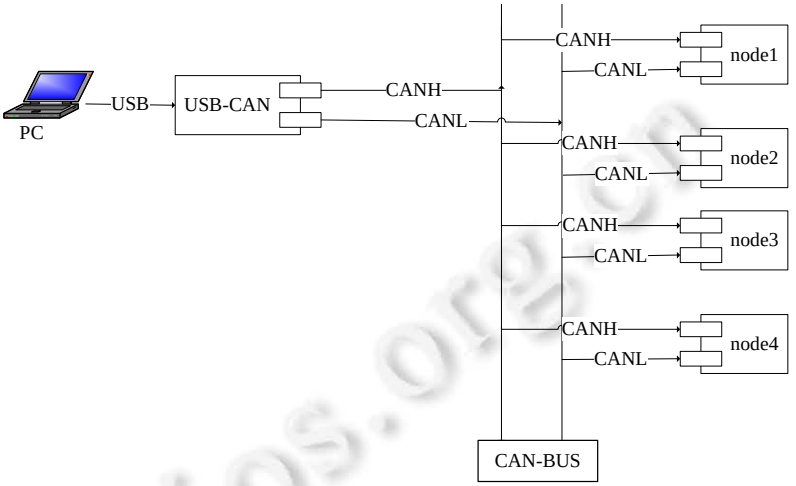

Fig.1 Robotic control system based on CAN

图 1 基于 CAN 总线的机器人控制系统

1.2 CAN协议介绍

- CAN 报文格式

在 CAN 协议规范中[16]提供了 4 种类型的帧:数据帧、远程帧、错误帧和过载帧.所有类型的帧都具有固定的格式.CAN 协议有两种帧格式:一种是标准格式,另外一种是扩展格式.其中,标准格式的标识符有 11 位,而扩展格式的标识符有 29 位.报文中的数据都是由显性位(0)和隐形位(1)构成的,其中,显性位的优先级比隐形位高.本文中采用的是标准帧格式,标准帧格式如图 2 所示:SOF 代表的是帧起始位,且其仅由一个显性位构成,第 1 个发送节点的帧起始前沿是各个节点开始发送的标志.11 位标识符和发送请求位(RTR)组成了数据标准帧的仲裁场,CAN 协议通过仲裁场实现静态优先级策略,即固定节点优先级.其中,11 位标识符的发送顺序为从高位到低位依次发送.当 RTR 为显性位时表示的是数据帧,为隐性位时代表是远程帧,因此,数据帧的优先级要高于远程帧,其中,远程帧的数据场为空.被请求节点必须通过 ID 辨识来响应请求.EOF 是帧结束位.

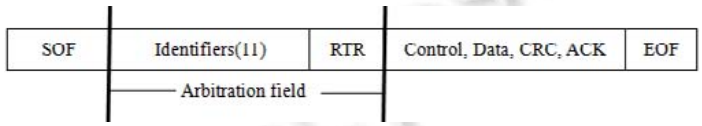

Fig.2 CAN message frame format

图 2 CAN 报文格式

- CAN 仲裁机制

CAN 总线采用的是固定优先级机制,如图 2 所示的报文格式中,仲裁场表示报文的优先级,即节点优先级,而不是表示报文发送的目的地址.CAN 总线采用非破坏性仲裁机制,具体仲裁过程为:如果发送时刻总线空闲,任何一个最先发送的数据帧都可以占用总线;如果有一个数据帧需要发送,但此刻总线使用权被其他数据帧占用,那么无论该数据帧优先级多高都要排队等候,直到总线空闲时才可以进行下一轮竞争.接收信息的节点在成功接收到数据后会向发送节点做出反馈,发出节点收到此回应则表明传输完毕,否则信息就会重发;当有两个或者两个以上的节点要发送数据时,就会产生冲突,这时则需要进行总线仲裁,其过程是逐位对报文的标识符进行比较.标识符小的节点(标识符越小优先级越高)优先发送数据,标识符大的自动退出,等待下一次空闲时刻的到

来.CAN 总线的这种仲裁机制,使得 CAN 总线较其他总线而言具有高实时性,这也是将 CAN 总线应用到机器人控制系统中的原因.但也正是由于 CAN 总线采用这种静态优先级策略,使其不能保证高负载下低优先级节点的实时性.

1.3 形式化架构

在对 CAN 协议规范和 CAN 源码分析的基础上,并结合基于 CAN 的机器人控制系统,本文对基于 CAN 的现场总线型控制系统进行了抽象、简化.抽象出了主控制器、关节控制器、收发器、仲裁器和总线这 5 个模型,并根据实际通信过程对这 5 个模型进行了层次划分,包括 3 部分:应用程序部分(主控制器和关节控制器)、收发器部分和总线部分.应用层通过收发器向总线发送/接收数据.其中,收发器部分包含了仲裁器,这样设计的原因是每个节点可以独立且并发的向总线发送数据,能够真实地反映多个节点同时请求总线的情况.具体架构如图 3 所示.框架中,控制器与收发器、仲裁器之间的同步由同步信道和全局变量给出.根据 CAN 总线的静态优先级策略,这里为每个节点分配了优先级,Master 节点 *ID* 号为 0,拥有最高优先级;Joint1 节点 *ID* 号为 1,优先级次之.以此类推,Joint2 节点 *ID* 号为 2,Joint3 节点 *ID* 号为 3,Joint3 节点的优先级最低.

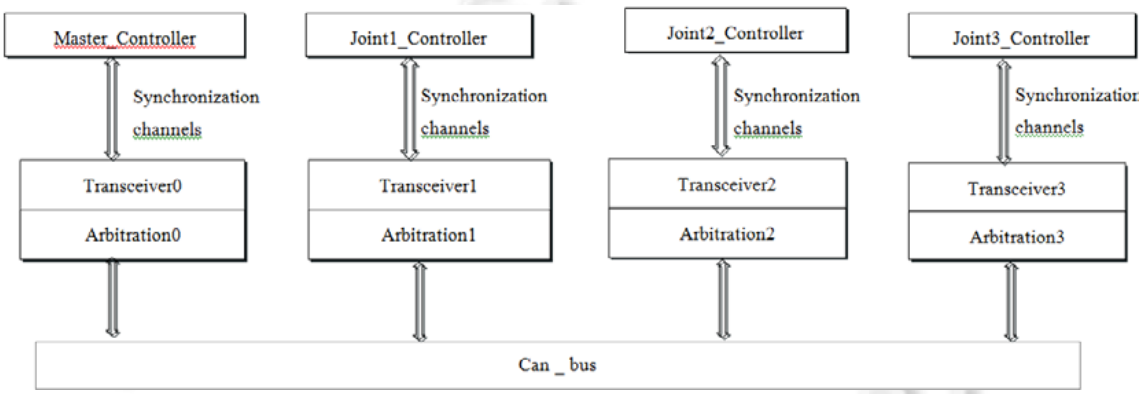

Fig.3 Formal frame

图 3 形式化架构

整个系统由上至下可以描述为:

- System=Application||Transceiver||Bus;
- Application=Master Controller||Joint Controller$_i$∈N(i);
- Transceiver=Transceiver$_j$∈(N+1)(j)||Arbitration$_j$∈(N+1)(j).

其中,||表示模型的并行组合;N 表示总线上挂接的关节节点数;N+1 表示总线上的节点总数,即,包括主节点.架构中每个模型都是一个模板,实现过程中每个进程都是一个模板的实例化.

2 基于 UPPAAL 的系统建模

根据上节抽象的形式化框架,本节具体介绍了框架中涉及到的主要模型形式化表示,并以该形式化描述为基础,在 UPPAAL 中具体实现了各个模型的时间自动机模型.

2.1 系统形式化表示

第 1 节提出了一种现场总线型控制系统的形式化架构,在采用该架构对系统进行验证和分析的整个过程中,构建系统的规范化模型是其中一项十分重要的工作,因为后续的验证和分析都需要在正确的系统模型下进行.本小节给出构建时间自动机模型的方法步骤,时间自动机(timed automata)是扩充的有限状态自动机,其引入了时钟的概念.所以,时间自动机提供了一个简单又全面的方法来刻画实时系统中的各种时间属性.构建时间自

动机模型的主要步骤为:

- 根据基于 CAN 的机器人控制系统的通信特点,分析每个模块的逻辑功能,为每个模型抽象出其可能所处的各种状态;
- 结合各个模块的功能和整个系统的运行过程,规范模块中状态迁移规则,包括条件约束、常量约束、时钟约束以及同步约束.

根据上述步骤就可以构建每个模块的时间自动机模型,本小节结合机器人控制系统的基本结构和上述步骤,以主控制器和关节控制器为例对其时间自动机模型的构建进行详细描述,并给出其形式化表示.

主控制器在基于 CAN 的机器人控制系统中表示上位主控计算机模块,在该形式化架构中,主要功能是根据需求在一个控制周期内向所有关节控制器发送控制命令,并根据相应的中断信号接收关节控制器的反馈信息,为下一个控制周期做准备.为与工业应用保持一致,模型中主控制周期设定为 16 个时间单位.主控制器节点具有空闲(idle)、发送(sending)、等待仲裁、控制周期内所有命令传输成功、接收(receiving)等 5 种工作状态,其状态图如图 4 所示.

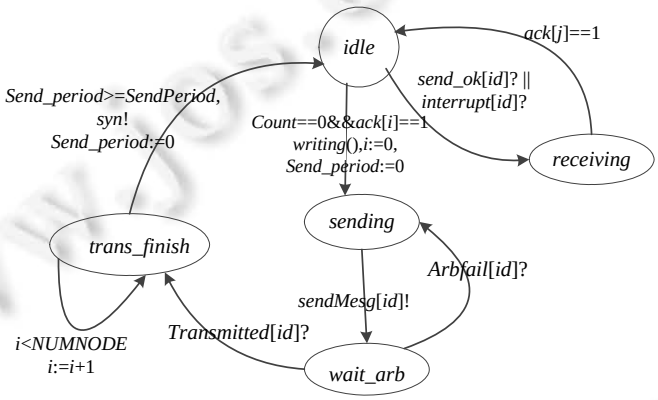

Fig.4 State figure of master

图 4 主控制器状态图

根据时间自动机网络语义,可以将该模型形式化描述为:

$(idle,[])\text{----}^{count==0\&\&ack[i]==1}\text{------}\rightarrow(sending,[])\langle writing(),i:=0,Send\_period:=0\rangle$

$(sending,[])\text{----}^{sendMesg[id]!}\text{------}\rightarrow(wait\_arb,[])$

$(wait\_arb,[])\text{----}^{arbfail[id]?}\text{------}\rightarrow(sending,[])$

$(wait\_arb,[])\text{----}^{transmitted[id]?}\text{------}\rightarrow(trans\_finish,[])$

$(trans\_finish,[])\text{----}^{i<NUMNODE}\text{------}\rightarrow(trans\_finish,[])\langle i:=i+1\rangle$

$(trans\_finish,[])\text{----}^{Send\_period\geqslant SendPeriod,syn!}\text{------}\rightarrow(idle,[])\langle Send\_period:=0\rangle$

$(idle,[])\text{----}^{send\_ok[id]?||interrupt[id]?}\text{------}\rightarrow(receiving,[])$

$(receiving,[])\text{----}^{ack[j]==1}\text{------}\rightarrow(idle,[])$

关节控制器在该系统中表示关节伺服控制系统或其他智能设备,主要功能是根据接收到的控制命令在执行周期内驱动直流电机完成相应的运动,并向主控制器反馈当前位置信息或状态信息.为与工业应用保持一致,模型中执行周期设定为 5 个时间单位.关节控制器节点具有空闲态(idle)、接收态(receiving)、执行态(execute)、等待反馈状态(ack_waiting)、发送反馈态(ack_send)、等待仲裁状态(waiting_arb)和反馈完成态(ack_finish)这 7 个工作状态.图 5 给出关节控制器的状态图模板,可以根据该模板创建出更多需要的节点模型.

根据时间自动机网络语义,可以将该模型形式化描述为:

$(idle,[])\text{----}^{syn?,packet[id-1].Messid==id}\text{------}\rightarrow(receiving,[])$

$(receiving,[])\text{----------}\rightarrow(execute,[t\_execut\leqslant 5])\langle t\_execut:=0\rangle$

(*execute*,[*t_execut*≤5])----$^{interrupt[id]!,t\_execut>5}$------→(*ack_waiting*,[])

(*ack_waiting*,[])----$^{packet[id-1].Mtype==2}$------→(*ack_send*,[])⟨*x*:=0⟩

(*ack_waiting*,[])----$^{packet[id-1].Mtype==1}$------→(*ack_send*,[])⟨*x*:=0⟩

(*ack_send*,[])----$^{sendMesg[id]!}$------→(*waiting_arb*,[])

(*waiting_arb*,[])----$^{Arbfail[id]?}$------→(*ack_send*,[])

(*waiting_arb*,[])----$^{transmitted[id]?}$------→(*ack_finish*,[])

(*ack_finish*,[])----$^{send\_ok[id]!}$------→(*idle*,[])⟨*resetPacket*(*id*),*x*:=0⟩

上述形式化描述中:一个括号(·)代表模型中的一个状态,括号中第 1 个元素表示状态名,第 2 个元素表示该状态的时钟约束;"---→"表示状态迁移,上面的元素表示该状态迁移对应的条件约束;同步信号 *syn*!,*syn*?,*interrupt*[*id*]!,*interrupt*[*id*]?等表示模型间的交互和同步约束,其中,*id* 指明参与交互的具体模型;表示总线上挂接节点个数的常量 *NUMNODE* 在该模型自动机中表示常量约束.其他模型的时间自动机模型都可以按着上述步骤构建,本文就不再赘述.

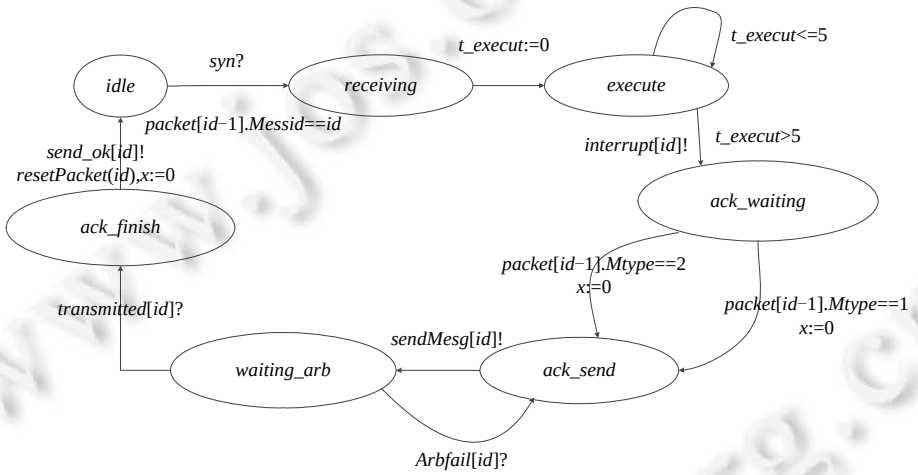

Fig.5 State figure of joint_controller

图 5 关节控制器状态图

2.2 符号化模型检测工具UPPAAL

目前,基于时间自动机的模型检验方法使用的验证工具是 UPPAAL[17],它是由瑞典 Uppsala 大学和丹麦 Aalborg 大学联合开发的.其采用一组带有有界整型变量的时间自动机,对可以被描述为非确定性的并行过程的实时系统进行模拟.每一个进程可以通过有限个控制结构、时间自动机、数值时钟和进程间的通道来描述.其中,通道用来实现进程间的通信同步,假设用 *c* 来表示一个通道,那么 *c*?代表接收了一个来自其他进程的事件,*c*!代表向其他进程发出了一个事件.在 UPPAAL 中,使用共享变量来实现异步通信.UPPAAL 为属性描述提供了简化版的计算树逻辑(CTL)公式,它由路径公式和状态公式两部分组成,前者量化模型的路径或轨迹而后者则描述单独的状态,其中,路径公式又可以分为可达性、活性和安全性.表 1 列出了具体的公式描述.

Table 1 Path formulas

表 1 路径公式

公式	描述
A⟨⟩*p*	对于所有路径,*p* 最终成立
A[]*p*	对于所有路径的所有状态,*p* 总成立
E⟨⟩*p*	存在一条路径,使得 *p* 最后成立
E[]*p*	存在一条路径,其所有状态 *p* 总成立
P imply q	如果 *p* 满足,那么 *q* 也满足

2.3 系统在UPPAAL中的实现

UPPAAL 中实现的每个模型都是对一个时间自动机的实例化,在工具中,可以通过参数设定实现多个实例.

2.3.1 主控制器模型

主控制器部分的时间自动机如图 6 所示,报文信息存放在结构体 *Packet* 中,包括报文类型(数据帧、远程帧)和报文 *ID* 号,这里,为了更明确表达模型,报文 *ID* 即为节点 *ID* 号.局部时钟变量 *Send_period* 表示主控制器的控制周期,主控制器在一个控制周期内向所有节点发送控制命令,待所有控制命令发出后,主控制器发送一个同步命令 *syn*!,下位关节节点控制器在接收到这个同步命令后 *syn*?,开始执行命令,做出相应的动作并向主控制器反馈当前位置或状态信息.初始时,主控制器位于初始状态,由初始态迁移到 *idle* 态的过程中,初始化时钟变量 *Send_period*.*idle* 态表示主控制器处于空闲状态,此时,主控制器既可以向关节控制器发送控制命令,也可以接收来自关节控制器的反馈信息.当总线上没有请求发送的节点且各个关节控制器在上一个控制周期后都已成功反馈时,主控制器由 *idle* 态迁移到 *start_sending* 态.*select* 选项 *e*:*int*[1,2]用来表示当前发送的帧类型,*e* 可以随机选取 1 或 2(1 表示数据帧,2 表示远程帧).因为值是随机选取的,所以可以准确地反映不同类型报文请求总线的情况.函数 *writing*()用来封装当前请求发送的帧信息.在工业中,机器人的通信过程是由主控制器封装各个关节节点的控制命令并同时发布,关节控制器根据标识符订阅自己的控制命令.为了能更准确地反映工业中机器人的通信过程,在建模时抽象了这一机制.主控制器发送控制命令后进入仲裁阶段,仲裁成功则迁移到 *trans_finish* 状态,仲裁失败重新回到 *start_sending* 态.*trans_finish* 状态的循环模拟了主控制器的封装过程,此时,结构体数组 *Packet*[*N*]中封装了各个节点的帧信息.控制命令发出后,发送同步信号 *syn*!并迁移到 *idle* 态.CAN 总线上的关节节点通过中断的方式向主控制器反馈信息,在该时间自动机中,主控制器接收到 *interrupt*[*id*]!信号,说明有关节节点要反馈信息,则主控制器进入准备状态.当主控制器接收到来自关节控制器的发送成功信号 *send_ok*[*e*]!后自动机进入接收状态,当表示关节节点完成响应的变量 *ack* 的值为 1 时,该自动机回到 *idle* 态,开始下一轮控制.

为了增强自动机的可复用性,将表示CAN 总线挂接的下位关节节点数的变量 *NUMNODE* 声明为常数变量,如果挂接的节点数增加只需修改这个常量的值即可,不用对自动机进行修改,这大大提升了效率,减少了大量工作量.

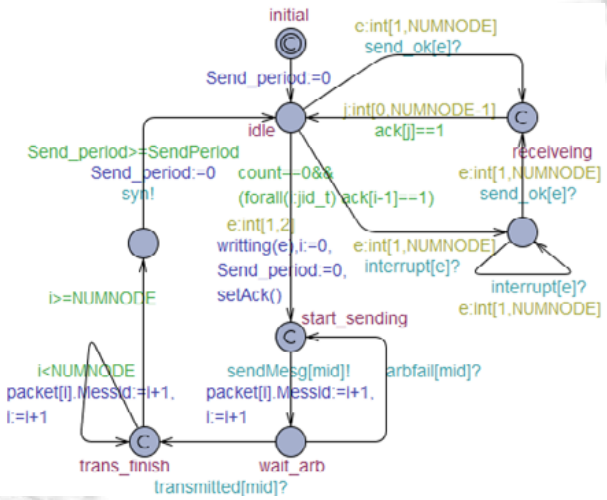

Fig.6 Module of master

图 6 主控制器模型

2.3.2 收发器模型

收发器部分的时间自动机如图 7 所示,初始状态是 *idle* 态,表示当前节点处于空闲状态,当收发器接收到来

自节点向总线发送的请求信号 *sendMesg*[*id*]!后,开始探测总线是否处于空闲状态:如果当前有节点占用总线,则转到 *waiting* 等待状态;如果总线空闲,总线时间自动机则会发送 *next*!信号,收发器自动机迁移到 *start_arb* 状态,同时触发仲裁自动机并发送仲裁开始信号 *arb_start*[*id*]!,其中,*id* 表示的是当前参与仲裁的节点.仲裁失败后,自动机迁移到 *req_denied* 状态,表示请求失败,并向仲裁器自动机发送仲裁结束信号 *arb_over*[*id*]!,同时,向上层控制器发送仲裁失败 *arbfail*[*id*]!信号.*count* 的值减 1,表示当前请求发送的节点数减 1.仲裁成功后,自动机迁移到 *req_suc* 状态,表示请求占有总线成功.同样也向仲裁器自动机发送仲裁结束信号 *arb_over*[*id*]!,进入 *sending* 状态,传输成功后,向上层控制器发送 *transmitted*[*id*]!信号.同时,更新 *active* 和 *count* 的值,重置总线值.在该自动机中引入了局部 *clock* 变量 *t*1,用来表示节点的传输时延,表达式 *t*1≤*Trans_t* 为位置不变量,表示自动机在发送状态最多只能停留 *Trans_t* 时间单位就要迁移到 *idle* 态.

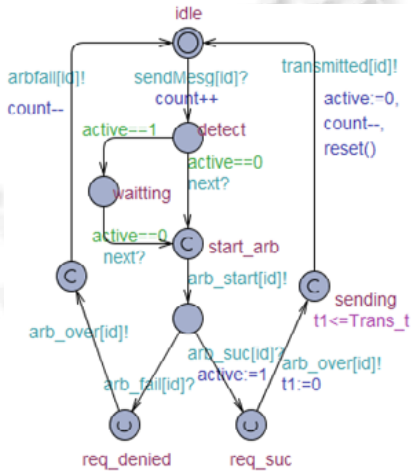

Fig.7 Module of transceiver

图 7 收发器模型

2.3.3 关节控制器模型

关节控制器部分的时间自动机如图 8 所示.

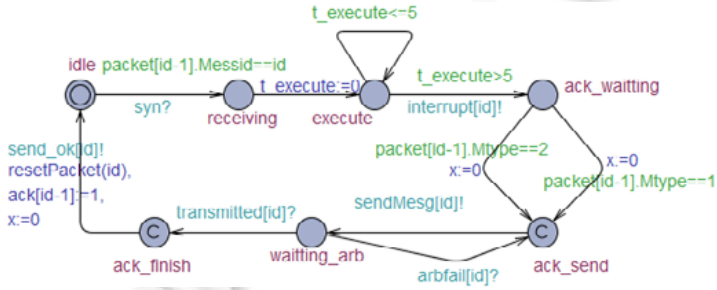

Fig.8 Module of joint_controller

图 8 关节控制器模型

该自动机是根据 CAN 总线上所挂接的设备数量来定义的 CAN 节点实例,每个实例表示了一个节点所处的状态,用来反映传感器的行为.初始态 *idle* 表示关节处于空闲状态,当自动机收到来自主控制器发送的同步信号后转到 *receiving* 接收态,并且开始执行相关命令.局部时钟变量 *t_execute* 表示执行周期,为跟目前工业中关节控制的执行周期保持一致,模型中将节点执行周期设为 5 个时间单位,在执行周期结束后,节点需要通过发送中断信号 *interrupt*[*id*]!将采集到的信息反馈到主控制中.此时,自动机迁移到反馈 *feedback* 状态.为保证节点信息能

够快速得到处理,CAN 总线在实际工作中采用的是中断的方式.在 *feedback* 状态引出两个分支分别表示反馈的是位置信息还是状态信息.在节点反馈信息时也需要参与总线的仲裁,只有仲裁成功后才能发送信息,如果仲裁失败,则退出总线等待下一次的仲裁.待信息成功发送后,向主控制器发送 *send_ok*[*id*]!信号触发主控制器进入接收状态,并将变量 *ack* 的值置为 1,表示该节点已经成功反馈信息.

2.3.4 仲裁器模型

仲裁器部分的时间自动机如图 9 所示,初始状态 *idle* 表示该节点处于空闲状态,等待 CAN 收发器发送仲裁请求 *arb_start*[*id*]!.当收到仲裁请求信号后进入 *start_arb* 状态开始总线仲裁过程,即,按位对帧仲裁场的 11 位标识符进行位仲裁,具体过程为:

计算总线值:自动机由 *start_arb* 状态迁移到 *listen_bus* 状态时计算当前总线值.即 *signal*[*i*]:=*signal*[*i*]**Id*[*id*][*i*],其中,*signal*[*i*]表示当前计算所得的总线值,它的初始值是 *signal*[*i*]={1,1,…,1},*Id*[*id*][*i*]表示当前节点所发送帧的仲裁场的 11 位标识符序列,两个数组进行按位“与”运算所得的结果即为当前所要求的总线值;

比较当前发送位的值与所求得的总线值是否相同:若不同,则节点请求发送失败,退出仲裁,此时,自动机所处状态为 *req_fail*;若当前发送位的值与所求得的总线值相同但 11 位标识符序列还没有完全比较完,则按此方式继续对下一位进行仲裁;若当前发送位的值与所求得的总线值相同并且 11 位标识符均已仲裁完毕,则表示该节点请求成功,仲裁过程结束,此时,自动机所处状态为 *req_success*.无论仲裁成功还是仲裁失败,仲裁时间自动机都会接收到仲裁结束信号 *arb_over*[*id*]!,并回到 *idle* 态等待下一次的仲裁.这样,整个系统才能持续运行.

模型在 *listen_bus* 状态引入状态不变量 $t \leqslant Tbit$ 描述总线上每个位时间的持续时间信息,其中,局部时钟变量 *t* 作为该状态的时钟约束,描述该状态下的时间迁移.输出分支中 $t \geqslant Tbit$ 作为状态迁移的条件约束用来保障总线上位时间的持续时间,即,总线上位传输成功后才会做后续的值比对.

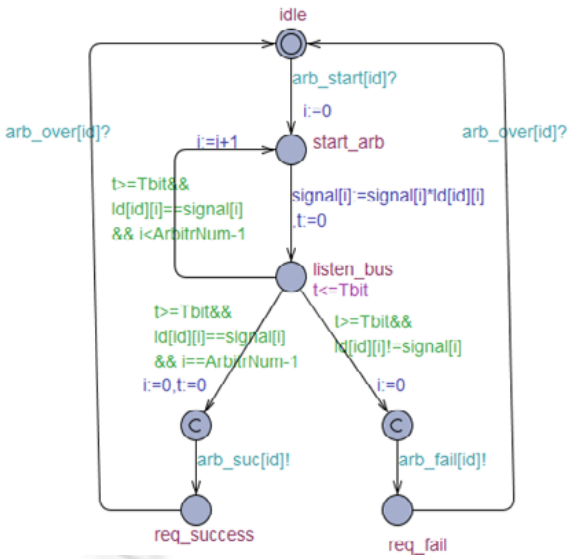

Fig.9 Module of arbitration

图 9 仲裁器模型

2.3.5 总线模型

总线部分时间自动机如图 10 所示,该自动机分为两个状态空闲态(*idle*)和忙态(*busy*),当没有节点向总线发送数据且也没有节点从总线上读取数据时,总线处于空闲态.只要有节点占有总线,总线即为忙碌状态.为表示总线的状态,引入了两个变量 *count* 和 *active*,*count* 代表向总线发送请求的节点数,每当有节点向总线发出请求时,*count* 的值都会加 1,当 *count* 值为 1 时,表示当前只有一个节点发送请求,此时,该节点可以占有总线发送数据;

当*count*的值大于1时,表示有多个节点向总线发送请求,此时需要参与总线仲裁,赢得总线的节点才可以发送或接受数据.*active*变量表示当前是否有节点占有总线,*active*的值为1,表示当前有节点占用总线,此时,总线处于忙碌状态;当 *active* 的值为 0 时,总线由忙碌状态转为空闲状态.

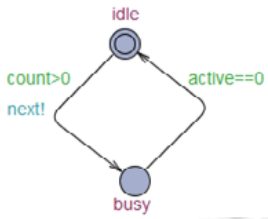

Fig.10 Module of bus

图 10 总线模型

2.4 建模经验

本节介绍在抽象建模过程中的一些经验.

- 经验 1. *clock* 变量作用域的定义

UPPAAL 中提供的 *clock* 变量是随着时间自动机的状态迁移而自动增加的,因此,变量在使用前必须要初始化.若声明的作用域不恰当,作用域范围内所有状态的迁移都会导致该变量值的增加,则导致该变量所记录的值不准确.本文在模型 *joint_controller* 中引入 *clock* 变量 *x*,用来记录关节节点仲裁时延.最初,该变量是在 *idle* 态迁移到 *receiving* 态时进行的初始化,导致变量记录的时延不精确.分析 UPPAAL 中时间变量的执行原理后,对变量的作用域进行了调整,初始化在 *ack_waitting* 态迁移到 *ack_send* 态时完成,从而精确地记录了节点的仲裁延迟.

- 经验 2. *committed* 状态的使用

committed 状态是 UPPAAL 提出的约束状态,该状态不允许时间的流逝,即:当时间自动机处于该状态时,不能在此状态停留.并且,*committed* 状态要求下一个状态必须有出口迁移路径.如果一个状态设置了必须满足的时间约束后,该状态不能设置成 *committed* 态,否则会出现死锁.但是在 *committed* 状态进行操作具有原子性,使用它可以明显减少系统状态空间.本文为了减少系统的状态空间和精确地记录仲裁延迟时间,在模型中多处引入了 *committed* 状态.也正是 *committed* 位置状态大量减少了模型状态迁移时的时间流逝.

- 经验 3. 合理设计自动机间的同步信道

时间自动机之间的同步是通过同步信道和全局变量实现的,但是在设计模型时,要注意合理地使用同步信道,引入的同步信道要尽量精而适用:如果引入的同步信道过少,不能精确地反映系统运行机制;但是引入的同步信号过多会增加系统的状态空间,则会出现系统一直运行而导致的内存耗尽问题.设计初期,仲裁器模型和收发器模型交互时没有设计 *arb_over*[*id*]?和 *arb_over*[*id*]!信号对,导致仲裁器无法循环执行,不能精确反映机器人周期性发送数据的机制.当然,同步信道问题并不是唯一导致内存耗尽的原因,当模型过于复杂或者状态迁移过多时,也会引起内存耗尽的问题.

3 属性抽象与形式化表达

为了验证系统的功能正确性,本文抽取了 7 个关键属性,并用计算树逻辑(CTL)公式表示出来.

CTL 由路径公式和状态公式两部分组成:状态公式表示模型中的一个状态,它可以是一个简单的表达式,如 $t\leqslant 5$,也可以是类似 *Bus.idle* 的表达式,其中,*Bus* 表示的是时间自动的名称,*idle* 表示时间自动机中的一个状态;路径公式描述的是模型中的执行轨迹.

属性的抽象表示如下.

- 安全性

对基于 CAN 的机器人通信系统,如果在某个时刻系统进入死锁状态,则表示整个系统不能正常运行,因此,对系统进行安全性验证十分有必要.在 UPPAAL 中,无死锁的属性表示为

$$A[]\ \text{not}\ deadlock.$$

- 活性

在系统中,主控制在一个控制周期内发送的控制命令,关节节点能否都成功接收到,关系着系统设计的高效性,也影响着机器人能否按照指令完成相应的动作,因此,对该属性的验证十分有必要.该属性的形式化规范表示为

$$E\langle\rangle\ Master.sending\ \text{and}\ (forall(i:jid\_t)\ i\ imply\ Joint\_Controller(i).receiving).$$

即:*Master* 处于 *sending* 状态时,各个关节节点最终都能处于 *receiving* 状态.

- 无饥饿性

该属性表示不同优先级的节点最终是否都能成功地向主控制器反馈信息,这在机器人实际运行中也十分重要.因为主控制器需要得到关节节点反馈的位置或状态信息做下一步的规划,如果关节节点不能成功反馈信息,就会影响到机器人运动的精度.该属性的形式化表示为

$$E\langle\rangle\ (forall(i:\ jid\_t)\ i\ imply\ Joint\_Controller(i).ack\_waitting)\ \text{and}\ Master.receiveing.$$

- 下位关节节点之间不可通信

在机器人通信系统中,各个下位关节节点是相互独立的,它们之间是不进行通信的.在该系统模型中,也对这个属性进行了验证.该属性的形式化规范如下:

$$E\langle\rangle\ \text{not}\ (forall(i:jid\_t)\ Joint\_Controller(i).ack\_waitting\ \text{and}\ Joint\_Controller(i).receiving).$$

对所有关节节点而言,当一个节点处于反馈状态时,其他节点都不会处于接收状态.

- 同一时刻总线上只有一个节点传输数据

CAN 总线协议是一种采用非破坏性仲裁机制的协议,在同一时刻,只允许一个节点占有总线进行数据传输.该属性的形式化规范如下:

$$A[]\ \text{not}\ (forall(i:jid\_t)\ forall(j:jid\_t)\ Joint\_Controller(i).ack\_send\ \text{and}\ Joint\_Controller(j).ack\_send).$$

对于所有的关节节点,不存在同时有两个节点处于请求发送态.

- 存在同一时刻有多个节点请求发送数据

虽然 CAN 总线不允许同一时刻传输多个节点信息,但是仍然存在同一时刻多个节点向总线发送传输请求的情况.出现这种情况后,这些节点则需要参与总线仲裁,只有仲裁成功的节点才可以向总线发送或接受数据.

$$E\langle\rangle\ (forall(i:jid\_t)\ forall(j:jid\_t)\ Joint\_Controller(i).ack\_waitting\ \text{and}\ Joint\_Controller(j).ack\_waitting).$$

- 高优先级节点总能获得总线权

由于 CAN 总线的仲裁机制,使得高优先级的节点总能仲裁成功.基于这个情况,在这个系统模型中,把主控制器的优先级设为最高,这样能够保证主控制器能及时向关节控制器发送控制命令,保证机器人能够按照控制命令执行相关运动.该属性的形式化规范如下:

$$A[]\ \text{not}\ Arbitration(0).req\_fail.$$

4 基于 CAN 的机器人通信系统验证

4.1 功能正确性验证

为保证基于 CAN 的机器人通信系统能够正常且准确地通信,对该系统进行功能正确性验证十分重要.在基于抽取的属性基础上,在模型检测工具 UPPAAL 中对这些属性进行了验证,验证结果见表 2.

对基于 CAN 的机器人通信系统的功能正确性验证表明:采用本文所提供的形式化验证方法,能够有效地验证系统的设计逻辑正确,系统在不存在信息传输失败等不稳定因素下,能够正常工作.同时,实验结果也表明了本文所提供的验证方法是可行的,它能够很好地对系统的功能正确性进行验证.通过对属性在不同系统规模下进行的验证,我们发现:依据本文所提供的形式化验证框架以及所构建的时间自动机验证模型,当系统规模达到 20 个节点时,仍然可以得到验证结果.因此,实验结果也证明了该方法的有效性.但是在实验过程中发现:随着节点数的不断增加,参与总线仲裁的节点仲裁时延也越来越长.这在一定程度上会影响机器人通信的实时性,因

此,对基于 CAN 总线的机器人通信系统进行实时性分析十分必要.

Table 2 Verification results

表 2 验证结果

名称	结果
安全性	满足
活性	满足
无饥饿性	满足
下位关节节点之间不可通信	满足
同一时刻总线上只有一个节点传输数据	满足
同一时刻有多个节点请求发送数据	满足
高优先级节点总能获得总线权	满足

4.2 实时性分析与验证

CAN 总线的通信延迟时间可以分为 4 个部分,包括数据生成延迟、仲裁延迟、传输延迟和接收数据延迟.这里,CAN 总线的总延迟时间用 *Rm* 表示,仲裁延迟用 *Tm* 表示,传输延迟用 *Cm* 表示,在这 4 部分的延迟时间中,数据生成延迟和接收数据延迟主要由节点的主控制器产生,很大程度上取决于数据采集系统的设计方案和主控制器的执行速度等,按照现在的处理器执行速度而言,该部分时间可以忽略不计.因此,报文总延迟时间可以表示为

$$Rm=Tm+Cm.$$

其中,数据传输延迟受报文长度和总线上报文传输速度的影响,根据 CAN 数据帧的标准帧和扩展帧格式可以知道,数据帧包括 44 位(标准帧)或者 64 位(扩展帧)比特位.根据位填充的规定,每出现 5 个连续的相同的位,就要在其后面加一个填充位,但是界定符、应答段和结束段不需要加填充位,所以在这两种不同的帧格式中,会有 34 位或者 54 比特位是参与位填充的.在帧的数据段可以封装 1~8 个字节的信息,所以在封装有 *N* 个字节的数据帧中,最多可以填充的位数是(*N*×8+34(54)−1)/5.设定发送一个比特位的时间是 *Tbit*,那当分别发送标准帧和扩展帧的情况下,它们的传输时间公式可以表示为[18]

$$C1=\{N\times 8+44+(N\times 8+34-1)/5\}\times T,$$

$$C2=\{N\times 8+64+(N\times 8+54-1)/5\}\times T.$$

目前,CAN 总线上报文的最快传输速度可以达到 1Mbps,这种情况下,发送一位的时间是 1μs.综上所述,CAN 总线的通信时延主要取决于仲裁延迟,因此,对 CAN 通信实时性的影响也主要在于仲裁延迟.这里的仲裁延迟是指低优先级节点在发送报文时,因为仲裁失败并等待总线空闲而产生的延迟时间.本文中,重点考虑了仲裁延迟对 CAN 实时性的影响.

为记录节点的仲裁延迟,在关节控制器时间自动机中引入了局部 *clock* 变量 *x*,用来记录节点参与仲裁的时间,包括仲裁失败后等待的时间和重新仲裁的时间.我们首先针对采用静态优先级策略的 CAN 总线进行了实验,发现:当 CAN 总线上挂接 7 个节点时,能在允许时间范围内完成总线仲裁;但是随着挂接的节点数逐渐增加,优先级较低的节点最坏仲裁时延会不断增大,甚至会退出总线仲裁,进而无法获得总线权.这就严重影响了通信的实时性.

为了解决通信的实时性问题,对 CAN 进行了扩展,产生了利用时间触发的通信协议 TTCAN[19].TTCAN 是基于表的静态调度算法,其总线上的通信是按照矩阵周期的调度安排周期性循环进行的.在网络通信开始之前,需要针对整个网络拓扑结构和应用层协议综合考虑,制定出矩阵周期.TTCAN 协议虽然较 CAN 而言提高了实时性,但是它不具备灵活性,一旦网络完成搭建,便不允许增加或减少通信节点.基于这个原因,Zuberi 最早将 EDF 调度算法引入到 CAN 总线中[20],得出动态调度算法在软实时消息的调度上要优于静态调度算法,但是采用 EDF 算法非常容易引起优先级倒置的现象发生.后续又有很多文献针对 EDF 算法进行了改进,在此基础上得到很多优化的动态调度算法[21].

在这些研究的基础上,本文引入了易实现且不用频繁构建调度表的改进的动态优先级方法,该方法的主要

思想是:从 CAN 报文仲裁场的 11 位标识符中预留一个最高优先级系列,在仲裁过程中,当节点仲裁失败的次数达到一定值后,将该节点的优先级进行升级,提升到最高优先级,即,将预留的序列赋给该节点.为保证正常通信,当节点仲裁成功完成传输后,要将该节点的优先级恢复到原来的优先级.这种方法可以避免为提升节点优先级而导致出现多个节点拥有相同优先级的问题发生.

基于这个方法,在 UPPAAL 中重新对关节控制器进行了建模,如图 11 所示.在该时间自动机中引入了动态优先级策略,为记录节点仲裁失败的次数,引入了整型变量 *failNum*,本文中设定:当仲裁失败的次数为 3 次时,对该节点的优先级进行提升.在自动机中,该方法通过函数 *Raise_pri*()实现,当节点通过提升优先级而获得总线完成传输后,要恢复其优先级,保证通信正常运行.其优先级的恢复由函数 *reset_pri*()实现.

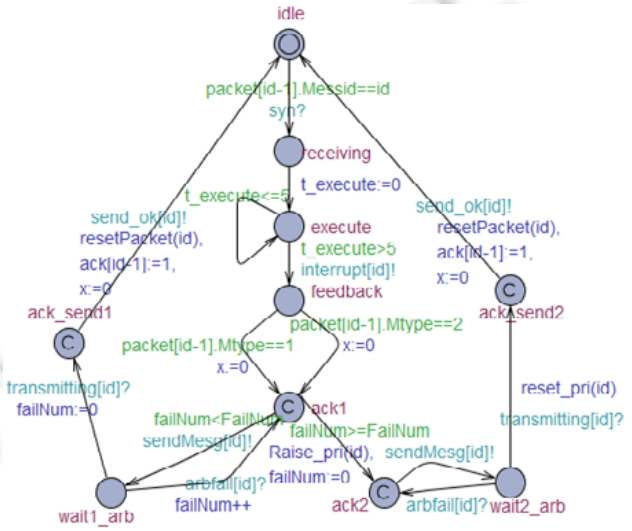

Fig.11 Module of joint_controller with dynamic policy

图 11 加入动态优先级后的关节控制器模型

在 UPPAAL 中,分别针对采用静态优先级策略和采用动态优先级策略的系统模型进行了实验,在实验中,不断增加总线节点的数量并记录节点完成仲裁的最坏仲裁延迟,实验结果如图 12 所示,图中横坐标表示总线上挂接的节点数,纵坐标表示节点完成仲裁的最坏仲裁延迟.其中,曲线 *s*[*n*]表示采用静态优先级策略的实验结果,曲线 *d*[*n*]代表采用动态优先级策略后的实验结果.

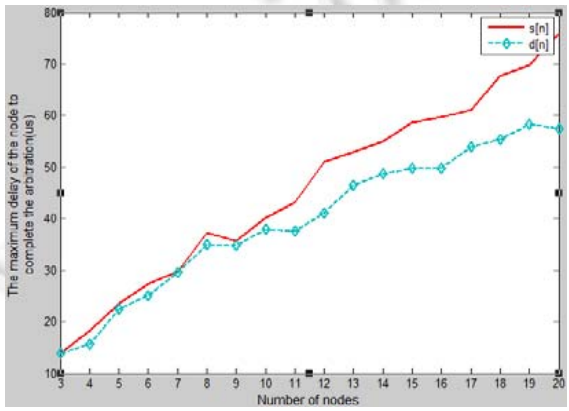

Fig.12 Results of the two methods

图 12 两种方法的实验对比结果

从实验结果可以看出:采用静态优先级策略时,当总线上的节点数达到 11 个后,最坏仲裁时延增长速率变大,当达到 17 个以后速率增长迅速变大,这表明当总线上的负载量越来越大时仲裁延迟会越来越大,这就在一定程度上影响了机器人通信的实时性.部署动态优先级后,这种情况明显改善,即使节点数达到 20 个,节点最坏仲裁延迟增长速率也不是特别迅速,并且相对比较缓和,这就说明采用动态优先级策略后能有效地减小节点仲裁产生的延迟,进而能够提升基于 CAN 的机器人通信的实时性.能够满足机器人的通信需求,并且还可以在一定程度上提升总线的负载,对服务型机器人的发展提供一定的保障.

5 相关工作

在参考了大量关于 CAN 协议调度方面的参考文献后,受这些研究的启发,想到在基于 CAN 的机器人通信系统中加入动态优先级策略,进而分析加入动态优先级策略后对该系统实时性的影响.其中,一些重要参考文献如下.

文献[22]针对工业实时通信中标准 CAN 总线协议在网络拥塞情况下出现某些帧无法发送和丢帧等问题,提出了一种静态和动态相结合的调度算法.该算法首先建立一张帧信息表;接着,利用该表来确定当前帧的类型;然后,通过表中的优先级进行实时传输.如果当前信道上出现相同优先级的帧,则使用动态调度算法进行调度传输.

文献[21]中,Burns 等人对 CAN 的原始调度性进行了分析,并指出了不足之处,即:在实际应用中,消息最终可能会错过最后截止时间而无法实现调度.作者针对这个问题,在文中提供了对原有方法的修改分析,并且引入了一种适用于 CAN 的最优优先级排序的方法.

文献[23]中,Davis 等人提出 CAN 的现有可调度性分析基于最高优先级,而在实际应用中,一些 CAN 设备驱动程序实现了 FIFO 而不是基于优先级的队列.在文中,作者对响应时间和 CAN 消息的最优优先级分配策略进行了分析,提出:其中一些节点使用 FIFO 队列,而其他节点使用优先队列.但是引入 FIFO 调度后,会对 CAN 的实时性造成影响.

由于 CAN 采用的是固定优先级策略和非破坏性仲裁机制,所以当总线上的负载量很大时,总是高优先级的节点获得总线而低优先级节点不能仲裁成功,进而会导致这些低优先级节点的饥饿现象.文献[24]中,Anwar 等人针对低优先级消息饥饿的问题提出一个方案,为系统分配一个可调节的消息 ID 窗口来识别 CAN 总线上的消息,代替了原来 CAN 协议中用单个标识符标识消息 ID 的方法.在该方案中,消息的优先级将在运行时进行调整,而不会对消息的内容或含义产生任何影响.该方法有助于在控制器局域网上实现动态消息调度.通过实验证明,该方案能有效地降低 CAN 网络上的饥饿概率.

文献[25]中,Murtaza 等人同样针对低优先级节点可能面临的饥饿现象提出了一种方法,就是在总线上引入一个 *master* 节点,由这个节点来解决饥饿现象.在文中,作者也引入了动态优先级的思想,他的主要做法是为每个节点分配两个不同的优先级,并将所有的这些节点信息都存入 *master* 节点中,当总线上出现饥饿现象时,该 *master* 节点根据记录的信息动态地改变饥饿节点的优先级.当总线正常传输时,*master* 节点只是不断监测总线,只有监测到饥饿节点时,该 *master* 节点才会处于活动状态.在整个通信过程中,*master* 节点保证每个节点都能够公平的占用总线.

6 总 结

本文提出了基于CAN的现场总线型控制系统的形式化验证方法,将基于CAN的机器人系统与模型检测方法连接起来,给出系统形式化架构和构建时间自动机模型的方法步骤,并提供了各个模块在 UPPAAL 中的具体实现;应用该形式化方法对基于 CAN 的机器人通信系统进行了功能正确性验证,在验证的基础上,对 CAN 的实时性进行了分析,发现采用常规的 CAN 总线不能保证日益发展的机器人通信的实时性,因此,在模型中部署了改进的动态优先级策略,并对模型的实时性进行了分析和验证.实验表明:采用动态优先级不仅能改善常规CAN总线的实时性,还能够在一定程度上改善CAN总线的负载.该研究表明:设计人员可以根据应用在CAN通

信的设计中通过改进仲裁策略,动态调整优先级来满足关键应用的实时性要求.此外,本文还分享了一些建模方面的经验.本文的不足之处是没有考虑传输失败后重传的情况,因为重传机制也会在一定程度上增加 CAN 总线的负载量,对低优先级节点的仲裁产生影响.

未来我们会将传输失败因素考虑进去,并借助概率模型检测技术分析在不同传输失败概率情况下对 CAN 总线负载的影响.进一步完善动态优先级策略,使得该方法应用范围更广泛,算法更健壮,并分析和验证引入传输失败概率后,该方法的健壮性和效果.

References:

[1] Nilsson DK, Larson UE, Picasso F, Jonsson E. A first simulation of attacks in the automotive network communications protocol flexray. In: Proc. of the Int'l Workshop on Computational Intelligence in Security for Information Systems (Cisis 2008). Genova: DBLP, 2009. 84–91. [doi: 10.1007/978-3-540-88181-0_11]

[2] Wang DQ, Gao SY, Chen YQ, Wang Y, Liu Q. Intelligent control system based on CAN-bus for car doors and windows. In: Proc. of the Int'l Conf. on Anti-Counterfeiting, Security, and Identification in Communication. IEEE, 2009. 242–245. [doi: 10.1109/ICASID.2009.5276906]

[3] Sabelhaus AP, Bruce J, Caluwaerts K, Manovi P. System design and locomotion of SUPERball, an untethered tensegrity robot. In: Proc. of the IEEE Int'l Conf. on Robotics and Automation (ICRA). IEEE, 2015. 2867−2873. [doi: 10.1109/ICRA.2015.7139590]

[4] Kaneko K, Harada K, Kanehiro F, Miyamori G. Humanoid robot HRP-3. In: Proc. of the IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems. IEEE, 2008. 2471−2478. [doi: 10.1109/IROS.2008.4650604]

[5] Kim JY, Park IW, Lee J, Kim MS, Cho BK, Oh JH. System design and dynamic walking of humanoid robot KHR-2. In: Proc. of the IEEE Int'l Conf. on Robotics and Automation. IEEE, 2005. 1431−1436. [doi: 10.1109/ROBOT.2005.1570316]

[6] Li XY, Huang M, Zhan J, Ni YL, Pang FY. CANoe-Based modeling and simulation for heavy lorry CAN bus network. In: Proc. of the AsiaSim 2012. Berlin, Heidelberg: Springer-Verlag, 2012. 107−114. [doi: https://doi.org/10.1007/978-3-642-34387-2_13]

[7] Liu T, Wang SL, Zhan NJ. Safety verification of trajectory planning for multiple robots. Ruan Jian Xue Bao/Journal of Software, 2017,28(5):1118−1127 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5207.htm [doi: 10.13328/j.cnki.jos.005207]

[8] Resten Y, Maler O, Marcus M, Pnueli A, Shahar E. Symbolic model checking with rich assertional languages. Theoretical Computer Science, 2001,256(1-2):93−112. [doi: https://doi.org/10.1016/S0304-3975(00)00103-1]

[9] Xiao D, Zhu YF, Liu SL, Wang DX, Luo YQ. Applied Mechanics & Materials. 2015,716−717:1382−1386. [doi: 10.4028/www.scientific.net/AMM.716-717.1382]

[10] Alur R, Dill DL. A theory of timed automata. Theoretical Computer Science, 1994,126(2):183−235. [doi: 10.1016/0304-3975(94)90010-8]

[11] Berendsen J, Gebremichael B, Vaandrager FW, Zhang MM. Formal specification and analysis of zeroconf using uppaalS. ACM Trans. on Embedded Computing Systems, 2011,10(3):1−32. [doi: 10.1145/1952522.1952527]

[12] Pan C, Guo J, Zhu LF. Modeling and verification of CAN bus with application layer using UPPAAL. Electronic Notes in Theoretical Computer Science, 2014,309:31−49. [doi: https://doi.org/10.1016/j.entcs.2014.12.004]

[13] Gu JS, Silva CWD. Development and implementation of a real-time open-architecture control system for industrial robot systems. Engineering Applications of Artificial Intelligence, 2004,17(5):469−483. [doi: 10.1016/j.engappai.2004.03.010]

[14] Behrmann G, David A, Larsen KG. A tutorial on UPPAAL. In: Bernardo M, ed. Proc. of the Formal Methods for the Design of Real-Time Systems. Springer-Verlag, 2004. 200−236. [doi: 10.1007/978-3-540-30080-9_7]

[15] Dai SX, Hong M, Guo B, Yang QH, Huang W, Xu BP. Schedulability analysis model for multiprocessor real-time systems using UPPAAL. Ruan Jian Xue Bao/Journal of Software, 2015,26(2):279−296 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4781.htm [doi: 10.13328/j.cnki.jos.004781]

[16] Battiston A. Software in C++ for communication between CAN bus and ROS in a robot vehicle [Ph.D. Thesis]. Padua University, 2014.

[17] Behrmann G, David A, Larsen KG. A tutorial on uppaal. In: Proc. of the Formal Methods for the Design of Real-Time Systems. Berlin, Heidelberg: Springer-Verlag, 2004. 200–236.

[18] Dong YK. Research on scheduling algorithm based on CAN bus [Ph.D. Thesis]. Beijing: Tsinghua University, 2011 (in Chinese).

[19] Führer T, Müller B, Dieterle W, Hartwich F, Hugel R, Walther M, Gmbh RB. Time triggered communication on CAN. In: Proc. of the Time Triggered CAN (TTCAN). Bibliogr, 2001.

[20] Zuberi KM, Shin KG. Non-Preemptive scheduling of messages on controller area network for real-time control applications. In: Proc. of the 1st IEEE Real-Time Technology and Applications Symp. (RTAS'95). Chicago: IEEE Computer Society, 1995. 240−249.

[21] Davis RI, Burns A, Bril RJ, Lukkien JJ. Controller area network (CAN) schedulability analysis: Refuted, revisited and revised. Real-Time Systems, 2007,35(3):239−272. [doi: 10.1007/s11241-007-9012-7]

[22] Liu JF, Gui WH, Huang ZW, *et al*. Study of the application on a scheduling algorithm of CAN bus in locomotive brake. Journal of Chinese Computer Systems, 2009,30(1):183−187 (in Chinese with English abstract).

[23] Davis RI, Kollmann S, Pollex V, *et al*. Controller area network (CAN) schedulability analysis with FIFO queues. In: Proc. of the Euromicro Conf. on Real-Time Systems. IEEE Computer Society, 2011. 45−56. [doi: 10.1109/ECRTS.2011.13]

[24] Anwar K, Khan ZA. Dynamic priority based message scheduling on controller area network. In: Proc. of the Int'l Conf. on Electrical Engineering. IEEE, 2007. 1−6. [doi: 10.1109/ICEE.2007.4287302]

[25] Murtaza AF, Khan ZA. Starvation free controller area network using master node. In: Proc. of the Int'l Conf. on Electrical Engineering. IEEE, 2008. 1−6. [doi: 10.1109/ICEE.2008.4553945]

附中文参考文献:

[7] 刘涛,王淑灵,詹乃军.多机器人路径规划的安全性验证.软件学报,2017,28(5):1118−1127. http://www.jos.org.cn/1000-9825/5207.htm [doi: 10.13328/j.cnki.jos.005207]

[15] 代声馨,洪玫,郭兵,杨秋辉,黄蔚,徐保平.多处理器实时系统可调度性分析的 UPPAAL 模型.软件学报,2015,26(2):279−296. http://www.jos.org.cn/1000-9825/4781.htm [doi: 10.13328/j.cnki.jos.004781]

[18] 董寅康.基于 CAN 总线的调度算法的研究[博士学位论文].北京:清华大学,2011.

[22] 刘剑锋,桂卫华,黄志武,等.一种 CAN 总线调度算法在机车制动机上的应用研究.小型微型计算机系统,2009,30(1):183−187.

孟瑶(1987－),女,河北保定人,硕士,主要研究领域为机器人系统软件安全,计算机网络协议分析.

李晓娟(1968－),女,博士,教授,CCF 专业会员,主要研究领域为系统形式建模与分析,机器人系统软件安全,计算机网络协议分析.

关永(1966－),男,博士,教授,博士生导师,CCF 专业会员,主要研究领域为高可靠嵌入式系统,形式化验证.

王瑞(1981－),女,博士,教授,CCF 专业会员,主要研究领域为机器人安全验证.

张杰(1967－),女,副教授,主要研究领域为嵌入式系统,形式化验证.