无人驾驶汽车决策系统的规则正确性验证

刘斌斌[1] 刘万伟[2] 毛晓光[2] 董 威[2]
(国防科学技术大学计算机学院 长沙 410073)[1]
(国防科学技术大学计算机学院计算机科学与技术系 长沙 410073)[2]

摘 要 无人驾驶车辆技术是当前科学研究的重点领域之一,目前无人车决策系统的开发过程中面临着安全性不足的问题。针对该问题,提出了验证驱动的基于代码自动生成的无人车决策系统开发框架。该框架利用模型检验技术对无人车决策系统进行环境建模,通过验证可以发现无人车决策系统的设计过程中不易发觉的缺陷,解决其安全性不足的问题,同时能够将安全检查与软件开发同步,降低其维护成本。基于该框架,设计并实现了无人车决策系统辅助开发工具 UNMANNED_RULE_EDIT(URE),目前该工具已初步应用于国内某单位研制的无人车上,为其开发研制工作提供了帮助。

关键词 无人驾驶汽车,决策系统,模型检验,环境建模
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.04.015

Correctness Verification of Rules for Unmanned Vehicles' Decision System

LIU Bin-bin[1] LIU Wan-wei[2] MAO Xiao-guang[2] DONG Wei[2]
(School of Computer, National University of Defense Technology, Changsha 410073, China)[1]
(Department of Computer Science and Technology, School of Computer, National University of Defense Technology, Changsha 410073, China)[2]

Abstract The technology of unmanned vehicle is one of the key areas of current scientific research. However, the development of decision system is facing the problem of insufficient security. Aiming at this problem, this paper presented a solution framework of verification-driven unmanned vehicle's decision system based on automatic code generation. We introduced the technology of model checking to model the environment of unmanned vehicles. So this framework can find the defects that hide deeply during the designing process. With this method, the problem of insufficient security can be solved, which can also reduce the maintenance cost by synchronizing the process of software development with security checking. The tool named UNMANNED_RULE_EDIT (URE) was also implemented based on this framework. Now the tool has been tentatively used in an unmanned vehicle to support their work.

Keywords Unmanned vehicles, Decision system, Model checking, Environment modeling

1 引言

随着科学技术的发展,无人车技术日渐成为科学研究的重点领域之一。无人车技术的核心在于决策系统的开发,其开发过程涉及的变量多,迁移规则复杂,开发和维护工作面临诸多困难,主要体现在以下两个方面:

1)早期的系统开发过程主要由人工编写代码。人工编写代码的时间开销大,开发成本高,且代码可读性和一致性差,极易出现逻辑和时序错误;其次,由于程序耦合度高,决策系统的可拓展性和可维护性差。

2)决策系统的安全性无法保证。无人车系统涉及的参数众多,其取值往往由开发者根据经验人为提供,具有主观性,容易存在缺陷或错误;同时,决策规则间可能存在交集,导致发生决策冲突,造成安全隐患;另外,系统的开发与安全检查过程不同步,系统开发完成后再进行安全检查会增大修正错误所付出的代价。

针对以上问题,本文提出了验证驱动的基于代码自动生成的无人车决策系统开发框架,并基于此设计并实现了一套无人车决策系统辅助开发工具 UNMANNED_RULE_EDIT (URE),为无人车决策系统的开发与维护提供了帮助。系统总体框架结构如图1所示。

首先,URE 工具通过引入代码自动生成技术,解决了人工编写代码存在的问题。开发者只需输入无人车决策系统中的若干规则,工具就会自动生成中间语言代码(简称中间码),进而自动生成可执行代码。可执行代码可直接整合到无人车决策系统的决策代码中使用。通过引入代码自动生成技术,

到稿日期:2015-11-30 返修日期:2016-02-21 本文受国家自然科学基金(60873120),国家重点基础研究发展规划(973)(2009CB723803)资助。
刘斌斌(1992—),男,硕士生,主要研究方向为计算机软件与理论,E-mail:liubinbin0032@sina.com;刘万伟(1980—),男,博士,副教授,主要研究方向为时序逻辑、模型检验等;毛晓光(1970—),男,教授,博士生导师,主要研究方向为形式化方法、软件工程等;董 威(1976—),男,教授,博士生导师,主要研究方向为高可信软件技术、软件工程等。

开发人员从书写代码转变为编写决策规则，实现了规则和应用程序的分离，降低了程序出错的几率，同时也减少了维护工作的成本和难度。该部分工作已由课题组早期工作完成，可参考文献[1]。

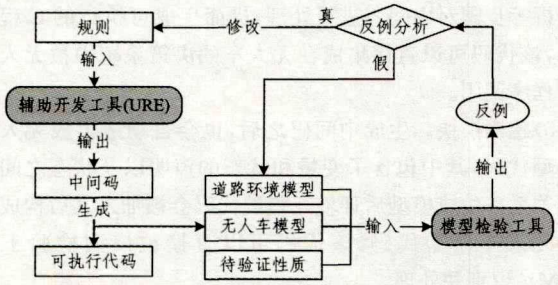

图 1　决策系统开发框架

解决无人车决策系统开发过程中安全性不足的问题是本文的主要工作。无人车决策系统在理论上是一个有穷的状态迁移系统，可以很好地应用模型检验技术来进行形式化验证。由于无人车决策系统开发过程中存在主观性问题，因此无法在数学上给出严密的结论来证明其在任何环境下运行都不会出现问题。而模型检验技术的优点就在于能够遍历所有状态空间来寻找反例，如果没有发现反例，则可以给出可靠的结论证明决策系统是正确的。因此，引入模型检验技术解决该问题是必然选择。

URE 工具生成规则对应的中间码后，在生成可执行代码的同时也会利用相同的原理生成无人车的模型代码。由于无人车决策系统是一个开放系统，单独的无人车模型无法独立运行，它需要从环境中获取信息来进行决策，因此还需要对无人车运行时所处的环境进行建模。在对环境进行建模之后，加入待验证的性质，就可以交给模型检验工具进行验证。对模型检验工具给出的结果进行分析，可以发现设计过程中存在的缺陷和错误。

2　模型检验技术

模型检验的研究起始于 80 年代初，Clarke 和 Emerson 等人提出了用于描述并发系统性质的 CTL 逻辑，设计了检测有穷状态系统是否满足给定 CTL 公式的算法，并实现了一个原型系统。此工作为对并发系统性质进行自动验证开辟了一条新的途径，成为了近 20 年来计算机科学基础研究的一个热点。模型检验已被应用于计算机硬件、通信协议、控制系统、安全认证协议等方面的分析与验证中，并取得了令人瞩目的成就[2]。

模型检验的基本思想就是通过计算机工具执行算法来验证系统的正确性。用户输入一个系统的模型的描述（系统可能的行为）和一个需求规约的描述（用户想要的行为），然后将证明过程留给机器完成。如果执行过程中发现错误，模型检验工具就会提供一个反例，该反例表明在哪种情况下会发生错误。反例中包含一个模型执行了一个用户不希望发生的行为的情景，因此，反例可以证明一个模型是错误的，并且提供模型需要修正的证据，这需要用户在继续下一步操作之前定位这个错误并对模型进行修正。如果没有发现任何错误，用户也可以优化它的模型描述（例如，通过考虑更多的设计决策，使模型变得更加具体/真实）并重启验证过程。模型检验技术的框架如图 2 所示。

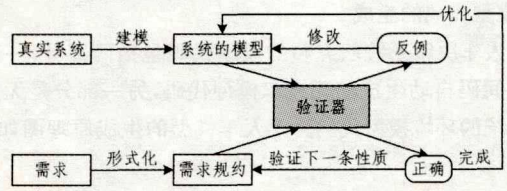

图 2　模型检验技术框架

3　无人车决策系统的规则正确性验证方法

3.1　无人车决策系统的环境建模方法

无人车决策系统是一个开放式系统，需要从它所运行的环境中获取一些重要信息来进行决策，如道路信息以及速度、方向等。单独的无人车模型无法独立运行，因此对无人车的环境进行建模是有必要的。无人车决策系统的环境建模就是模拟无人车的运行环境，为无人车系统提供变量的各种取值情况，再通过模型检验技术搜索状态空间来验证是否在某些变量的取值下无人车的运行会出现安全隐患。

无人车的环境建模方法如图 3 所示，环境建模的主要来源有两个：1）中间码部分，中间码包含了无人车运行时所需的所有信号变量的声明，这一部分会直接转换到环境模型代码中；2）真实的道路物理信息约束，该部分会根据真实情况对无人车决策系统运行时所涉及到变量的取值范围进行约束。这样就构成了一个状态空间，模型检验工具就能够在这个状态空间中进行验证工作。

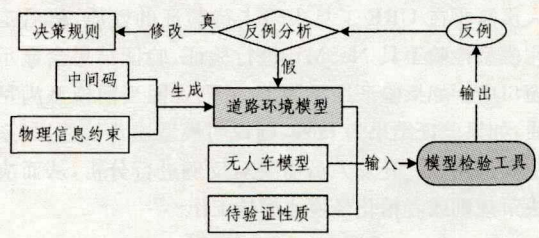

图 3　环境建模方法示意图

环境建模不是一次性的工作，是对反例进行分析，反复迭代、反复精化的过程。反例迭代过程如下：如果验证结果为假，工具就会提供一个反例，反例中包含了无人车系统中各个变量取值的详细时序信息，开发人员需要对反例进行分析。如果变量的这种取值情况在物理环境中是真实存在的，即符合物理约束，那么该条反例为真反例，进而该条反例就可以用来指导修改无人车系统的决策规则；如果变量的这种取值情况在真实物理环境中不会发生，那么该条反例为假反例，则说明环境模型中存在缺陷，为了避免该条反例再次出现，就需要在环境模型中以不变式的形式引入新的约束条件来排除此种取值情况。通过验证不同的性质，反复迭代上述过程，就可以逐步精化环境建模的工作。

通过对无人车运行环境的建模，可以模拟无人车运行时的道路物理环境，为无人车的决策过程提供参数等信息。无人车的环境模型与无人车规则模型集成在一起就构成了完整的无人车运行模型，可以用来验证无人车在各种场景中的运

行情况。从而通过模型检验技术解决无人车系统在规则设计过程中难以察觉的缺陷和错误等问题。

3.2 模型代码的生成

无人车决策系统的完整的模型代码包括两部分，一部分是由中间码自动生成的无人车模型代码，另一部分是无人车决策系统的环境模型代码。无人车模型的生成原理图如图 4 所示。

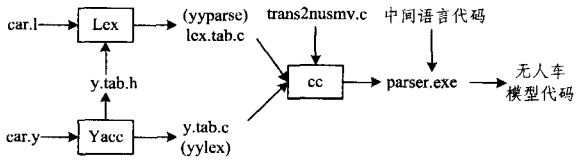

图 4 无人车模型代码生成原理

开发人员首先以表格的形式在工具 URE 中录入规则，规则包含 3 部分：信号变量声明、状态声明以及迁移关系声明。之后工具会自动地将其转换为中间语言代码。中间语言代码作为模型生成的输入文件，经过词法分析器 Lex 和语法分析器 Yacc 分别进行词法分析和语法分析，构造出语法树，再由程序 trans2nusmv. c 处理之后就自动地生成了无人车的模型代码。

生成无人车的模型代码后，将无人车的环境建模代码与它进行合成，就完成了无人车决策系统的整体建模工作，之后再加入待验证的性质就可以将证明过程交给模型检验工具进行验证。

3.3 无人车决策系统的验证过程

完成无人车决策系统的建模后，即可进行模型检验工作。开发人员需要在 URE 工具中输入待验证的性质，然后工具会调用模型检验工具 NuSMV 进行验证，验证结果会显示在工具窗口中。如果验证结果为 true，则说明当前模型满足该条性质；如果验证结果为 false，则说明模型违背该性质，系统会给出反例路径。开发人员需要对反例进行分析，从而决定修改决策规则或者精化环境建模的工作。

4 工具的实现方法

基于决策系统的开发框架，本文设计并实现了无人车决策系统辅助开发工具 UNMANNED_RULE_EDIT。该工具包含以下几个模块：1)规则模块；2)图示模块；3)中间码模块；4)代码模块；5)建模模块；6)验证模块。下面介绍各个模块的功能。

1)规则模块。规则模块包含 3 个部分：变量声明、状态声明和迁移关系声明。开发人员可以在此模块中对无人车的规则进行编辑与录入，也可以采用从文件中直接导入中间码文件的形式录入规则。

2)图示模块。系统根据开发人员在规则模块中录入的规则可以生成规则迁移图，用来表示状态与状态之间的迁移关系以及相对应的迁移条件；另外，开发人员可以通过直接绘图的方式编辑迁移关系图，系统也可以将其转化为规则列表中的规则。

3)中间码模块。用户在完成对决策规则的录入之后，系统会自动地生成相对应的中间码表示形式，中间码包含了信号变量和状态的声明，以及状态之间的迁移关系和相应的迁移条件等信息，中间码也是接下来转换为可执行 C 代码和无人车模型代码的输入文件。

4)代码模块。生成中间码之后，系统会经过词法分析、语法分析等步骤对中间码进行处理，进而生成可执行的 C 程序代码，该代码可以直接集成在无人车的决策系统中被无人车决策程序调用。

5)建模模块。生成中间码之后，也会自动地生成无人车的模型代码，其中包含了变量和状态的声明以及状态之间的迁移关系。生成模型后环境建模的代码会被加入其后构成完整的无人车模型代码，该代码可以直接被模型检验工具 NuSMV 识别并处理。

6)验证模块。验证模块主要包含两部分内容：性质部分和验证结果部分。开发人员可以在性质文本框中输入待验证的性质，然后点击验证按钮，系统便会自动地调用模型检验工具给完整的无人车模型代码加上待验证的性质，交给模型检验工具 NuSMV 进行验证工作。模型检验生成的结果会在第二部分——验证结果的窗口中显示出来。如果验证结果为真，则会显示 true；如果验证结果为假，则显示 false 并给出反例。

结束语 本文提出了验证驱动的基于代码自动生成的无人车决策系统开发框架，并基于此设计并实现了无人车决策系统辅助开发工具 UNMANNED_RULE_EDIT(URE)。通过对无人车决策系统进行环境建模，利用模型检验技术进行验证，解决了无人车决策系统规则设计中存在的缺陷和错误，增强了其安全性，同时能够将开发过程与安全检查过程同步，大大降低了其维护成本。无人车决策系统辅助开发工具 URE 为无人车系统的开发工作提供了极大的便利，提高了开发效率，降低了缺陷风险，保证了软件质量。另外，本课题提出的环境建模方法具有一定的通用性，对于其他系统开发(如列控、机器人、无人平台等)都提供了很大的借鉴价值。

目前的开发框架还不够完善，反例分析部分的工作暂时还停留在人工分析阶段，接下来将集中于将反例分析自动化，并将其自动反馈到规则的修改中，这也将是我们下一步工作的重点。

参 考 文 献

[1] LAN Y, LIU W W, DONG W, et al. Research on the Rule Editing and Code Generation for the High-Level Decision System of Unmanned Vehicles [J]. Computer Engineering and Science, 2015, 37(8): 1510-1516. (in Chinese)
兰韵，刘万伟，董威，等. 无人驾驶汽车决策系统的规则描述与代码生成方法[J]. 计算机工程与科学，2015，37(8)：1510-1516.

[2] LIN H M, ZHANG W H. Model Checking: Theories, Techniques and Applications [J]. Journal of Electronics, 2002, 30(12): 1907-1912. (in Chinese)
林惠民，张文辉. 模型检测：理论，方法与应用[J]. 电子学报，2002，30(12)：1907-1912.

[3] WANG R. Research on SAT based Symbolic Model Checking [D]. Changsha: University of Defense Technology, 2014. (in Chinese)

(下转第 113 页)

表 1　实验原始数据

漏洞名称	标准漏洞数	SDTPM			PMD		
		规则行数	检测时间/s	漏洞检测数	规则行数	检测时间/s	漏洞检测数
CWE78_OS_Command_Injection	95	6	0.028	95	23	1.080	95
CWE89_SQL_Injection	172	9	0.031	172	35	2.233	158
CWE209_Information_Leak_Error	36	7	0.020	36	20	0.802	36
CWE327_Use_Broken_Crypto	18	8	0.015	18	24	0.094	18
CWE330_Insufficiently_Random_Values	18	8	0.016	18	27	0.984	18
CWE547_Hardcoded_Security_Constants	18	9	0.013	18	23	0.916	18
CWE597_Wrong_Operator_String_Comparison	18	7	0.018	18	21	0.354	18
CWE614_Sensitive_Cookie_Without_Secure	18	14	0.036	17	0	0	0
CWE698_Redirect_Without_Exit	35	13	0.041	34	0	0	0
CWE764_Multiple_Locks	4	13	0.047	6	0	0	0
CWE765_Multiple_Unlocks	2	14	0.041	4	0	0	0
CWE832_Unlock_Not_Locked	4	12	0.034	8	0	0	0

表 2　实验统计数据

	标准漏洞数	漏洞检测数	误报数	漏报数	误报率/%	漏报率/%	准确率/%	检测种类数	内存使用情况/MB
SDTPM	438	444	8	2	1.8	0.5	98.2	12	48
PMD	375	361	0	14	0	3.7	100	7	15

注：误报率＝误报数/漏洞检测数；漏报率＝漏报数/标准漏洞数；准确率＝(漏洞检测数－误报数)/漏洞检测数

从表 1、表 2 中可以看出，SDTPM 系统在规则的简洁性、检测效率、检测结果的漏报率以及检测漏洞种类方面都优于 PMD，两者的准确率相差不大。SDTPM 系统采用基于模式匹配的安全漏洞检测方法，在检测过程中同时考虑到源码中可能存在的跨函数、分支路径和环路等漏洞产生的场景，使得检测结果的漏报率大大降低。但在内存使用情况方面，SDTPM 却大大超出了 PMD，这主要是因为 SDTPM 使用了源代码中间表示存储器，占用内存较大，且随着源代码规模的增加，存放源代码的中间表示所占用的内存空间也会越来越大。SDTPM 系统检测的漏报率和效率是以空间的消耗为代价的，这也是本系统的不足之处。

结束语　通过分析常用的静态分析方法，在吸取其优点的基础上，提出了一种基于模式匹配的安全漏洞检测方法，并实现了原型系统 SDTPM。实验结果表明，SDTPM 系统检测漏洞时具有漏报率低、扩展性好的特点。同时从实验结果来看，较大的空间消耗成为了影响检测程序源代码规模的一个制约条件。后续工作：1)研究针对大规模程序源代码的中间表示的存储方式来降低系统内存的消耗；2)对源代码的中间表示进行改进，去掉不必要的数据结构和源代码信息以减小内存消耗。

参 考 文 献

[1] JUENEMAN R R. Securing wireless medicine confidentiality, integrity, nonrepudiation, & malware prevention[C]//2011 8th International Conference & Expo on Emerging Technologies for a Smarter World (CEWIT). IEEE, 2011: 1-5.

[2] ALBREIKI H H, MAHMOUD Q H. Evaluation of static analysis tools for software security[C]// 2014 10th International Conference on Innovations in Information Technology (INNOVATIONS). IEEE, 2014: 93-98.

[3] EGELE M, SCHOLTE T, KIRDA E, et al. A survey on automated dynamic malware-analysis techniques and tools [J]. ACM Computing Surveys (CSUR), 2012, 44(2): 6.

[4] STANCU C, WIMMER C, BRUNTHALER S, et al. Comparing points-to static analysis with runtime recorded profiling data[C]// Proceedings of the 2014 International Conference on Principles and Practices of Programming on the Java platform: Virtual machines, Languages, and Tools. ACM, 2014: 157-168.

[5] CHELF B, ENGLER D, HALLEM S. How to Write System-specific, Static Checkers in Metal[C]// Proceedings of the 2002 ACM SIGPLAN-SIGSOFT workshop on Program Analysis for Software Tools and Engineering. Charleston, SC, USA. ACM, 2003: 51-60.

[6] HALLEM S, CHELF B, XIE Y, et al. A system and language for building system-specific, static analyses[C]// Proceedings of the ACM SIGPLAN Conference on Programming language Design and Implementation. ACM, 2002: 69-82.

[7] ARAUJO J E, SOUZA S, VALENTE M T. Study on the relevance of the warnings reported by Java bug-finding tools [J]. IET Software, 2011, 5(4): 366-374.

[8] KIM Y, KIM M, KIM Y J, et al. Industrial application of concolic testing approach: A case study on libexif by using CREST-BV and KLEE[C] // 2012 34th International Conference on Software Engineering (ICSE). IEEE, 2012: 1143-1152.

(上接第 74 页)

王瑞. 基于 SAT 的符号化模型检验技术研究[D]. 长沙：国防科学技术大学，2014

[4] YANG J J. Study on SAT-based Bounded Model Checking and its Applications [D]. Guangzhou: Sun Yat-sen University, 2008. (in Chinese)
杨晋吉. 基于 SAT 的有界模型检验及其应用研究[D]. 广州：中山大学，2008.

[5] SHEN S Y. Explaining Counter Example of Model Checking [D]. Changsha: University of Defense Technology, 2005. (in Chinese)
沈胜宇. 模型检验的反例解释[D]. 长沙：国防科学技术大学，2005

[6] PNUELI A. The Temporal Logic of Programs [C]//Proc. of 18th IEEE Symposium on Foundation of Computer Science (FOCS' 77). 1977: 46-57.

[7] EMERSON E A, CLARKE E M. Characterizing Correctness Properties of Parallel Programs Using Fixpoints [C]// Proc. of the 7th Int. Colloquium on Automata, Languages and Programming (ICALP'80). 1980: 169-181.