中不断地获取新的输入和产生结果.这个组件中的事件约束可以用 CCSL 描述,图 3 描绘了这些事件的 9 个约束.虚线箭头部分表示了两个中间事件 tmp_1 和 tmp_2 与其他事件的之间的约束.譬如,我们可以得到:

$$tmp_1 \triangleq in_1+in_2.$$

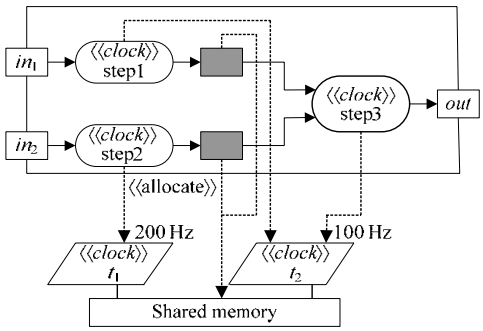

Fig.2 A component in a practical application

图 2 一个实际应用的组件

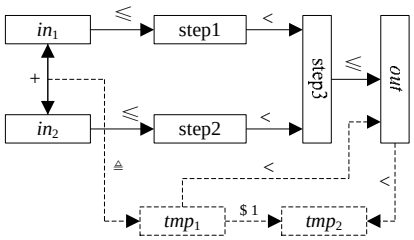

Fig.3 An abstraction as a clock constraint specification

图 3 抽象成时钟约束规范

通过设置边界值 n,我们将这 9 个约束转换成上述的求解公式检测其可满足性.表 5 展示了用 Z3 求解的结果,若公式满足,则表示存在一个死锁.否则表示不存在.另外,我们分别用 $tmp_1 \triangleq in_1 \wedge in_2$ 和 $tmp_1 \triangleq in_1 \vee in_2$ 代替 $tmp_1 \triangleq in_1+in_2$ 检测该公式的满足性.

Table 5 Verification result on the deadlock in the example

表 5 验证死锁的结果

上界	$tmp_1 \triangleq in_1+in_2$		$tmp_1 \triangleq in_1 \wedge in_2$		$tmp_1 \triangleq in_1 \vee in_2$	
	死锁?	时间(s)	死锁?	时间(s)	死锁?	时间(s)
1	Yes	4.40	No	0.01	No	0.01
10	Yes	4.20	No	0.60	No	0.30
20	Yes	14.20	No	2.40	No	1.70
30	Yes	98.60	No	6.10	No	4.10
40	Yes	50.00	No	7.00	No	7.40
50	Yes	141.10	No	10.00	No	12.40

从表中我们看到:$tmp_1 \triangleq in_1+in_2$ 造成了死锁,而 $tmp_1 \triangleq in_1 \wedge in_2$ 和 $tmp_1 \triangleq in_1 \vee in_2$ 则没有检测到死锁.图 4 表示了在用 $tmp_1 \triangleq in_1+in_2$ 关系且上界设为 10 时,相关的时钟从第 1 步~第 10 步的行为及在第 11 步时发生死锁的情景.

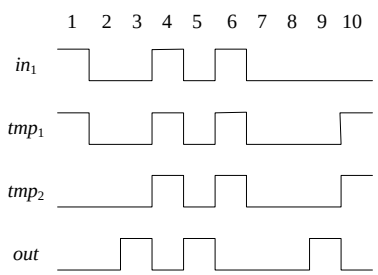

Fig.4 Clocks with deadlock

图 4 存在死锁的时钟集合

在第 11 步时,若 in_1 发生,因为 $tmp_1 \triangleq in_1+in_2$ 约束,tmp_1 必须发生,从而导致 tmp_2 也必须发生.由于 $out \triangleq tmp_1\$1$, out 在此步不会发生.而 tmp_2 与 out 存在优先关系约束,out 要比 tmp_2 先发生,从而导致死锁.死锁产生的主要原因是 $tmp_1 \triangleq in_1+in_2$ 约束制约太弱,不能保证 in_1 和 in_2 在 out 之前都要发生.而通过下确界与下确界关系约束,则可以

保证这一点,从而避免死锁发生.效率方面,随着界的增大,执行时间增大,然而都可以在较短的时间内完成检测.另外,从表中可以看出,无死锁的求解时间明显较短.

3.4 LTL模型检测

借助 SMT 方法实现 CCSL 的 LTL 模型检测的主要思想是将一个 LTL 公式ψ,根据既定的边界值 n 判断其与由 CCSL 转化成的 SMT 公式的可满足性.若结果是可满足的,则证明ψ是不可满足的,所求的解是ψ不被满足的实例.若验证结果是不满足的,则说明在 n 步之前ψ是被 CCSL 约束满足的.

LTL 公式转换成 SMT 公式分为两种.

- 第 1 种是δ为限界调度,根据 LTL 的语义,对于 LTL 公式ψ,以 $G(\psi_1 \wedge \psi_2)$为例,我们用公式$[\![\psi]\!]_n^1$表示满足ψ的上界为 n 的限界调度δ,通过判断ψ中的连接词来分解公式,得到公式$\forall i \geqslant 1, ([\![\psi_1]\!]_n^1 \wedge [\![\psi_2]\!]_n^1)$.再对子公式$[\![\psi_1]\!]_n^1 \wedge [\![\psi_2]\!]_n^1$进行判断分解直到子公式只有原子公式为止.递归这个过程,最终就得到了该 LTL 公式的 SMT 公式;
- 第 2 种是δ为周期调度,比第 1 种情况较为复杂.其主要思想如下:用公式${}_l[\![\psi_1]\!]_k^1$表示满足 LTL 公式ψ的周期调度$\delta_{l,k}$.因为$\delta_{l,k}$到了 k 步后就会重复 l 步~k 步间的内容,因此逻辑连接词并不受影响.而对于时态连接词则不同,比如 X(next),验证 $X\psi$在 k 步时的满足性就等价于验证ψ在 l+1 步的满足性;又如 G(globally),在无限调度(周期调度)下才有意义,验证 $G\psi$在 l 步前的满足性就等价验证ψ是否在 l 步前处处满足.而在 l 步后,因为周期的存在我们只需验证在一个周期内的满足性即可.

结合第 3.1 节,对于 CCSL 约束集Φ,我们可以通过 LTL 公式对其时间属性进行模型检测.令ψ表示该 LTL 公式,若$[\![\Phi]\!] \wedge \neg [\![\psi]\!]$是不可满足的,则证明该约束集具有$\psi$表示的时间属性.当然,我们通常设置一个边界值 n,用公式$[\![\Phi]\!]_n \wedge \neg [\![\psi]\!]_n^1$来验证限界调度下的满足情况.但在实际系统验证中,周期调度更有实际意义,我们用公式${}_k^l[\![\Phi]\!]_n \wedge \neg {}_l[\![\psi]\!]_k^1$来表示.我们在文献[19]中给出了从 LTL 公式到 SMT 公式的转换过程,并且以一个交通灯控制系统[14]和车载系统中的车窗控制系统为例,验证了该方法的有效性.有兴趣的读者可见文献[19]了解更多细节.

4 工 具

为了使以上对于CCSL形式化验证与分析的方法有更好的可重复性和扩展性,我们将其集成于一个原型工具,其 UML 活动图如图 5 所示.

工具的核心思想是:根据各个形式化方法将给定的 CCSL 约束、LTL 公式转换成对应的 SMT 公式,并放入 SMT 求解器中求解,最终得到结果.它具有以下几点功能.

(1) 在转换前需要输入运行参数:程序超时时间,防止运行时间过长而不能终止的情况;SMT 求解器的选择,可以选择使用 Z3 还是 CVC4 求解 SMT 公式;边界值的设定,当为 0 时表示无界的检测,一般我们会输入个正整数作为检测的界;

(2) 对于输入的 CCSL 约束集合,选择需要进行的操作,如果需要 LTL 模型检测,则要输入 LTL 公式.根据不同操作,工具自动进行转换成对应的 SMT 公式并求解;

(3) 对于 SMT 求解器返回的结果进行处理,不仅提示有没有解,并且对于满足的情况画出结果,整个工具采用图形化界面,具有较强的可操作性;

(4) 采用基于 nodejs 的跨平台框架 electron 进行开发,使得工具便于维护与移植.

图 6 是工具在死锁检测时的页面截图,展示了该原型工具在参数设置、公式输入和结果处理上的交互逻辑.

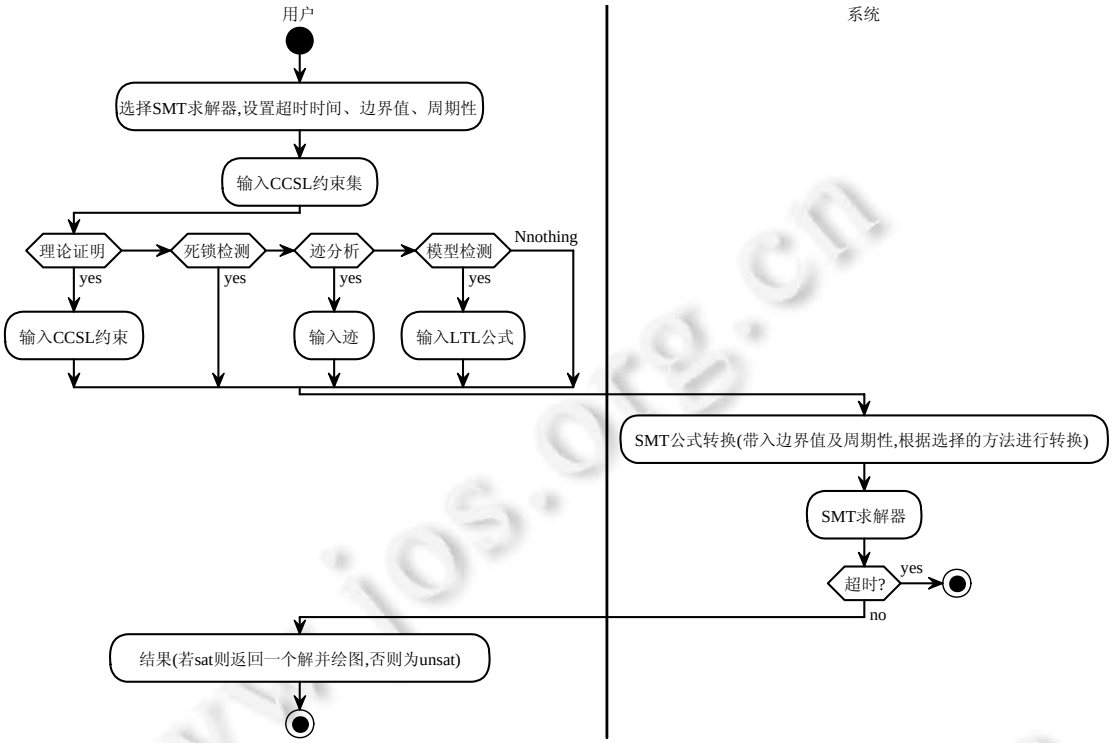

Fig.5 UML activity diagrams of the tool

图 5 该工具的 UML 活动图

Fig.6 Screenshot of the tool on deadlock detection

图 6 死锁检测时的工具截图

5 相关工作

对于CCSL约束的形式化分析方法的相关工作大多基于状态迁移系统与自动机原理.这些方法的优势在于可以利用现有的验证工具对CCSL验证分析.然而受限于目标模型的表达能力,有些工作中只考虑了部分CCSL约束关系.另外,由于对应的验证工具只能实现某种验证与分析,无法对CCSL进行其他方面的分析.例如:Yin等

人提出在 Promela 里对 CCSL 操作符进行转换[7],然后用 Spin 工具进行模型检测;Gaston 等人是将其转换成 Büchi 自动机[8],然后比较 CCSL 与时态逻辑的表现.但这两个方法仅考虑了 CCSL 安全子集.Yu 等人提出用 SINGAL 做 CCSL 的转换[18],但也仅包含一部分,并且 SINGAL 依赖三界逻辑.Suryadevara 等人是将其转换为时间自动机[9],但 UPPAAL 只能验证 CCSL 安全和物理时间的那部分约束.Mallet 等人提出基于状态的 CCSL 语义并将每个约束转换为一个迁移系统[6],然而一些如 Precedence,Supremun 这样的约束并不能用有限状态迁移系统表示.Zhang 等人用 Maude 定义了一套 CCSL 可执行语义[10]进行仿真和模型检测.

本文提出的方法是将 CCSL 约束转换为 SMT 公式并进行统一化求解.本文提出的方法不是单一的针对某个问题而展开,而是系统性地将多个问题统一而有效的解决.虽然该方法在不设置界的情况下满足性问题同样不能完全得到解决,但是基于 SMT 求解器的高效性及其在有界条件下在不同分析中的应用,基于 SMT 的方法依然是对 CCSL 形式化分析的有效方法之一.

已经有工作[16,17]将 LTL 公式转换成 SAT 公式,它们将有限状态自动机模型转换成命题逻辑公式,LTL 公式的转换考虑到两种情况:一种是路径里没有循环,另一种是存在循环,同本文提出的对限界调度和周期调度的验证方法思路基本一致.不同之处有两点:其一是模型不同,本文基于 CCSL 的调度模型;其二是转换成公式为 SMT 公式.

6 总结以及未来工作

本文提出了一种基于 SMT 的统一的 CCSL 形式化分析方法并集成于一个原型工具,介绍了该方法在 CCSL 形式化分析中多种不同的应用,包括有效性证明、迹分析、死锁检测以及 LTL 模型检测.同时展示了他们在具体实例中的应用.实验结果和数据说明了基于 SMT 的方法对 CCSL 形式化分析的有效性及高效性.

在本文工作的基础上有更多的工作值得研究:如何自动确定合适的边界值、如何将描述 CCSL 约束时间属性的 CTL 公式转换成 SMT 公式等.另外,拟将该方法应用于工业案例中更复杂系统的验证,以检验该方法在实际工业案例中的有效性.此外,原型工具也需要进一步完善用于复杂系统的 CCSL 建模与分析,如引入图形化操作实现对 CCSL 约束的定义、对验证结果进行图形化表示使其更容易理解等.

本文中,CCSL 约束有效性证明采用的是对证明的公式取反证明其不可满足的方法.受限于转化的公式的可满足性问题的不可判定性,这种方法是不完全的.*K*-induction 是另一个潜在的方法,是基本数学归纳证明方法的扩展,形式化验证工具 SAL 实现了基于 *K*-induction 方法的定理证明[21].利用 *K*-induction 实现 CCSL 某些定理的证明可避免设定边界值证明的局限性,这又是一个值得深入研究的工作.

References:

[1] Object Management Group. UML profile for MARTE: Modeling and analysis of real-timeembedded systems. 2011.

[2] André C. Syntax and semantics of the clock constraint specification language (CCSL). Research Report, RR-6925, INRIA, 2009.

[3] Mallet F, André C. On the semantics of UML/MARTE clock constraints. In: Proc. of the IEEE Int'l Symp. on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC 2009). IEEE, 2009. 305-312. [doi: 10.1109/ISORC.2009.27]

[4] Kang EY, Schobbens PY. Schedulability analysis support for automotive systems: From requirement to implementation. In: Proc. of the 29th Annual ACM Symp. on Applied Computing. ACM Press, 2014. 1080–1085. [doi: 10.1145/2554850.2554929]

[5] Mallet F. MARTE/CCSL for modeling cyber-physical systems. In: Proc. of the Formal Modeling and Verification of Cyber-Physical Systems. Springer Fachmedien Wiesbaden, 2015. 26–49. [doi: 10.1007/978-3-658-09994-7_2]

[6] Mallet F, De Simone R. Correctness issues on MARTE/CCSL constraints. Science of Computer Programming, 2015,106:78–92. [doi: 10.1016/j.scico.2015.03.001]

[7] Yin L, Mallet F, Liu J. Verification of MARTE/CCSL time requirements in Promela/SPIN. In: Proc. of the 2011 16th IEEE Int'l Conf. on Engineering of Complex Computer Systems (ICECCS). IEEE, 2011. 65–74. [doi: 10.1109/ICECCS.2011.14]

[8] Gascon R, Mallet F, Deantoni J. Logical time and temporal logics: Comparing UML MARTE/CCSL and PSL. In: Proc. of the 2011 18th Int'l Symp. on Temporal Representation and Reasoning (TIME). IEEE, 2011. 141–148. [doi: 10.1109/TIME.2011.10]

[9] Suryadevara J, Seceleanu C, Mallet F, *et al*. Verifying MARTE/CCSL mode behaviors using UPPAAL. In: Proc. of the Int'l Conf. on Software Engineering and Formal Methods. Berlin, Heidelberg: Springer-Verlag, 2013. 1–15. [doi: 10.1007/978-3-642-40561-7_1]

[10] Zhang M, Mallet F. An executable semantics of clock constraint specification language and its applications. In: Proc. of the Int'l Workshop on Formal Techniques for Safety-Critical Systems. Cham: Springer-Verlag, 2015. 37–51. [doi: 10.1007/978-3-319-29510-72]

[11] Zhang M, Mallet F, Zhu H. An SMT-based approach to the formal analysis of MARTE/CCSL. In: Proc. of the Int'l Conf. on Formal Engineering Methods. Springer Int'l Publishing, 2016. 433–449. [doi: 10.1007/978-3-319-47846-3_27]

[12] Barrett C, Stump A, Tinelli C. The smt-lib standard: Version 2.0. In: Proc. of the 8th Int'l Workshop on Satisfiability Modulo Theories. Edinburgh, 2010. 13–14.

[13] De Moura L, Bjørner N. Z3: An efficient SMT solver. In: Proc. of the Tools and Algorithms for the Construction and Analysis of Systems. 2008. 337–340. [doi: 10.1007/978-3-540-78800-3_24]

[14] Cohen B, Venkataramanan S, Kumari A. System Verilog Assertions Handbook—For Formal and Dynamic Verification. Vhdlcohen Publishing, 2005.

[15] Biere A, Cimatti A, Clarke EM, *et al*. Bounded model checking. Advances in Computers, 2003,58:117–148.

[16] Cimatti A, Pistore M, Roveri M, *et al*. Improving the encoding of LTL model checking into SAT. In: Proc. of the Int'l Workshop on Verification, Model Checking, and Abstract Interpretation. Berlin, Heidelberg: Springer-Verlag, 2002. 196–207. [doi: 10.1007/3-540-47813-2_14]

[17] Zhang W. SAT-Based verification of LTL formulas. In: Proc. of the FMICS/PDMC. 2006. 277–292. [doi: 10.1007/978-3-540-70952-7_18]

[18] Yu H, Talpin JP, Besnard L, *et al*. Polychronous controller synthesis from MARTE CCSL timing specifications. In: Proc. of the 9th ACM/IEEE Int'l Conf. on Formal Methods and Models for Codesign. IEEE Computer Society, 2011. 21–30. [doi: 10.1109/MEMCOD.2011.5970507]

[19] Zhang M, Ying Y. Towards SMT-based LTL model checking of clock constraint specification language for real-time and embedded systems. In: Proc. of the 18th ACM SIGPLAN/SIGBED Conf. on Languages, Compilers, and Tools for Embedded Systems. ACM Press, 2017. 61–70. [doi: 10.1145/3078633.3081035]

[20] Zhang M, Dai F, Mallet F. Periodic scheduling for MARTE/CCSL: Theory and practice. In: Proc. of the Science of Computer Programming. 2017. [doi: 10.1016/j.scico.2017.08.015]

[21] De Moura L, Owre S, Rueß H, *et al*. SAL 2. In: Proc. of the Int'l Conf. on Computer Aided Verification. Berlin, Heidelberg: Springer-Verlag, 2004. 496–500. [doi: 10.1007/978-3-540-27813-9_45]

[22] Jin JW, Ma FF, Zhang J. Brief Introduction to SMT solving. Journal of Frontiers of Computer Science and Technology, 2015,9(7):769–780 (in Chinese with English abstract).

附中文参考文献:

[22] 金继伟,马菲菲,张健.SMT 求解技术简述.计算机科学与探索,2015,9(7):769–780.

应云辉(1992－),男,浙江宁海人,硕士,主要研究领域为形式化方法.

张民(1982－),男,博士,副教授,CCF 专业会员,主要研究领域为形式化方法,软件工程.