doi:10.3969/j.issn.1672 - 4348.2016.01.017

实时互斥协议的形式化建模与自动验证

唐郑熠, 陈义, 薛醒思, 杨荣华, 王金水

(福建工程学院 信息科学与工程学院, 福建 福州 350118)

摘要: 实时互斥协议是一类重要且复杂的系统协议,其性质分析工作通常是通过数学方法来进行,不利于使用与推广。针对这一问题,提出基于形式化方法的实时互斥协议验证技术。采用时间自动机对一个典型的实时互斥协议进行建模,并定义了它的语义。同时,分析了该协议所应具有的性质并转化为形式化公式。最后,使用模型检测工具 UPPAAL 对协议性质进行了自动验证。验证结果表明,该协议虽然满足互斥与无死锁两个基本性质,但无法保证进程活性。该方法具有自动化程度高、验证速度快的特点,易于运用与推广。

关键词: 实时; 互斥; 模型检测; 时间自动机

中图分类号: TP393.08　　**文献标志码**: A　　**文章编号**: 1672 - 4348(2016)01 - 0076 - 04

The formal modelling and automatic verification of real-time mutual exclusion protocol

Tang Zhengyi, Chen Yi, Xue Xingsi, Yang Ronghua, Wang Jinshui

(College of Information Science and Engineering, Fujian University of Technology, Fuzhou 350118, China)

Abstract: Real-time mutual exclusion protocol is an important and complex system protocol. The nature analysis of the protocol is commonly made by mathematic methods, which is difficult to use and hard to be introduced. A real-time mutual exclusion verification technique based on formal technique was proposed. The modelling of a typical real-time exclusion protocol was conducted via a time automata, the semantics of which was defined. The nature of the protocol was analysed, which was transformed into formal formula. The automatic verification of the nature of the protocol was performed via model testing tool UPPAAL. The protocol is mutually exclusive and deadlock free, but it cannot ensure the activity of processing. The results indicate that the technique is highly automatic with a high speed of verification and is applicable.

Keywords: real-time; mutual exclusion; model testing; time automata

互斥问题是由图灵奖得主 Dijkstra 提出的有关进程互斥访问临界区的重要问题[1],指的是多个并发进程由于竞争某些共享资源而产生的相互制约关系。进程互斥的目的,是为了保证多个相互冲突的并发进程能够独占性访问共享的系统资源[2],即不允许出现两个及以上的并发进程同时访问共享资源(临界区)的情况。解决互斥问题的常见方法包括基于调度的方法、基于令牌的方法、以及基于消息交互的方法。但这些解决方法的空间复杂度都达到了 $O(n)$[3],在并发进程数量较多的情况下,会消耗较多的寄存器资源,严重影响系统性能。为了降低算法的空间复杂度,研究人员通过时间约束来达成互斥性。然而随之而来的,是实时互斥协议的正确性及可靠性的分析

收稿日期: 2015 - 12 - 15

基金项目: 福建省科技厅基金资助项目(JK2012033);福建省中青年教师教育科研项目(JA15336,JB14069,JB12146);福建工程学院科研启动基金项目(GY - Z13112,GY - Z13113,GY - Z15007)

第一作者简介: 唐郑熠(1984 -),男,福建福州人,讲师,博士,研究方向:形式化方法。

问题。时间约束本身就具有较高的抽象性，而作用于多个并发进程上的多个时间约束，相互之间又会产生难以预计的影响，从而大大提高了算法分析与验证的复杂性。因此，实时算法的分析主要是通过数学方法来进行。

本文将研究另一种实时算法的分析技术，即基于模型检测的验证技术。模型检测是一类验证系统性质的算法与技术，其基本原理是，将给定的系统规约为模型 M，期望的系统性质归约为逻辑公式 φ，通过状态空间搜索的方法来验证 φ 在 M 中是否成立[4]。模型检测方法的技术门槛较低、自动化程度高，更易于使用与推广，是公认的、最具工业应用前景的形式化分析技术。

本文对一个重要的实时互斥协议——Fischer 协议进行了建模与验证，采用时间自动机构建其形式化模型，分析了该协议的重要性质并转化为形式化公式，并通过 UPPAAL 模型检测技术对协议性质进行了验证。与已有的同类工作相比[5]，本文的验证结果同样证明了 Fischer 协议能够保证进程互斥且不会产生死锁，但更进一步地验证出其无法保证进程活性，而这一问题是现有构建的模型所无法验证的。由此证明，本文所构建的形式化模型更为精确地描述了 Fisher 协议，可作为进一步协议分析与验证工作的基础。

1　Fischer 协议

Fischer 协议[6]是由图灵奖得主 Leslie Lamport 与美国科学家 Michael Fischer 共同提出的一个实时互斥算法，它为并发进程访问临界区的行为增加了时间约束，从而达成了互斥性。Fischer 协议为每个进程赋予一个编号（从 1 开始），并使用一个共享的变量 id（初值为 0）控制进入临界区的进程。一个编号为 i 的并发进程访问临界区的行为过程，可以用伪代码描述如下：

```
while(true) {
    Noncritical Section;
L: if(id≠0) then goto L;
    id := i;
    delay(k);
    if(id≠i) then goto L;
    Critical section;
    id := 0;
}
```

任意进程在释放临界区后，都会将标识变量 id 的值置为 0，故若 $id \neq 0$ 时，说明临界区没有被占用。因此当进程 i 在准备访问临界区前，应先判定 id 的值是否为 0：若 $id \neq 0$，则继续等待；若 $id \neq 0$，则将 id 的值赋为自己的编号 i。由于进程从完成 id 值的判定到完成对 id 的赋值，并不是即时的，在这段延迟内，可能有其它进程插入，如图 1 所示。“read”表示读取 id 的值，“write(x)”表示将 id 赋值为 x，数字表示操作的执行顺序。

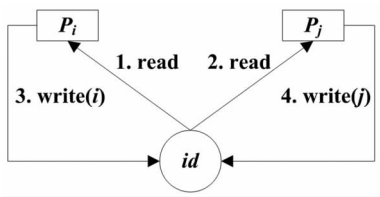

图 1　赋值延迟现象

Fig. 1　Value assigning delay

显然，若 id 的初值为 0，则最终其值为 j。此时，若两个进程同时进入临界区，则会违背互斥性。因此，Fischer 协议要求进程 i 在完成对 id 的赋值后，延迟 k 个时间单位，再对 id 进行一次判定：若 $id \neq i$，则进入临界区；若 $id \neq i$，则进入新一轮的等待。

Lamport 使用逻辑推理方法[6]，证明了 Fischer 协议具有互斥性，且无死锁。这两个性质也成为了互斥算法的设计标准。

2　基于时间自动机的形式化建模

由于 Fischer 协议是一个实时协议，因此需要采用能够支持时间属性的形式化语言进行建模，同时该语言还要能够转化为可搜索的状态空间。依据这两个要求，本文选取时间自动机作为建模语言。

2.1　时间自动机

时间自动机是一种扩展的 Büchi 自动机[7]，采用时钟变量约束自动机的状态与迁移，自动机只能停留在满足时间约束的状态上，且只能执行满足时间约束的迁移。下面给出它的语法与语义。

定义 1　时钟　一个时钟 c 是一个变量，$c \in R^+$。

定义 2　时钟解释　一个时钟解释是一个映射 $u: C \rightarrow R^+$，为每个时钟赋予一个非负实数。U 为全体时钟解释，u^0 表示将全部时钟解释为 0，有

以下相关记号：

· $u+t$：$\forall c \in C: u(c+t) = u(c) + t, t \in R^+$。

· $u \models \varphi$：时钟解释 u 满足时间约束 φ。

· $[Y:=0]u(Y \subseteq C)$：将 Y 中的时钟解释为0，其余时钟保持 u 中的解释不变。

定义3 时间约束 设 C 是时钟的有穷集合，C 上的时间约束集 $B(C) = \{\varphi \mid \varphi = (c \sim n) \text{or} (c_1 \wedge c_2) \text{or} (\text{true}), c \in C, \sim \in \{<, \leqslant, =, \neq, \geqslant, >\}, n \in N^+\}$。

定义4 时间自动机 时间自动机为一个七元组 $TA = (L, sl, C, B(C), f, \Sigma, E)$，其中：

· L 是位置的有穷状态集合，$sl \in L$ 为初始状态。

· C 是时钟集合，$B(C)$ 是时间约束集合。

· $f: L \to B(C)$ 是时间约束函数，为每个状态指定一个时间约束。

· $\sum$ 是迁移标号集合。

· $E \subseteq L \times B(C) \times \Sigma \times 2^C \times L$ 为迁移集合，$e = (l, \varphi, a, Y, l') \in E$。

定义5 实时状态 实时状态集合 $S \subseteq L \times U, s = (l, u) \in S$。

定义6 时间自动机的语义 时间自动机的语义是一个基于实时状态的标号迁移系统 $(S, ss, \to)$，$ss = (sl, u^0)$ 为初始实时状态，关系 $\to \subseteq S \times (R^+ \cup \sum) \times S$，包括两种类型：

· $(l, u) \to^d (l, u+d)$，若 $\forall d': 0 \leqslant d' \leqslant d \Rightarrow u + d' \models f(l)$。

· $(l, u) \to^a (l', u')$，若 $\exists e = (l, \varphi, a, Y, l') \in E$，且 $u \models \varphi, u' = [Y:=0]u, u' \models f(l')$。

2.2 Fischer 协议的形式化模型

Fischer 协议描述的是进程的行为过程，而多个进程之间是相互独立的，每个进程可以用一台时间自动机描述，称为并发进程自动机，但需要对迁移标号进行扩展：将迁移标号扩展为二元组 $<g, op>$：g 是一个布尔表达式，称为卫式；op 是一组赋值表达式，称为操作。

定义7 并发进程自动机 一个并发进程自动机 $CPA_i = (L_i, sl_i, C_i, B(C)_i, f_i, \Sigma_i, E_i)$，其中：

· $L_i = \{\text{Wait}, \text{Req}, \text{Delay}, \text{CS}\}$.

· $sl_i = \text{Wait}$.

· $C_i = \{c\}$.

· $B(C)_i = \{c \leqslant n, c < k+1, c > k\}$.

· $f_i = \{(c \leqslant n, \text{Req}), (c < k+1, \text{Delay})\}$.

· $\Sigma_i = \{a_1 = (id = 0, \varnothing), a_2 = (\varnothing, \{id := i\}), a_3 = (id = i, \varnothing), a_4 = (id \neq i, \varnothing), a_5 = (\varnothing, \{id := 0\})\}$.

· $E_i = \{(\text{Wait}, \varnothing, a_1, \{c\}, \text{Req}), (\text{Req}, \varnothing, a_2, \{c\}, \text{Delay}), (\text{Delay}, c > k, a_3, \varnothing, CS), (\text{Delay}, \varnothing, a_4, \varnothing, \text{Wait}), (CS, \varnothing, a_5, \varnothing, \text{Wait})\}$.

定义7中的 id 是所有进程共享的标识变量，i 是进程的编号。

用图形表示法，并发进程自动机可以表示为如图2所示。出于简洁的目的，图2中将卫式与迁移的时间约束合并为一个部分，称为迁移约束；操作与时钟重置合并为一个部分，称为迁移效果。

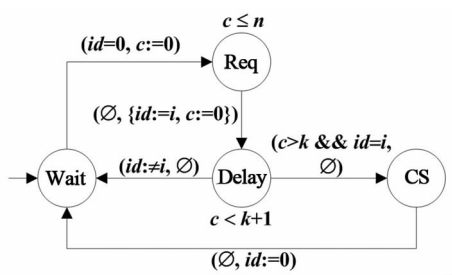

图2 并发进程自动机

Fig. 2 Simultaneous forward automata

并发进程在判定临界区未被其他进程访问的情况下（$id = 0$），进入 Req 状态，准备对 id 进行赋值。由于赋值不是即时的，因此允许其在 Req 状态停留至多 n 个时间单位。在进入 Delay 状态后，进程需延时 k 个时间单位，才能尝试进入临界区（CS），因此从 Delay 到 CS 的迁移上添加了时间约束 $c > k$。且为了保证其在延时后会离开 Delay 状态，在该状态上增加了时间约束 $c < k+1$。

多个并发进程构成一个网络，称为进程网络。进程网络还需要一个共享的标识变量 id，用于控制进程的访问行为。

定义8 进程网络 一个进程网络 $PN = (\{CPA_i \mid 1 \leqslant i \leqslant n\}, id)$，包含 n 台并发进程自动机 PTA_i，以及一个公共变量 $id (0 \leqslant id \leqslant n)$。$PN$ 的状态是由所有的 CPA 及 id 共同决定的，用状态向量表示 $\boldsymbol{l}^V = <l_1, l_2, \cdots, l_n, id>; l_i \in L_i$；初始状态向量 $\boldsymbol{sl}^V = <sl_0, sl_1, \cdots, sl_{n-1}, 0>, sl_i \in PTA_i$；状态向量的时间约束函数 $f^V(sv) = f_0(l_0) \wedge f_1(l_1) \cdots \wedge f_{n-1}(l_{n-1}), l_i \in l^V$.

为了定义进程网络的语义，首先给出实时状

态向量的定义：

定义 9　实时状态向量　实时状态向量集合 $Q \subseteq L^V \times U \times \{id\}$，$q = (\boldsymbol{l}^V, u, id) \in Q$.

定义 10　进程网络的语义　包含 n 台 *CPA* 的进程网络的语义，是一个基于实时状态向量的标记迁移系统 $(Q, \boldsymbol{sq}, \rightarrow)$，$\boldsymbol{sq} = (\boldsymbol{sl}^V, u^0, 0)$ 是初始实时状态向量，迁移关系 $\rightarrow \subseteq Q \times (R^+ \cup \sum) \times Q$，分为时间迁移与状态迁移：

· 时间迁移：$(\boldsymbol{l}^V, u, id) \rightarrow^d (\boldsymbol{l}^V, u + d, id')$，若 $\forall d' : 0 \leqslant d' \leqslant d \Rightarrow u + d' \mid = f^V(\boldsymbol{l}^V)$.

· 状态迁移：$(\boldsymbol{l}^V, u, id) \rightarrow^a (\boldsymbol{l}^V[l'_j/l_j], u', id')$，若 $\exists (l_j, \varphi, a, Y, l'_j) \in E_i$，且 $u \mid = \varphi$，$u' = [Y := 0]u$，$u' \mid = f^V(\boldsymbol{l}^V[l'_j/l_j])$.

3　Fischer 协议的性质验证

为了验证 Fischer 协议的正确性，应首先分析其所应具有的性质，并转化为形式化公式，然后使用相应的模型检测工具进行性质验证。

3.1　Fischer 协议的性质及其形式化

作为互斥协议，Fischer 协议首先要保证的就是互斥性，即在任何时刻，处于 CS 状态的进程，至多只有一个。可用形式化公式表示为：

$$\forall q \in Q: \text{若 } \exists l_i, l_j \in q.\boldsymbol{l}^V: \quad l_i = \text{CS} \wedge l_j = \text{CS} \Rightarrow i = j \qquad (1)$$

Fischer 协议所应满足的第二个性质是无死锁，即在任何时刻，都至少存在一个进程，能够发生状态迁移。可用形式化公式表示为：

$$\forall q \in Q: \text{若 } \exists (\boldsymbol{l}^V, u, id) \rightarrow^a (\boldsymbol{l}^V[l'_j/l_j], u', id') \qquad (2)$$

因为如果所有进程都进入死锁状态，则失去了访问临界区的可能，此时即使满足了互斥性，也是无意义的。

除了以上两个基本性质，互斥协议还应保证进程具有活性，即任意进程一旦开始申请进入临界区（Req 状态），则最终一定能够进入临界区（CS 状态）。可用形式化公式表示为：

$$\forall q \in Q: \text{若 } \exists q.\boldsymbol{l}^V.l_i = \text{Req} \Rightarrow \exists q' \in Q: \quad q.\boldsymbol{l}^V.l_i = \text{CS} \wedge q[\rightarrow]_{tr} q' \qquad (3)$$

其中，$[\rightarrow]_{tr}$ 是进程网络语义中迁移关系的传递闭包。

活性是为了防止出现进程饥饿现象，即某个进程申请了临界区资源，但却始终无法进入临界区。

3.2　基于 UPPAAL 的性质验证

UPPAAL[8] 是瑞典 Uppsala University 与丹麦 Aalborg University 的科学家与技术人员共同研发的一个实时系统模型检测工具，它以扩展的时间自动机作为建模语言、以简化的 TCTL 逻辑作为性质描述语言[9]，适合于描述能够被划分为多个并行子结构的实时系统。UPPAAL 采用了先进的状态空间约减技术，具有出众的搜索效率，同时其建模语言能够完整地描述所构建的并发进程自动机。

将 3.1 节中分析的三个性质用 UPPAAL 的建模语言分别表示如下：

性质①　A[ ] forall (i: id_t) forall (j: id_t) P(i).cs && P(j).cs imply i == j

性质②　A[ ] not deadlock

性质③　P(1).req - - >P(1).cs

其中，P 是进程名称，后面括号中的变量与数字表示进程编号；id_t 是自定义数据类型，是范围 [1, 6] 的整型数据，因为实验中设置了 6 个进程；deadlock 是 UPPAAL 支持的关键字，表示系统模型出现了死锁。

使用 UPPAAL 模型检测器对上述 3 个性质进行验证，验证结果表明系统模型性质①与性质②，但不满足性质③。通过分析 UPPAAL 给出的轨迹文件，发现了违反性质③的系统运行路径，如图 3 所示。由于该执行路径只涉及 P(1)、P(5)、P(6) 进程，因此用 (S_1, S_2, S_3) 的形式表示系统状态向量，其中 S_1 表示进程 P(1) 的状态、S_2 表示进程 P(5) 的状态、S_3 表示进程 P(6) 的状态。

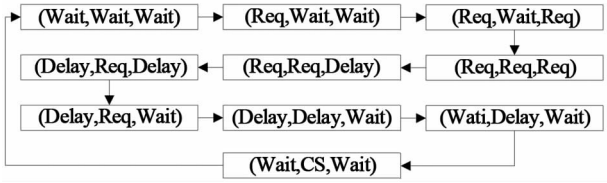

图 3　违反进程活性的系统运行路径

Fig. 3　System operation route against processing activity

该路径违反性质③的原因在于：当 P(1) 进入 Delay 状态后，P(5) 随后也进入了 Delay 状态，从而导致 P(1) 在延时结束后只能返回 Wait 状态；而 P(5) 在从 CS 状态返回到 Wait 状态后，P(1)

（下转第 85 页）

参考文献：

[1] 孙吉贵,刘杰,赵连宇. 聚类算法研究[J]. 软件学报,2008,19(1):48 - 61.

[2] 余正涛,樊孝忠,郭剑毅,等. 基于潜在语义分析的汉语问答系统答案提取[J]. 计算机学报,2006,29(10):1889 - 1893.

[3] 吴飞,韩亚洪,庄越挺,等. 图像 - 文本相关性挖掘的 Web 图像聚类方法[J]. 软件学报,2010,21(7):1561 - 1575.

[4] 吴夙慧,成颖,郑彦宁. K-means 算法研究综述[J]. 现代图书情报技术,2011(5):28 - 35.

[5] 翟东海,鱼江,高飞,等. 最大距离法选取初始簇中心的 K-means 文本聚类算法的研究[J]. 计算机应用研究,2014,31(3):713 - 719.

[6] Jain A K. Data clustering: 50 years beyond k-Means[J]. Pattern Recognition Letters,2010,31(8):651 - 666.

[7] Song Q B, Ni J J, Wang G T. A fast clustering-based feature subset selection algorithm for high-dimensional data[J]. IEEE Trans on Knowledge and Data Engineering,2013,25(1):1 - 14.

[8] Aldahdooh R T, Ashour W. Distance-based initialization method for K-means clustering algorithm[J]. International Journal of Intelligent Systems and Applications,2013,5(2):41 - 51.

[9] 李法运,农罗锋. 基于向量语义相似度的改进 K-Means 算法[J]. 情报科学,2013,31(2),34 - 37.

[10] 吴夙慧,成颖,郑彦宁,等. K-means 算法研究综述[J]. 现代图书情报技术, 2011,34(5):28 - 37.

(责任编辑：肖锡湘)

(上接第 79 页)

可以仍然处于 Wait 状态,从而形成循环。

由此可知,Fischer 协议无法避免进程饥饿现象,对于单个申请访问临界区的进程,不能保证其一定能够最终获取临界区资源。

4　结语

形式化方法是用于保障系统安全性与可靠性的重要手段,也是当前系统分析研究领域的前沿与热点,并在多类不同系统的分析中得到了成功运用。实时互斥协议作为一种时间敏感的系统协议,采用具有严格数学语义的形式方法进行性质分析,是十分有效且必要的。使用时间自动机构建了 Fischer 协议的形式化模型,并通过模型检测技术验证其满足互斥与无死锁两个重要性质,但同时也验证出其不满足进程活性。由此表明,形式化技术尤其是模型检测技术是一种有效的协议分析技术,能够成为保障各类复杂协议正确性及可靠性的有力手段。

参考文献：

[1] Dijkstra E W. Solution of a problem in concurrent programming control[J]. Communications of the ACM,1965,8(1):289 - 294.

[2] Hesselink W H. Mutual exclusion by four shared bits with not more than quadratic complexity[J]. Science of Computer Programming,2015,102(5):57 - 75.

[3] Lynch N, Shavit N. Timing-based mutual exclusion[C]//Proc of IEEE Real-Time Systems Symposium. Phoenix, America: IEEE Computer Society,2002:2 - 11.

[4] Machin M, Dufosse F, Blanquart J P, et al. Specifying safety monitors for autonomous systems using model-checking[J]. Lecture Notes in Computer Science,2014, 8666:262 - 277.

[5] Behrmann G, David A, Larsen K G. Formal methods for the design of real-time systems[M]. Berlin: Springer-Verlag, 2004:200 - 236.

[6] Lamport L. A fast mutual exclusion algorithm[J]. Acm Transactions on Computer Systems Tocs Homepage,1987,5(1):1 - 11.

[7] Beatrice B. An introduction to timed automata[J]. Lecture Notes in Control and Information Science,2013,433:169 - 187.

[8] Behrmann G, David A, Larsen K G, et al. Developing UPPAAL over 15 years[J]. Software: Practice and Experience, 2011,41(2):133 - 142.

[9] Behrmann G, David A, Larsen K G, et al. UPPAAL 4.0[J]. Quantitative Evaluation of Systems,2006,4(12):125 - 126.

(责任编辑：肖锡湘)