基于Petri网的CSP并发系统验证技术研究

刘彦青 赵岭忠 钱俊彦
(桂林电子科技大学计算机科学与工程学院 桂林 541004)

摘 要 通信顺序进程(CSP)和Petri网是两种重要的并发系统建模工具。CSP语言具有高度抽象性,可有效刻画并发进程之间的各种相互作用,但在物理结构的描述与验证分析方面显得不足。Petri网是一种形式化、图形化的并发系统建模和分析工具,侧重于系统的物理结构描述和性质分析。结合两者优点,首先利用CSP描述待验证的并发系统,然后将其转化为Petri网来分析系统的动态行为特性,最后利用性质分析工具TINA对系统性质进行分析和验证。实验结果表明,传统的CSP进程性质验证工具不能验证CSP进程的安全性,但其转化为Petri网后可有效地分析出导致安全性不能满足的危险因素,从而扩大了CSP描述的并发系统可验证性质的范围。

关键词 通信顺序进程(CSP),并发系统,Petri网,性质验证,安全性
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.10.050

Verification of Concurrent CSP Systems Based on Petri Net

LIU Yan-qing ZHAO Ling-zhong QIAN Jun-yan
(School of Computer Science and Engineering, Guilin University of Electronic Technology, Guilin 541004, China)

Abstract Communicating sequential processes (CSP) and Petri net are two important formal methods for analyzing concurrent systems. The CSP language features high level abstraction which is useful to effectively describe the interactions between concurrent processes, but it has weaknesses in the description and analysis of the physical structure of systems. Petri net is a tool for modeling and analyzing concurrent systems in a formal and graphical form, focusing on the physical description of system structure and properties analysis. This paper combined the advantages of both CSP and Petri net. At first, we used CSP to describe concurrent systems and then translated them into Petri net to analyze the dynamic behavior of the system. Finally, we used the model checking tool TINA to analyze and verify the system properties. It is shown that the safety property of CSP processes cannot be verified with existing properties analysis tools like PAT, however, it can be analyzed when the CSP description is transformed into Petri net. This transformation effectively expands the scope of verifiable properties of concurrent systems described by CSP.

Keywords Communicating sequential processes (CSP), Concurrent systems, Petri net, Property verification, Safety

1 引言

随着信息技术的高速发展,并发系统得到广泛应用。并发系统的描述和验证也随之成为学术界研究的一个重要方向。形式化方法不仅可以对系统进行精确的描述和有效的验证,还可通过定性、定量分析保证系统的正确性和可靠性,因而成为并发系统验证研究中的热点。

通信顺序进程(CSP)[1]是一种高度抽象的形式化描述语言,具有严格的数学理论支撑,可有效刻画并发系统进程之间的各种相互作用。然而,在CSP并发系统的验证方面,由于其本身特性所限,还存在以下不足[3]:1)用CSP语言描述的并发系统没有反映系统物理结构信息,不便于与此相关的系统特性的分析和验证;2)尽管对并发有一定的刻画,但不能全面反映复杂系统的真实并发行为。Petri网[2-4]是一种形式化、图形化的并发系统建模和分析工具,虽然在原始模型抽象方面不如CSP,但能够精确地分析系统的静态特性和动态行为性质。为此,本文结合两者优点,首先,利用CSP的高度抽象特点来描述待验证的并发系统,从而得到一个包含较少状态的抽象CSP模型,然后将该模型转化为Petri网来分析系统的动态行为特性。许多学者在该方向做了一定的研究,归纳而言,在建模方面,大致有两条途径:1)从CSP语法方面考虑,以文献[5,6]为代表。文献[5]将CSP的基本语法用Petri网表示;文献[6]提出了类CSP语言(CSP-like language),并将其转化为拥有独立环的Pr/T网。二者均未考虑嵌套和并行命令,同时在该技术的扩展及应用方面没有进行更深入的研究。2)从CSP语义方面考虑,以文献[7-11]为代表。文献

到稿日期:2014-10-17 返修日期:2015-02-03 本文受国家自然科学基金(61262008,61100186),广西自然科学基金(2013GXNSFBA019267),广西教育厅重点项目:高可信软件的安全性验证研究,广西高等学校高水平创新团队及卓越学者计划,广西可信软件重点实验室基金项目(kx201113),桂林电子科技大学创新团队资助。

刘彦青(1988-),女,硕士生,主要研究方向为并发系统验证;赵岭忠(1977-),男,博士,教授,主要研究方向为形式化技术,E-mail:zhaolingzhong163@163.com(通信作者);钱俊彦(1973-),男,博士,教授,主要研究方向为模型检验。

[7-9]利用CSP进程的迹语义来描述系统行为，然后将每个进程对应的迹转化为Petri网，但没有给出非确定性选择进程的转化技术，也没有对建立的模型进行安全性分析。该工作的较新研究成果来自文献[10,11]，其利用CSP进程的操作语义来描述系统行为，并运用给定的语义直接转换为Petri网，但转化算法较为复杂，产生的Petri网冗余信息较多，不利于后续的模型检测。此外，文献[12]是从CCS异步通信的角度利用开放Petri网分析相关性质，文献[13]是运用标记Petri网编码CSP的同步通信进程。在性质分析验证方面，文献[14]用Petri网对CSP进行性能分析，使用带有非负实数的向量来描述命令执行频率之间的关系，给出了一类CSP进程的时间复杂度。该研究没有涉及利用Petri网对CSP进程的活性特别是安全性进行验证的问题。文献[15]采用Petri网对CSP描述进行可靠性分析，用基于CSP的原语来描述待分析系统，对转换后的系统进行参数敏感性分析等，分析过程较繁杂。

本文利用CSP进程的迹语义来描述系统行为，全面考虑了CSP的确定性和非确定性选择进程，详细给出将进程迹转化为Petri网的过程。在性质验证方面，利用Petri网分析工具TINA(Time Petri Net Analyzer)[16-19]分析转化后模型的活性和安全性。实验结果表明，传统的CSP进程性质分析工具不能验证CSP进程的安全性，但转化为Petri网后可有效地分析出导致安全性不能满足的危险因素，从而扩大了CSP描述的并发系统可验证性质的范围。此外，转化后的Petri网还在一定程度上减少了CSP并发操作中的冗余信息。

本文第2节介绍CSP和Petri网的基础知识；第3节讨论如何将CSP并发描述转化为Petri网模型；第4节给出本文方法的实验结果；最后总结本文工作并指出未来研究方向。

2 基础知识

2.1 CSP

CSP[1]理论中，进程是描述客体的基本单位，进程间通过通信事件进行交互。进程的基本结构包括：前缀、递归、一般选择、非确定选择、并发等。进程的迹是进程到某个时刻为止的行为的顺序记录。

定义1[1] 设x是一个事件，P是一个进程，αP表示进程P的字母表，$traces(P)$表示进程P的迹。若$x\in\alpha P$，则$P=x\rightarrow STOP$表示一个进程，$traces(P)=\{\langle\rangle\}\cup\{\langle x\rangle\}$表示进程$P$的迹。

CSP存在一系列指称语义模型，如迹模型(trace model)、失败/发散集模型(failure-divergence model)、稳定失败模型(stable failure model)，它们分别提供了进程的不同行为信息。本文主要用到CSP的一种指称语义模型——迹模型。CSP进程的迹模型包含一系列迹，其中每条迹表示进程的一条行为序列，该序列由进程可执行的事件组成，并以事件发生的先后顺序进行排列。该模型显式地提供了进程的所有可能行为序列。下面给出基本结构进程的迹模型计算方式。

若进程P为$STOP$进程，则P的迹模型为：

$traces(STOP)=\{t|t=\langle\rangle\}$

若进程P为：$P=a\rightarrow STOP$，则P的迹模型为：

$traces(P)=\{\langle\rangle\}\cup\{\langle a\rangle\}$

若进程P为：$P=a\rightarrow Q$(其中，进程$Q=a\rightarrow\cdots\rightarrow STOP$)，则$P$的迹模型为：

$traces(P)=\{\langle\rangle\}\cup\{\langle a\rangle\frown t|t\in traces(Q)\}$(“$\frown$”是迹之间的连接算子)

若进程P为：$P=a\rightarrow\cdots\rightarrow b\rightarrow P$，设进程$Q=a\rightarrow\cdots\rightarrow b\rightarrow STOP$，进程$Q$的最长迹为$q$，$q$的所有子迹模型为$sub\text{-}t$，则$P$的迹模型为：

$traces(P)=\{\langle\rangle\}\cup\{(q\frown\langle b,a\rangle)^n\}\cup\{t|t=r\frown s,r\in(q\cap\langle b,a\rangle)^n,s\in sub\text{-}t,n=0,1,2\cdots\}$

若进程P为：$Q\ \square\ T$，Q和T为前缀进程或递归进程，则进程P的迹模型为：

$traces(P)=\{q|q\in traces(Q)\}\cup\{t|t\in traces(Q)\}$

若进程P为：$Q\amalg T$，Q和T为前缀进程或递归进程，则进程P的迹模型为：

$traces(P)=\{q|q\in traces(Q)\}\cup\{t|t\in traces(Q)\}$

若进程P为：$Q\parallel T$，Q和T为前缀进程或递归进程，则进程P的迹模型为：

$traces(P\parallel Q)=\{t|(t\uparrow\alpha P)\in traces(P)\wedge(t\uparrow\alpha(Q)\in traces(Q)\wedge t\in(\alpha P\cup\alpha Q)^*\}$

其中，$t\uparrow\alpha P$表示t中属于αP的事件组成的新迹，$(\alpha P\cup\alpha Q)^*$表示字母$\alpha P\cup\alpha Q$任意排列组成的迹集合。

进程的并发组合是构造并发系统的核心，须满足以下条件：1)交互双方进程有部分字母表相同；2)进程间的共同事件为通信事件，需共同参与，其余事件则不需共同参与。假设有事件a、b、c、d，进程P、Q，且$a\in(\alpha P-\alpha Q)$，$b\in(\alpha Q-\alpha P)$，c、$d\in(\alpha Q\cap\alpha P)$，可以得到并发规则$CR_{csp}$。

$CR_{csp\text{-}1}:(c\rightarrow P)\parallel(c\rightarrow Q)=(c\rightarrow(P\parallel Q))$

$CR_{csp\text{-}2}:(c\rightarrow P)\parallel(d\rightarrow Q)=STOP$，若$c\neq d$

$CR_{csp\text{-}3}:(a\rightarrow P)\parallel(c\rightarrow Q)=(a\rightarrow(P\parallel(c\rightarrow Q))$

$CR_{csp\text{-}4}:(c\rightarrow P)\parallel(b\rightarrow Q)=(b\rightarrow((c\rightarrow P)\parallel Q))$

$CR_{csp\text{-}5}:(a\rightarrow P)\parallel(b\rightarrow Q)=(a\rightarrow(P\parallel(b\rightarrow Q))\parallel b\rightarrow((a\rightarrow P)\parallel Q))$

2.2 Petri网与时间Petri网

任何系统都可抽象为状态(或条件)、活动(或事件)及二者之间关系的三元结构。在Petri网中，状态用位置(place)表示，活动用迁移(transition)表示。迁移的作用是改变状态，位置的作用是决定迁移能否发生，迁移和位置之间的这种依赖关系用流来表示。Petri的具体定义如下[2-4]：

定义2 Petri网结构是一个三元组$N=(P,T;F)$，其中：

1)$P=\{p_1,p_2,\cdots,p_n\}$是有限位置集合；

2)$T=\{t_1,t_2,\cdots,t_n\}$是有限迁移集合，其中，$P\cup T\neq\emptyset$，$P\cap T=\emptyset$；

3)$F\subseteq(P\times T)\cup(T\times P)$为流关系。

位置集和迁移集是Petri网的基本成分，流关系是从它们构造出来的。图形表示中，用圆圈表示位置，用方框表示迁移，用有向弧表示流关系。

定义3 一个四元组$PN=(P,T;F,M_0)$称作普通Petri网，其中：

1)$N=(P,T;F)$是一个Petri网结构。

2)$M:P\rightarrow Z$(非负整数集合)是位置集合上的标识(marking)向量。对于任意位置$p\in P$，$M(p)$表示标识向量M中位置p所对应的分量，称为位置p上的标识或者令牌数目。

M_0 是初始标识向量。

定义 4 一个五元组 $Z=(P,T;F,M_0,I)$称作时间 Petri 网当且仅当：

1)$\Sigma=(P,T;F,M_0)$是一个 Petri 网。

2)$I:T\to Q^+\times(Q^+\cup\{\infty\})$，其中，$Q^+$表示非负有理数，$\forall t\in T, I(t)=[eft(t),lft(t)]\wedge eft(t)\leqslant lft(t)$。其中，$I$ 称作 Petri 网的时间函数，$eft(t)$和 $lft(t)$分别称作 t 的最早发生时间(earliest firing time)和最晚发生时间(latest firing time)。

在含有令牌的 Petri 网中，依据迁移的使能(enable)条件，可以使得使能的迁移引发(fire)，迁移的引发会依据引发规则实现令牌的移动，不断变化着的令牌重新分布就描述了系统的动态行为。

迁移的使能条件：对于 Petri 网 $PN=(P,T;F,M)$，如果 $\forall p_i, p_i\in {}^*t\to M(p_i)\geqslant 1$，则称 t 在 M 下使能，记作$M[t\rangle$，其中 $p_i\in P$，${}^*t$ 是 p_i 的输入集。

迁移的引发规则：对于 Petri 网 $PN=(P,T;F,M)$，任何在 M 下使能的迁移 t 将会引发，迁移 t 的引发使得位置中令牌重新分布，从而将标识 M 变成新标识 M'，此时称 M' 为 M 的后继标识，记作 $M[t\rangle M'$。

2.3 Petri 网的可达树

定义 5 设 $PN=(P,T;F,M_0)$是一个普通 Petri 网，M 和 M'是两个标识向量，如果存在 $t\in T$，使得 $M[t\rangle M'$，则称M'是从 M 直接可达的。如果存在变迁序列 $t_1,t_2,\cdots,t_k$ 和标识序列 $M_1,M_2,\cdots,M_k$ 使得：

$$M[t_1\rangle M_1[t_2\rangle M_2\cdots M_{k-1}[t_k\rangle M_k$$

则称 M_k 是从 M 可达的。从 M 可达的一切标识的集合记为 $R(M)$。

定义 6 设 $PN=(P,T;F,M_0)$是一个普通 Petri 网，树 $T=(V,E)$称作 Petri 网 PN 的可达树，如果 $V=R(M)$，$E\subseteq T$，且对于 $\forall t\in E$，$\exists M,M'\in V, M[t\rangle M'$。

对于有界 Petri 网，可达树中包含了其所有可达标识，因而所有的性质都可由可达树来分析。为此，下文中所涉及的 Petri 网均限定为有界 Petri 网。

3 CSP 并发描述到 Petri 网的转换规则

3.1 转换规则

为了方便制定相关进程的转化规则，本文约定如下：

1)不允许递归进程的合取组合。原因是该组合可能定义一个无限状态机[1]，从而无法确保转化后的 Petri 网结构是一个有限状态空间。

2)参与不同事件的进程描述是连续和有限的。目的是确保系统的行为均可由对应 CSP 进程的迹来描述，且所描述的状态空间是有限的。

在下文的转换规则中，将相关进程的迹转化为对应的 Petri 网。其中 Petri 网中的状态用位置(即圆圈)表示，事件用迁移(即方框)表示，有向弧表示流关系，黑点表示令牌数。将给定的 CSP 描述转化为 Petri 网，得到的结果是不唯一的，因为为了结合子网，必须引进空位置和空迁移来保证 Petri 网的完整性。

规则 1 前缀进程

定义 7(前缀) 设 x 是一个事件，Q 是一个进程，且 $x\in\alpha Q$，$\alpha Q=\alpha(x\to P)$，则 $Q=x\to P$ 表示进程 Q 先执行事件 x，然后按照 P 的行为进行动作。其中 x 称为前缀，$x\to P$ 称为前缀表达式。

前缀表示行为动作有限的顺序进程，所以 $Q=x\to P$ 是可终止的，由此可知 P 也是可终止的。表现为前缀表达式进程的迹是一种事件节点有限的有序线形结构。

前缀进程的迹定义如下：$traces(e1\to P)=\{\langle\ \rangle\}\cup\{\langle e1\rangle^\wedge s\mid s\in traces(P)\}$，其中 $e1\in\alpha P$，$\alpha(e1\to P)\in\alpha P$。等价 Petri 网是一个位置和迁移的序列，其中每个位置代表进程的状态，每个迁移代表进程迹的事件。前缀进程转化为 Petri 网的步骤如下：

1)确定每一个 CSP 进程的状态集和事件集。对任何事件 e 的每一次出现均关联两个状态$'s_e$ 和 s_e'，分别表示出现该事件之前和之后的情况。

2)将所有的状态 s 用 Petri 网位置表示(仍标记为 s)，所有事件 e 的出现用 Petri 网迁移表示(仍标记为 e)，对每一次出现 e，增加从$'s_e$ 到 e，以及从 e 到 s_e'的有向弧。

例：前缀进程 $Q=a\to b\to P$，其中 $P=c\to d\to STOP$。

第一步，与事件 a 关联的前后状态为 $s1$、$s2$，与事件 b 关联的前后状态为 $s2$、$s3$，同理，可得事件 c 的状态为 $s3$、$s4$，事件 d 的状态为 $s4$、$s5$；

第二步，用有向弧将事件与对应的前后状态相连，可得最终等价 Petri 网，如图 1 所示。

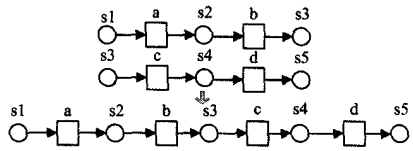


图 1 前缀进程的 Petri 网转换

规则 2 递归进程

定义 8(递归) 设 X 是一个进程变量，$A=\alpha X$，则递归进程的一种表示为 $X=F(X)$，其中，$F(X)$为包含进程变量 X 的前缀表达式。该递归方程在事件集合 A 上的唯一解为 $\mu X:A\cdot F(X)$，其中 X 为局部变量，A 为进程 P 的字母表，$F(X)$为包含进程 P 的前缀表达式。

递归进程的迹是由事件节点组成的有序环形结构。假设 P 是一个递归进程，$\alpha\ trace(P)=\{t\}$，t 的形式为$\langle e1,e2,e3,\cdots,eq\rangle^*$，其中 $e1,e2,\cdots,eq$ 都是进程 P 的事件。前缀进程转化为 Petri 网的步骤如下：

1)同前缀进程转化步骤 1)；

2)将所有的状态 s 用 Petri 网位置表示(仍标记为 s)，所有事件 e 的出现用 Petri 网迁移表示(仍标记为 e)，对每一次出现 e，增加从$'s_e$ 到 e，以及从 e 到 s_e'的有向弧，直到再次出现所求进程 P；

3)然后删除与就近事件关联的状态 s_e'并将其有向弧指向初始状态，产生一个循环 Petri 网。

例：递归进程 $P=a\to b\to P$。

第一步，按照前缀进程规则，与事件 a 关联的前后状态为 $s1$、$s2$，与事件 b 关联的前后状态为 $s2$、$s3$；

第二步，用有向弧将事件 a、b 与对应的前后状态相连，如图 2 所示；

第三步，进程 P 开始递归前的最后一个事件是 b，因此删

除状态 $s3$，并添加指向初始状态的输入弧，产生等价的循环 Petri 网，如图 2 所示。

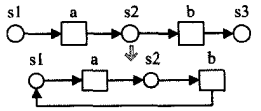


图 2　递归进程的 Petri 网转换

规则 3　选择进程

CSP 体系中前缀与递归表示可以描述顺序执行事件的客体，即单支结构的进程。然而，有些客体本身存在多种行为执行可能，其在与外界环境的相互作用下，实时决定行为事件的执行轨迹，即该进程是一种本身具有分支结构的进程。外界环境影响是指，或由外界环境确定性地选择（即确定性选择，又称外部选择）决定执行其中一个分支行为，或由内部自身非确定地选择（即非确定性选择，又称内部选择）决定执行其中一个分支子进程。习惯上，称选择进程中由外部环境确定的进程为确定性进程，由内部确定的进程为非确定性进程。

定义 9（确定性选择）　设 x 和 y 是两个不同事件，P 和 Q 是两个进程，则 $R=(x\rightarrow P\ \square y\rightarrow Q)$ 表示若进程 R 第一个发生的事件是 x，则客体的后续行为按 P 进行动作；如果发生的第一个事件是 y，则客体的后续行为按 Q 进行动作。其中，$\alpha(x\rightarrow P\square y\rightarrow Q)=\alpha P=\alpha Q$。

确定性选择进程是一种具有分支结构的进程，外部环境的选择决定执行哪一条迹。假设 P 和 Q 是可选择进程，即复合进程的行为是 P 或者 Q，转换步骤如下：

1）依照上述前缀进程转化规则，将进程 P、Q 分别转化为对应的 Petri 子网；

2）根据确定性选择进程的特点，需添加一个空事件表示外部环境，对应的两条有向弧连接两个子网的初始状态；

3）在进程 P、Q 对应的初始事件之间添加一个空状态表示新的初始位，在添加的状态中放置一个令牌并将其定义为初始标记。

例：进程 $P=a\rightarrow b\rightarrow STOP$，$Q=c\rightarrow d\rightarrow STOP$，则 P 和 Q 的确定性选择进程为 $(a\rightarrow b\rightarrow STOP)\square(c\rightarrow d\rightarrow STOP)$。

第一步，进程 P 中与事件 a 关联的前后状态为 $s1$、$s2$，与事件 b 关联的前后状态为 $s2$、$s3$，同理，可得进程 Q 中事件 c 的状态为 $s4$、$s5$，事件 d 的状态为 $s5$、$s6$，用有向弧将事件与对应的前后状态相连，如图 3 所示；

第二步，在两个子网对应的初始状态 $s1$，$s4$ 之间添加一个空事件 environment 表示外部环境，由外部环境来选择执行哪个进程；

第三步，在两个进程对应的初始事件 a，c 之间添加一个状态 $s0$，在 $s0$ 中添加一个令牌表示初始标记，生成最终 Petri 网，如图 3 所示。

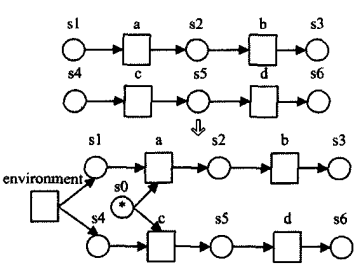


图 3　确定性进程的 Petri 网转换

定义 10（非确定性选择）　设 P 和 Q 是两个进程，则 $R=P\prod Q(P$ 或 $Q)$ 表示进程 R，或者按 P 动作，或者按 Q 动作，R 在两者之间的选择是在外部环境不知道或不能控制的情况下任意进行的，即是内部非确定性的任意选择。其中，$(\alpha P\prod Q)=\alpha P=\alpha Q$。

假设 P 和 Q 是可选择进程，即复合进程的行为是 P 或者 Q，转换步骤如下：

1）依照上述前缀进程转化规则，将进程 P、Q 分别转化为对应的 Petri 子网。

2）根据非确定性选择进程的特点，需要添加一个空状态和两个空事件及对应的有向弧才能保证 Petri 网的完整性，因为只有通过令牌的迁移才能确定选择哪个进程，这就需要一个公共的状态来分配令牌。

3）在添加的空状态中放置一个令牌，并将其定义为初始标记。

例：进程 $P=a\rightarrow b\rightarrow STOP$，$Q=c\rightarrow d\rightarrow STOP$，则 P 和 Q 的非确定性选择进程为 $(a\rightarrow b\rightarrow STOP)\prod(c\rightarrow d\rightarrow STOP)$。

第一步，进程 P 中与事件 a 关联的前后状态为 $s1$、$s2$，与事件 b 关联的前后状态为 $s2$、$s3$，同理，可得进程 Q 中事件 c 的状态为 $s4$、$s5$，事件 d 的状态为 $s5$、$s6$，用有向弧将事件与对应的前后状态相连，如图 4 所示；

第二步，添加空状态 $s0$ 和对应的两个空事件 m、n；

第三步，在 $s0$ 中添加一个令牌表示初始标记，生成最终 Petri 网，如图 4 所示。

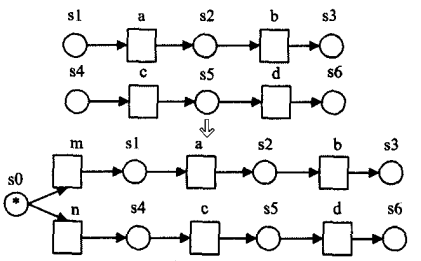


图 4　非确定性进程的 Petri 网转换

规则 4　并发进程

定义 11（并发）　设 P 和 Q 是进程，$P\neq Q$，P 和 Q 并发组合所得进程对于 P，Q 共同拥有的事件共同执行，对其他事件则交替执行，记为：$P\parallel Q$，其中 $\alpha(P\parallel Q)=\alpha P\cup\alpha Q$，“$\parallel$”称为进程的并发组合算子。

进程 P，Q 的字母表有一个非空的交互，转换步骤如下：

1）通过上述前缀进程转化规则，将进程 P、Q 分别转化为对应的 Petri 子网。

2）识别两个 Petri 子网相同的事件部分，并将其合并。

3）若没有相同的事件，则其余部分各自独立进行转换。这时需要添加一个空状态和一个空事件及对应的弧才能保证 Petri 网的完整性，因为只有通过令牌的迁移才能确定每个进程独立执行。

4）在添加的空状态中放置一个令牌，并将其定义为初始标记。

例：进程 $P=a\rightarrow b\rightarrow c\rightarrow STOP$，$Q=d\rightarrow b\rightarrow e\rightarrow STOP$，则 P 和 Q 的并发进程为 $(a\rightarrow b\rightarrow c\rightarrow STOP)\parallel(d\rightarrow b\rightarrow e\rightarrow STOP)$。

第一步，进程 P 中与事件 a 关联的前后状态为 $s1$、$s2$，与事件 b 关联的前后状态为 $s2$、$s3$，与事件 c 关联的前后状态为

$s3$、$s4$，同理，可得进程 Q 中事件 d 的状态为 $s5$、$s6$，事件 b 的状态为 $s6$、$s7$，事件 e 的状态为 $s7$、$s8$，用有向弧将事件与对应的前后状态相连，如图 5 所示；

第二步，进程 P、Q 转化后的子网有相同的事件 b，因此将事件 b 合并；

第三步，添加空状态 $s0$ 和对应的空事件 m，并用有向弧连接对应的各个结点；

第四步，将初始标记添加到 $s0$，得到最终 Petri 网，如图 5 所示。

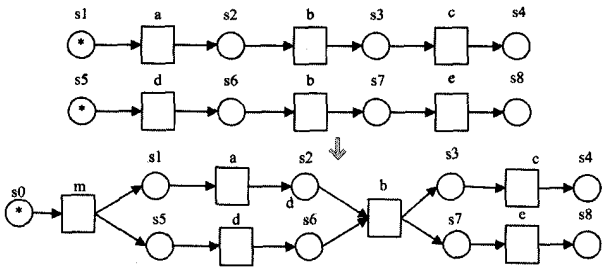


图 5　并发进程的 Petri 网转换

3.2　CSP 进程与 Petri 网的等价性

由以上规则转化所得 Petri 网与原 CSP 进程的等价性可由文献[11]所阐述的思想来说明。其基本思想是引入 Petri 网语言的概念，CSP 进程 S 与一个 Petri 网 N 等价，当且仅当进程 S 的迹与 Petri 网 N 产生的语言相同。

例：假设有一台自动售货机 VMS，投入一枚硬币记作 $s0$，出来一块巧克力记作 $s1$，打印一张发票记作 $s2$。

由定义 1 可得，该自动售货机 VMS 的迹 $traces(VMS)=\{\langle\rangle,\langle s0\rangle,\langle s0,s1\rangle,\langle s0,s1,s2\rangle\}$。

Petri 网语言的定义如下：

定义 12[11]　给定一个 Petri 网 $N=(\langle P,T;F\rangle,M_0,\rho,\tau,L_P,L_T)$，迁移 T 的所有可能引发序列记作 $FS(N)$：

$$FS(N)=\bigcup_{M\in R(M)}\{\sigma\mid M_0\xrightarrow{\sigma}M\}$$

定义 13[11]　Petri 网 N 所产生的语言 $L_A(N)$ 定义为：

$$L_A(N)=\{\langle\rangle\}\cup\{\langle L_T(s_1),\cdots,L_T(s_m)\rangle\mid\exists\sigma=t_1\cdots t_n\in FS(N)\wedge(\forall i,1\leqslant i\leqslant m\wedge(m\leqslant n\leqslant 1\wedge k\leqslant n:s_i=t_k)\wedge L_T(t_k)\in A\wedge(\neg\exists j,1\leqslant j<i\wedge l\geqslant k:s_j=t_l)\}$$

其中，ρ 是位置字母表，τ 是迁移字母表；L_P 和 L_T 都是标记函数，$L_P:P\to\rho$，$L_T:T\to\tau$，假设 $A=\tau$。

CSP 进程与 Petri 网等价的定义如下：

定义 14[11]　给定一个 CSP 进程的迹描述记作 S 和一个 Petri 网记作 N，称 S 和 N 是等价的当且仅当 $traces(S)=L_\Sigma(N)$，其中下标 Σ 表示迁移字母表。

例：上例中的自动售货机 VMS，利用本文所给的规则可以转化为图 6 所示的 Petri 网 N，其中状态（位置）字母表 $\rho=\{a,b,c,d\}$，事件（迁移）字母表 $\tau=\{s0,s1,s2\}$。

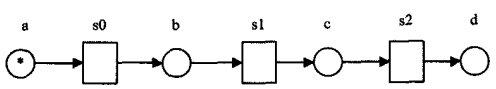


图 6　自动售货机 VMS 转化成的 Petri 网

由定义 13 可得，该 Petri 网语言 $L_\tau(N)=\{\langle\rangle,\langle s0\rangle,\langle s0,s1\rangle,\langle s0,s1,s2\rangle\}$。显然，$traces(VMS)=L(N)$。

3.3　举例说明

针对铁路公路交叉口交通控制器，建立一个安全的系统，确保：当有火车经过时，铁路栅门放下，红灯亮，绿灯灭；当火车离开时，铁路栅门抬起，绿灯亮，红灯灭。

如图 7 所示，假设 $P1$，$P2$ 是两个信号灯，有一辆火车沿 $P1$ 到 $P2$ 的方向运行。该控制系统可以由 3 个进程组成，分别为火车（T），栅门（G）和信号灯（L）。

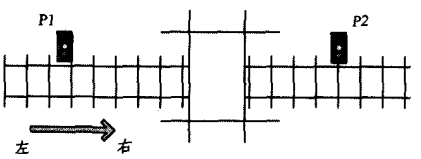


图 7　铁路公路交叉口交通控制器

火车的事件集：

$te1$—火车到达信号传送到 P1；

$te2$—火车通过公路；

$te3$—火车离开信号传送到 P2。

栅门事件集：

$ge1$—P1 红灯亮信号传送给栅门；

$ge2$—栅门放下；

$ge3$—P2 绿灯亮信号传送给栅门；

$ge4$—栅门抬起。

信号灯事件集：

$le1$—火车到达信号传送到 P1；

$le2$— P1 红灯亮信号传送给栅门；

$le3$—火车离开信号传送到 P2；

$le4$— P2 绿灯亮信号传送给栅门。

通过上述三者的事件集可知，火车事件集中的 $te1$ 和 $te3$ 分别等价信号灯事件集中的 $le1$ 和 $le3$，栅门事件集中的 $ge1$ 和 $ge3$ 分别等价信号灯事件集中的 $le2$ 和 $le4$。于是，3 个进程的字母表分别是：

$\alpha T=\{te1,te2,te3\}$；$\alpha G=\{ge1,ge2,ge3,ge4\}$；$\alpha L=\{te1,ge1,te3,ge3\}$。

下面考虑三者的状态集，结合实际情况，每个进程的事件前后都有对应的状态，火车进程对应的状态集为：{approaching（靠近公路），scross（开始通过公路），ecross（结束通过公路），leaving（正在离开）}；栅门对应的状态集为：{open（栅门打开），closing（正在关闭），close（关闭），opening（正在打开）}；信号灯对应的状态集为：{waiting（等待），redsig（红灯），leavesig（离开），greensig（绿灯），reset（重置）}。

最后得到 3 个进程的迹如下：

$traces(T)=\{\langle\rangle\}\cup\{\langle te1,te2,te3\rangle^*\}$

$traces(G)=\{\langle\rangle\}\cup\{\langle ge1,ge2,ge3,ge4\rangle^*\}$

$traces(L)=\{\langle\rangle\}\cup\{\langle te1,ge1,te3,ge3\rangle^*\}$

根据 3.1 节给出的转化规则可得到与 3 个进程等价的 Petri 网，如图 8—图 10 所示。

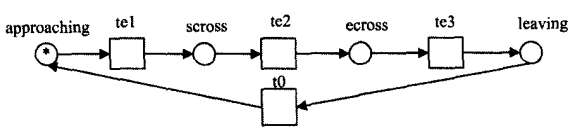


图 8　火车 T 进程对应的 Petri 网

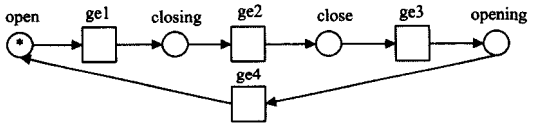


图 9　栅门 G 进程对应的 Petri 网

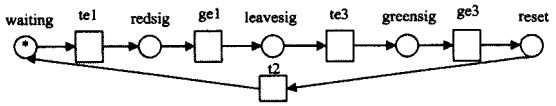


图 10　信号灯 L 进程对应的 Petri 网

根据上述规则，两两首先进行并发，可得下图。

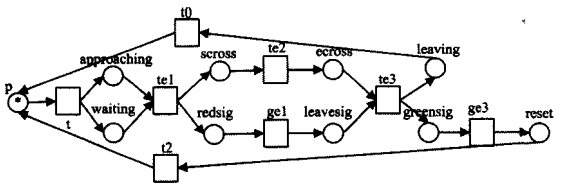


图 11　$T \| L$ 并发进程对应的 Petri 网

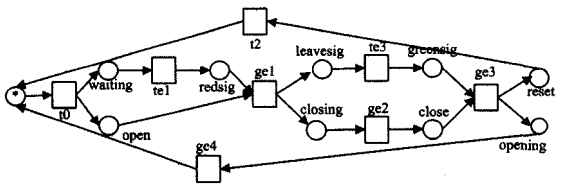


图 12　$G \| L$ 并发进程对应的 Petri 网

由图 11 和图 12 可知，两对并发进程都有信号灯进程的参与。下面，将 3 个进程并发组合时，会出现两个信号灯进程，有一个是冗余信息。根据 Petri 网的消解原理[2]，可以消除冗余，得到最终并发图，如图 13 所示。

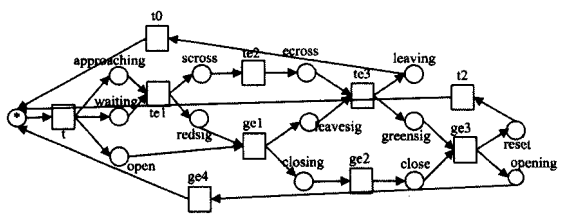


图 13　$T \| L \| G$ 并发进程对应的 Petri 网

4　实验

本节以铁路公路交叉口交通控制器为例，首先给出每个进程的迹描述，然后把该描述转化为 Petri 网，利用相关分析工具进行系统性质的分析和验证。

4.1　实例描述

3.3 节给出的铁路公路交叉口交通控制器对应的 CSP 进程描述如图 14 所示。

1. Train = approaching → te1 → scross → te2 → ecross → te3 → leaving → t0 → Train;
2. Light = waiting → te1 → redsig → ge1 → leavesig → te3 → greensig → ge3 → reset → t2 → Light;
3. Gate = open → ge1 → closing → ge2 → close → ge3 → opening → ge4 → Gate;
4. Crossing = Light ‖ Gate;
5. System = Train ‖ Crossing;
6. #assert System deadlockfree;

图 14　CSP 并发系统描述

用模型分析工具 PAT[20] 对上述 CSP 并发系统描述进行自动化验证时，只能验证其无死锁性，不能验证该系统的安全性。为此，下面依本文方法把 CSP 转化为 Petri 网进行分析。

4.2　转化为 Petri 网后的安全性验证

虽然仅基于并发进程的 CSP 描述不能验证该并发系统的安全性，但转化为 Petri 网之后可以通过可达树[2]分析并确定危险因素是否存在以及存在的位置。对于图 13，其部分可达树如图 15 所示，树中的结点表示经过使能迁移的引发所产生的标识，结点之间的连线表示了标识和迁移之间的关系，其中 $t0$，$te1$，$ge1$ 等表示 Petri 网中的迁移。下图中的标识（除去添加的位置结点 p）依次对应图 13 中的（open，waiting，approaching，redsig，scross，closing，leavesig，ecross，close，greensig，leaving，opening，reset）。

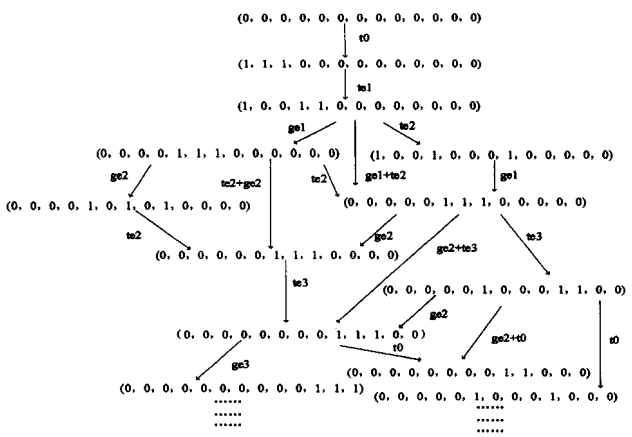


图 15　Petri 网的部分可达树

由图 15 可得，出现下面两种情况是非常危险的：1）当标识分别位于 open，redsig 和 scross 时，如果迁移 $te2$ 首先发生，则栅门在打开的情况下，火车通过公路；2）：当标识分别位于 closing，leavesig，ecross 时，如果迁移 $ge2$ 首先发生，则当火车离开后，栅门处于关闭状态。然而，通过上述 CSP 并发系统迹的描述，无法正确验证危险的存在，但 Petri 网的可达树能够分析出不安全的因素。

根据上述的分析结果可得，转化后的 Petri 网虽然存在危险因素，但可以通过增加时间区间消除不安全性，即用时间 Petri 网来进行后续的分析。在时间 Petri 网中，$eft(t)$ 和 $lft(t)$ 分别称作 t 的最早发生时间（earliest firing time）和最晚发生时间（latest firing time）。该时间限制可形式化地描述为：$eft(crossing) > lft(ge1) + lft(ge2)$，其中 $eft(crossing)$ 指火车通过栅门的最早发生时间（即所用最短时间），$lft(ge1)$ 指 P1 红灯信号传给栅门的最晚发生时间（即所用最长时间），$lft(ge2)$ 指栅门整个关闭过程所需最长时间。用检测工具 TINA 对改进后的 Petri 网（如图 16 所示）进行分析验证。

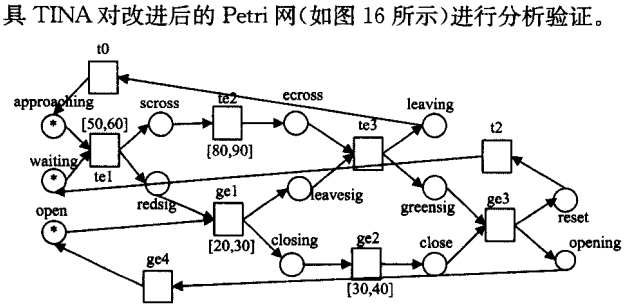
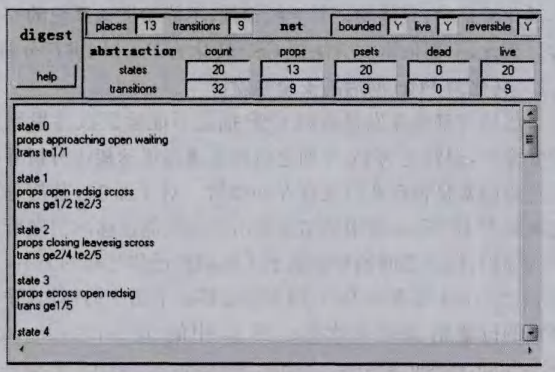


图 16　改进后的时间 Petri 网

图 16 的分析结果如图 17 所示。




```
Struct version 3.2.2--06/24/14--LAAS/CNRS
parsed net all_1
13 places, 9 transitions
net all_1
tr ge1[20,30] open redsig->closing leavesig
tr ge2[30,40] closing->close
tr ge3 close greensig->opening reset
tr ge4 opening->open
tr t0 leaving->approaching
tr t2 reset->waiting
tr te1[50,60] approaching waiting->redsig scross
tr te2[80,90] scross->ecross
tr te3 ecross leavesig->greensig leaving
pl approaching[1]
pl open[1]
pl waiting[1]
```

图 17　TINA 自动化检测结果分析

由图 17 分析可知，改进后的 Petri 网具有安全性、有界性和活性。同时，TINA 详细地给出了在每个状态下迁移的标签顺序和每一次迁移发生的步骤等可达性分析结果。

可见，将 CSP 并发系统转化为 Petri 网，可有效地分析系统的安全性，扩大 CSP 描述的并发系统可验证性质的范围。

结束语　本文研究了基于 Petri 网的 CSP 并发系统验证技术，弥补了仅基于并发系统的 CSP 描述的安全性质无法被验证的缺点。首先利用 CSP 描述待验证的并发系统，全面考虑了 CSP 的确定性和非确定性选择进程，然后将并发系统转化为 Petri 网来分析动态行为特性，利用性质分析工具 TINA 对系统性质进行分析和验证。实验表明将 CSP 进程转化为 Petri 网可允许在设计过程的早期阶段进行错误检测，有效地分析出危险因素，从而扩大 CSP 描述的并发系统可验证性质的范围。此外，转化后的 Petri 网还部分消除了 CSP 并发操作中的冗余信息。未来将研究基于 Petri 网的 CSP 并发系统调试技术，并把本文成果推广应用于同步/异步通信系统的验证。

参 考 文 献

[1] Hoare C A R. Communicating Sequential Processes [M]. Prentice Hall International, 2004: 45-108

[2] 袁崇义. Petri 网应用[M]. 北京：科学出版社，2013：31-55
Yuan Chong-yi. The Application of Petri Nets [M]. Beijing: Science Press, 2013: 31-55

[3] 蒋昌俊. Petri 网的行为理论及其应用[M]. 北京：高等教育出版社，2003：8-9
Jiang Chang-jun. The Behavior Theory of Petri Nets and its Application [M]. Beijing: Higher Education Press, 2003: 8-9

[4] Girault C, Valk R. 系统工程 Petri 网—建模、验证与应用指南[M]. 王生原，余鹏，霍金健，译. 北京：电子工业出版社，2005：60-130

[5] Cindio F, Michelis G, Pomello L, et al. A Petri net model of CSP [C]// Proceedings of Convención Informática Latina (CIL'81). Barcelona, 1981: 392-406

[6] Goltz U, Reisig W. CSP-programs as nets with individual tokens [M]// Advances in Petri Nets 1984. Springer, 1985: 169-196

[7] Mazzeo A, Mazzocca N, Russo S, et al. Formal specification of concurrent systems: a structured approach [J]. The Computer Journal, 1998, 41(3): 145-162

[8] Mazzeo A, Mazzocca N, Russo S, et al. A Systematic Approach to the Petri Net Based Specification of Concurrent Systems[C]// Proceedings of the Safety-Critical Real-Time Systems. Napoli, Italy, 1997: 3-20

[9] Mazzocca N, Russo S, Vittorini V. Integrating trace logic and Petri nets specifications [C]// Proceedings of the 30th Hawaii International Conference on System Sciences: Software Technology and Architecture. 1997: 443-451

[10] Llorens M, Oliver J, Silva J, et al. Transforming communicating sequential processes to Petri nets[C]// Proceedings of the 7th International Conference on Engineering Computational Technology(ECT 2010). Val-encia, Spain, 2010: 1-16

[11] Llorens M, Oliver J, Silva J, et al. Generating a Petri net from a CSP specification: A semantics-based method[J]. Advances in Engineering Software, 2012, 50(1): 110-130

[12] Baldan P, Bonchi F, Gadducci F. Encoding Asynchronous Interactions Using Open Petri Nets[C]// Proceedings of the 20th International Conference, CONCUR 2009. Bologna, Italy, 2009: 99-114

[13] Baldan P, Bonchi F, Gadducci F. Encoding Synchronous Interactions Using Labelled Petri Nets[C]// Proceedings of the 16th IFIP WG 6.1 International Conference, COORDINATION 2014. Berlin, Germany, 2014: 1-16

[14] Magott J. Performance evaluation of communicating sequential processes(CSP) using Petri nets[J]. IEE Proceedings E: Computers and Digital Techniques, 1992, 139(3): 237-241

[15] Sheldon F T, Kavi K M, Ka-mangar F A. Reliability Analysis of CSP Specifications: a New Method Using Petri Nets[C]// Proceedings of the AIAA Computing in Aerospace Conference. USA, 1995: 317-326

[16] Berthomieu B, Ribet P O, Vernadat F. The tool TINA——Construction of Abstract State Spaces for Petri Nets and Time Petri Nets[J]. International Journal of Production Research, 2004, 42(14): 2741-2756

[17] Berthomieu B, Vernadat F. Time Petri Nets Analysis with TINA[C]// Proceedings of the 3rd International Conference on the Quantitative Evaluation of Systems (QEST 2006). Riverside, CA, 2006: 123-124

(下转第 291 页)

表 4　实时跟踪算法与 CT 算法的比较

	实时跟踪算法	CT 算法
David 平均跟踪速度	22.3194 帧/s	9.4910 帧/s
David 序列准确率	98%	89%
Kitesurf 速度	12.5791 帧/s	7.5897 帧/s
Kitesurf 准确率	41.67%	38.55%
Sylv 速度	20.1109 帧/s	9.2533 帧/s
Sylv 准确率	88.81%	60.07%

在运行过程中，跟踪学习检测算法能够准确地跟踪目标，在目标大小发生改变的情况下，跟踪的目标框大小也有相应的改变。CT 算法也可以准确地跟踪目标，但是 CT 算法在跟踪的过程中，目标框发生了一定程度的漂移，并且目标框大小不能随着目标大小的改变而改变。

以上实验表明，跟踪学习检测算法的运行速度大大超过 CT 算法；并且实时跟踪算法能准确地框出目标位置，目标框能随着目标大小的变化而变化。

结束语　针对 TLD 运行速度慢及在实时跟踪场合准确率低的问题，本文提出了基于动态滑动扫描框的检测方法，实现了实时视频跟踪算法。实时跟踪算法能对目标进行实时准确的跟踪，对安防监控、交通监控等实时监控领域有巨大意义。

虽然实时跟踪学习检测算法具有优秀的运行速度和准确率，但是在目标被遮挡的时，其检测的准确率会下降，而且它目前只能检测单个目标。针对这两个问题，将来的工作可以分为以下两步：(1)参考霍夫森林算法[16]，基于随机森林和霍夫投票来提高在目标被遮挡时检测的准确率。(2)参考多目标信息融合算法，比如 PHD 算法[17]，将其与实时跟踪学习检测算法进行融合，使其可以实现多目标的跟踪[18,19]。

参 考 文 献

[1] Kalal Z, Mikolajczyk K, Matas J. Tracking-learning-detection [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2012, 34(7): 1409-1422

[2] Kalal Z, Matas J, Mikolajczyk K. Pn learning: Bootstrapping binary classifiers by structural constraints[C]// 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2010: 49-56

[3] Fernando W S P, Udawatta L, Pathirana P. Identification of moving obstacles with pyramidal Lucas Kanade optical flow and kmeans clustering[C]// Third International Conference on Information and Automation for Sustainability, 2007 (ICIAFS 2007). IEEE, 2007: 111-117

[4] Kalal Z, Mikolajczyk K, Matas J. Forward-backward error: Automatic detection of tracking failures[C]// 2010 20th International Conference on Pattern Recognition (ICPR). IEEE, 2010: 2756-2759

[5] 江博. 基于 Kalman 的 TLD 目标跟踪算法研究[D]. 西安：西安科技大学，2013
Jiang Bo. The research of TLD target tracking algorithm based on Kalman[D]. Xi'an: Xi'an University of Science and Technology, 2013

[6] 高帆，吴国平，刑晨，等. TLD 目标跟踪算法研究[J]. 电视技术，2013, 37(11): 70-74, 202
Gao fan, Wu Guo-ping, Xing Chen, et al. TLD target tracking algorithm[J]. Video Engineering, 2013, 37(11): 70-74, 202

[7] Zhang K, Zhang L, Yang M H. Real-time compressive tracking [M]// Computer Vision-ECCV 2012. Springer Berlin Heidelberg, 2012: 864-877

[8] Bradski G, Kaehler A. Learning OpenCV: Computer vision with the OpenCVlibrary[M]. O'Reilly Media, Inc., 2008

[9] 孙凯，刘士荣. 多目标跟踪的改进 Camshift/卡尔曼滤波组合算法[J]. 信息与控制，2009, 38(1): 9-14
Sun Kai, Liu Shi-rong. The optimism of multi-target tracking algorithm the combination of Camshift and Kalman[J]. Information and Control, 2009, 38(1): 9-14

[10] Babenko B, Yang M H, Belongie S. Robust object tracking with online multiple instance learning[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2011, 33(8): 1619-1632

[11] Grabner H, Grabner M, Bischof H. Real-Time Tracking via Online Boosting[J]. BMVC, 2006, 1(5): 6

[12] Grabner H, Leistner C, Bischof H. Semi-supervised on-line boosting for robust tracking[M]// Computer Vision-ECCV 2008. Springer Berlin Heidelberg, 2008: 234-247

[13] Adam A, Rivlin E, Shimshoni I. Robust fragments-based tracking using the integral histogram[C]// 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE, 2006: 798-805

[14] Mei X, Ling H. Robust visual tracking and vehicle classification via sparse representation[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2011, 33(11): 2259-2272

[15] Hare S, Saffari A, Torr P H S. Struck: Structured output tracking with kernels[C]// 2011 IEEE International Conference on Computer Vision (ICCV). IEEE, 2011: 263-270

[16] Gall J, Lempitsky V. Class-specific hough forests for object detection[M]// Decision Forests for Computer Vision and Medical Image Analysis. Springer London, 2013: 143-157

[17] Clark D E, Bell J. Multi-target state estimation and track continuity for the particle PHD filter[J]. IEEE Transactions on Aerospace and Electronic Systems, 2007, 43(4): 1441-1453

[18] Zhou Xiao-long, Li Y F, He Bing-wei, et al. GM-PHD-Based Multi-Target Visual Tracking Using Entropy Distribution and Game Theory[J]. IEEE Transactions on Industrial Informatics, 2014, 10(2): 1064-1076

[19] Zhou Xiao-long, Li Y F, He Bing-wei. Game-Theoretical Occlusion Handling for Multi-Target Visual Tracking[J]. Pattern Recognition, 2013, 46(10): 2670-2684

(上接第 250 页)

[18] Louazani A, Sekhri L, Kechar B. A time Petri net model for wormhole attack detection in wireless sensor networks[C]// Proceedings of the 2013 International Conference on Smart Communications in Network Technologies (SaCoNeT). Paris, France, 2013: 1-6

[19] Adjir N, de Saqui-Sannes P, Rahmouni K M. Conformance Testing of Preemptive Real-Time Systems[J]. International Journal of Embedded and Real-Time Communication Systems, 2013, 4(4): 1-26

[20] Liu Y, Sun J, Dong J S. Pat 3: An extensible architecture for building multi-domain model checkers[C]// 2011 IEEE 22nd International Symposium on Proceedings of the Software Reliability Engineering (ISSRE). Hiroshima, Japan, 2011: 190-199