带抑制弧的时延着色 Petri 网模型检测技术

杨年华[1,2] 虞慧群[1,2] 孙 华[1]

(华东理工大学计算机科学与工程系 上海 200237)[1] (上海市计算机软件评测重点实验室 上海 201112)[2]

摘 要 带抑制弧的时延着色 Petri 网(Timed Colored Petri Nets with Inhibitor Arcs,TCPNIA)是一种描述实时嵌入式系统的模型。给出了从 TCPNIA 到时间自动机的结构化转换算法,以利用变迁冲突调解机制保证 TCPNIA 模型和转换后的时间自动机模型语义等价;并给出了语义等价的证明和算法复杂度分析。层次化方法被用来提高模型检测的时间与空间效率。通过实际案例展示了该技术的应用和可行性。

关键词 时延着色 Petri 网,抑制弧,时间自动机,冲突调解,模型检测

中图法分类号 TP302 **文献标识码** A

Model Checking of Timed Colored Petri Nets with Inhibitor Arcs

YANG Nian-hua[1,2] YU Hui-qun[1,2] SUN Hua[1]

(Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai 200237, China)[1]

(Shanghai Key Laboratory of Computer Software Evaluating and Testing, Shanghai 201112, China)[2]

Abstract TCPNIA(Timed Colored Petri Nets with Inhibitor Arcs, TCPNIA) is a model for specifying embedded systems. This paper proposed a structural transformation method from TCPNIA to TA(Timed Automata, TA). A collision mediation mechanism was introduced to ensure the semantics equivalence between TCPNIA and the transferred counterpart. The semantics equivalence was proved. The complexity of the transformation algorithm was analyzed. Hierarchical method was utilized to improve time and space efficiency in model checking. A case study shows the applicability and feasibility of the technique.

Keywords Timed colored Petri net, Inhibitor arc, Timed automaton, Collision mediation, Model checking

实时嵌入式系统已经广泛应用于通信、汽车、铁路、航空、核电站等控制系统。这些系统的失效或者实时性要求得不到满足都将引起灾难性的后果。形式化方法是保证系统安全性、可靠性、实时性等可信性特征的有效手段。Petri 网[1]是一种基于严格数学理论的形式化方法。针对嵌入式系统建模的需要,研究者们提出了对 Petri 网的各种改进模型[2-4]。这些模型要么无法描述任务优先级关系,要么无法利用传统 Petri 网的分析技术。着色 Petri 网可以表达多种数据,时延 Petri 网可表示完成一个特定功能的时延,抑制弧可以描述任务的优先级。带抑制弧的时延着色 Petri 网(Timed Colored Petri Net with Inhibitor Arcs, TCPNIA)[5]以这 3 种 Petri 网为基础,引入了控制函数与变迁函数,适合描述复杂的实时嵌入式系统,又能利用传统 Petri 网的分析技术。

模型检测[6,7]是验证系统功能模型是否满足 CTL[6]或 TCTL[8]等公式描述需求规范的形式化方法。目前已有比较成熟的模型检测工具,如 UPPAAL[9],Kronos[10]。但是这些工具都是以时间自动机(Timed Automata, TA)[11]作为功能模型的描述工具。为了利用已有模型检测工具来验证 TCPNIA 模型的正确性,需要将 TCPNIA 模型转换为 TA。

时间 Petri 网(time Petri net)到 TA 的转换已经有很多成熟的研究成果[12-14],尤其是 F. Cassez 等[14]给出了时间 Petri 网和时间自动机的语义对应关系,并且给出了转换的正确性证明。但时延 Petri 网(Timed PN)到 TA 的转换算法研究却比较少[15],其主要原因在于时延 Petri 网变迁执行过程需要消耗时间,在转换后的 TA 模拟过程中可能引起其它变迁自动机的错误并发,从而导致错误状态的引入。本文在 Z. Gu 方法[15]的基础上,提出了一种将 TCPNIA 转换为 TA 的算法。该算法引入变迁冲突调解机制来解决自动机的错误并发,使其执行语义与时延 Petri 网等价。此算法也可以用于解决一般时延 Petri 网转换成语义等价的 TA 问题。

层次化建模是提高系统重用性、降低模型复杂性的有效方法。层次化验证能降低模型验证的复杂性,提高模型验证效率,减轻模型检测中状态空间爆炸问题,是验证大型、复杂系统的有效途径。文献[16-18]对 Petri 网的层次化方法进行了研究。

本文主要贡献在于:i. 针对时延 Petri 网,提出变迁冲突调解机制,防止 TA 系统并行执行过程中出现错误状态;ii. 给出了 TCPNIA 中的时延变迁语义,提出了 TCPNIA 到 TA 转

到稿日期:2010-03-03 返修日期:2010-05-18 本文受国家自然科学基金(60473055,60773094),上海市曙光计划(07SG32)资助。

杨年华(1978—),男,博士生,CCF 学生会员,主要研究方向为可信计算、嵌入式系统、形式化方法等,E-mail:cnynh@163.com;虞慧群(1967—),男,博士,教授,博士生导师,主要研究方向为软件工程、高可信计算和形式化方法等;孙 华(1977—),女,博士生,主要研究方向为信息安全、信誉管理等。

换的结构化方法，并给出了 TCPNIA 到 TA 转换算法的语义等价正确性证明和复杂性分析；iii. 利用层次化验证方法降低模型验证的复杂性，提高验证的时间与空间效率。

本文第 1 节给出 TCPNIA 的形式化定义；第 2 节介绍时间自动机与模型检测方法；第 3 节给出 TCPNIA 到 TA 的转换算法及其正确性证明；第 4 节通过实例来描述 TCPNIA 模型到 TA 的转换及 TCPNIA 的层次化验证方法；最后给出本文的结论与下一步的工作。

1 带抑制弧的时延着色 Petri 网(TCPNIA)

1.1 TCPNIA 语法

定义 1 TCPNIA 是一个元组 $\Sigma=\langle P,T,C,K,W,I,G,F,D,TS,M_0\rangle$。

其中，$P=\{p_1,p_2,\cdots,p_m\}$ 表示一个有限的库所集合；$T=\{t_1,t_2,\cdots,t_n\}$ 表示一个有限的变迁集合；$C\subseteq(P\times T)\cup(T\times P)$ 是一组有限的弧集；K 表示库所容量，即 $\forall p\in P,K(p)=1$；W 表示弧上的权，即 $\forall c\in C,W(c)=1$；M_0 是 Σ 的初始标识。(P,T,C,K,W) 为一个普通 Petri 网结构。$I\subset P\times T$ 称为抑制弧，$I\cap C=\emptyset$；$G:2^Z\to\{\text{true},\text{false}\}$ 是定义在变迁 T 上的控制函数，如果没有定义，则表示 $G=$ true；F 表示变迁函数；D 表示每个变迁发生所需持续的时间区间 $[\alpha(t),\beta(t)]$，即变迁 t 需要持续 $d(t)(\alpha(t)\leqslant d(t)\leqslant\beta(t))$ 才能完成；$TS:P\to R^+\cup\{0\}$ 表示库所中 token 被更新的时间戳，初始值均为 0；M_0 为初始标识。

用 ${}^\circ t=\{p\in P|(p,t)\in C\}$ 表示变迁 t 的控制前集；$t^\circ=\{p\in P|(t,p)\in C\}$ 表示变迁 t 的控制后集；${}^*t=\{p|(p,t)\in I\}$ 表示抑制前集。类型函数 $\tau(p)$ 表示库所中 token 值的类型。某个特定的库所，其类型始终不变。规定一个给定变迁 t 的所有后集库所 token 中的值具有相同的数据类型，用 $\tau(t^\circ)$ 表示。库所 P 中的标记 $M=\langle v,ts\rangle$，其中 v 表示数据值，如果该库所表示一个控制信息，那么 v 可以为任何值；如果 v 为空，则表示 token 为 0，否则 token 为 1；ts 表示 token 被更新的时间戳。

用 L 表示一个有限的谓词集合。$\forall p\in P,v_M(p)$ 表示在标识 M 下，库所 p 中 token 表示的数据值。定义在变迁 t 上的控制函数 G 是将 $\{p|p\in{}^\circ t\}=\{p_1,p_2,\cdots,p_a\}$ 中各个 token 表示的数据值映射到布尔空间的函数，即 $G:\tau(p_1)\times\tau(p_2)\times\cdots\times\tau(p_a)\to\{\text{true},\text{false}\}$。

定义 2 变迁 t 如果没有定义变迁函数，变迁激发完成后，变迁后集的 token 中 v 值可以为 $\tau(p)$ 中的任意值；如果在变迁 t 上定义了变迁函数，那么变迁激发完成后，变迁后集的 token 中 v 值为 $f_t:\tau(p_1)\times\tau(p_2)\times\cdots\times\tau(p_a)\to\tau(t^\circ)$，其中 ${}^\circ t=\{p_1,p_2,\cdots,p_a\}$。

TCPNIA 采用加权有向图来表示图形结构，圆圈表示库所，矩形表示变迁。其中弧边上的权为 1，弧上的字母代表关联库所中的 token 中 v 值的变量。矩形框内部左端(或上端)谓词公式表示变迁的控制函数，右端(或下端)公式表示变迁函数 F。矩形框边上的 $[\alpha(t),\beta(t)]$ 表示变迁激发所需时延的区间。从库所引向变迁的带箭头的弧以及箭头上的小圆圈一起表示抑制弧。图 1 表示两个整数相乘迭代算法的 TCPNIA 模型。

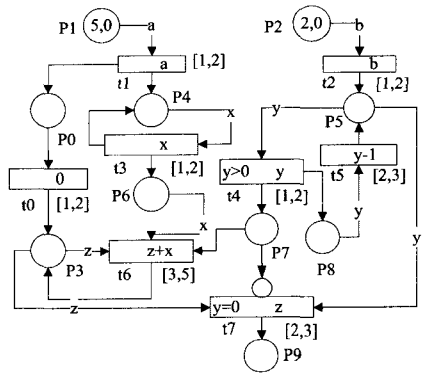

图 1 乘法器的 TCPNIA 模型

1.2 TCPNIA 语义

若一个变迁的所有控制前集中的库所均有 token，所有抑制前集中的库所均没有 token，不在控制前集中的控制后集库所没有 token，并且控制前集中库所的值满足变迁控制函数 $G_t=$ true 时，则变迁使能。可以形式化地定义如下。

定义 3 变迁 t 是使能的，当且仅当

$[\forall p_i\in{}^\circ t:M_{token}(p_i)=1]\wedge(G_t=\text{true})\wedge$

$[\forall p_j\in{}^*t:M_{token}(p_j)=0]\wedge$

$[\forall p_k\in t^\circ:(p_k\notin{}^\circ t\wedge M_{token}(p_k)=0)]$

一个使能变迁的激发使标识从 M^e 变为 M^{e+1}。定义 4 描述变迁的激发规则。以下用 $ts_{trigger}$ 描述变迁激发时间，$ts_{trigger}=\max_{i=1..a,k=1..b}(M^e_{ts}(p_i),M^e_{ts}(q_k))$，就是与变迁 t 有弧连接的所有库所标记的最大时间戳，其中 ${}^\circ t\cup t^\circ=\{p_1,p_2,\cdots,p_a\}$，${}^*t=\{q_1,q_2,\cdots,q_b\}$。

定义 4 一个使能变迁 t 被激发，使网标识从 M^e 变为 M^{e+1}，库所的标记通过以下规则依次发生变化：

i. $\forall p_i\in{}^\circ t$ 有 $M^{e+1}_{token}(p_i)=0,M^{e+1}_{ts}(p_i)=ts_{trigger}+d(t)$；

ii. $\forall p_j\in t^\circ$ 有 $M^{e+1}_{token}(p_j)=1,M^{e+1}_{ts}(p_j)=ts_{trigger}+d(t)$，其中 $\alpha(t)\leqslant d(t)\leqslant\beta(t)$ 表示变迁 t 的激发持续时间；如果 t 中有变迁函数 f_t，则 $v_{M^{e+1}}(p_j)=f_t(p_i,p_2,\cdots,p_a)$，其中 ${}^\circ t=\{p_1,p_2,\cdots,p_a\}$，否则 $v_{M^{e+1}}(p_j)=1$。

定义 4 中一个变迁 t 激发后，其控制前集中所有库所的 token 置空，库所中 ts 的值更改为变迁激发完成时间 $ts=ts_{trigger}+d(t)$。变迁 t 激发完成后，变迁 t 后集中库所的时间戳也为 $ts=ts_{trigger}+d(t)$。如果变迁中定义了变迁函数，则变迁后集库所中的 token 值按照变迁函数计算；如果没有定义变迁函数，则置为 1。

和传统的 Petri 网一样，多个使能的变迁只要不冲突，它们就可能同时激发。然而时延 Petri 网中变迁激发开始需要从输入库所中移走 token，直到变迁激发完成，在输出库所中放入新的 token，需要持续 $d(t)(\alpha(t)\leqslant d(t)\leqslant\beta(t))$ 个时间单位。在这 $d(t)$ 个时间单位中，一些冲突的变迁可能被激发，从而产生错误的状态。图 1 中 t_6 开始激发后，从 p_7 中移走了 token，从而使得 t_7 在 t_6 激发完成以前可能被激发，从而导致产生错误的状态。在模拟 TCPNIA 执行的程序中可以通过增加冲突条件的判断，避免与变迁 t 冲突的变迁 t_i 在 t 激发过程中被激发。

定义 5(冲突变迁) 给定 TCPNIA 中的某个变迁 t，$P_t=\{p|p\in{}^\circ t\}$，$P_t^{in}=\{p|p\in{}^\circ t\cup p\in{}^*t\}$，$Q_t=\{q|q\in t^\circ\}$，$\forall t_i\in T$，$t_i\neq t$，$P_{t_i}=\{p|p\in{}^*t_i\}$，$Q_{t_i}=\{q|q\in t_i{}^\circ\}$，$t_i$ 是变迁 t 的冲突

变迁，当且仅当$(P_t \cap P_{t_i} \neq \emptyset) \vee (Q_t \cap Q_{t_i} \neq \emptyset) \vee (Q_{t_i} \cap P_t^{in} \neq \emptyset)$；将变迁 t 的冲突变迁集合记为 $conf(t)$。

对有 n 个变迁的 TCPNIA 模型，可以用 $n \times n$ 的矩阵 MA 来表示所有变迁的冲突。矩阵中元素的值为 0 表示无冲突，1 表示冲突；矩阵 $MA[i,j]=1$ 表示变迁 t_i 在激发过程中，t_j 不能激发。对于变迁 t_j，如果 $\exists i, MA[i,j]=1$，并且变迁 t_i 正在激发过程中，那么变迁 t_j 不能激发，用函数 $hasCollision(t_j)$ 来表示变迁 t_j 处于这种冲突中。

我们用数组 tag[n]来表示变迁 t_i 是否正在激发过程中，如果 tag[i]=1 表示变迁 t_i 正在激发过程中，否则 tag[i]=0。所以变迁 t_i 在开始激发时需要将 tag[i]值置为 1，激发完成后置为 0。

TCPNIA 是对时延 Petri 网的一种扩展，它的语义也是对时延 Petri 网语义的一种扩展，可以用时间转换系统（Timed Transition Systems，TTS）[19]来描述。以 m 个库所，n 个变迁的 TCPNIA 为例，用 $tk \in (\{0,1\})^m$ 表示 m 个库所中各自的 token 数量；$v \in R^m$ 表示各个库所中 token 携带的数据值；$u \in (R_{\geqslant 0})^n$ 表示每个变迁自开始激发以来所流逝的时间；$f(v(^0t))$表示 t 的变迁函数。

定义 6（TCPNIA 语义） TCPNIA 模型的语义可以定义为一个时间转换系统 $TTS_N=(S,s_0,\rightarrow)$，其中 $S=(tk,v,tag,u)=(\{0,1\})^m \times R^m \times \{0,1\}^m \times (R_{\geqslant 0})^n$ 表示状态集合，$s_0=(tk_0,v_0,0,0)$表示初始状态，$\rightarrow \in S \times (T,R_{\geqslant 0}) \times S$ 由以下转换关系组成：

i. $(tk,v,tag,u) \xrightarrow{t_i^{pre}} (tk'',v,tag'',u'')$，如果 $tk(^\circ t_i)==1, tk(^* t_i)==0, tk(t_i^\circ)==0, \neg hasCollision(t_i), g_i(v), tk''(^\circ t_i)=0, tag''=tag[tag_i \mapsto 1], u''=u[u_i \mapsto 0]$；

ii. $(tk'',v,tag'',u'') \xrightarrow{d} (tk'',v,tag'',u''+d)$，对 $\forall d \in R_{\geqslant 0}$ 且 $d \leqslant ub$；

iii. $(tk'',v,tag'',u''+d') \xrightarrow{t_i^{pos}} (tk',v',tag',u'+d')$，如果 $\forall d' \in R_{\geqslant 0}, lb \leqslant d' \leqslant ub, tk'(t_i^\circ)=1, v'(t_i^\circ)=f_{t_i}(v(^0t_i)), tag'=tag''[tag_i'' \mapsto 0]$。

其中，变迁 t_i 的功能由 $t_i^{pre}t_i^{pos}$ 共同完成，以上三步共同完成一个变迁 $(tk,v,u) \rightarrow (tk',v',u')$。

2 时间自动机与模型检测

2.1 时间自动机

时间自动机[11]是对有限状态自动机增加一组时钟约束的一种扩充。所有时钟以相同的速度计时，它们记录从开始（或者被重置）以来所流失的时间单位。

设 C 为时钟变量集合，时钟约束 φ 可以定义为 $\varphi:=c \sim k | \varphi_1 \wedge \varphi_2$，其中 $c \in C, k$ 是有理数集合 Q 中的一个常量，$\sim \in \{<,\leqslant,==,\geqslant,>\}$，将 φ 的集合记为 $B(C)$。

定义 7（时间自动机）[20] 时间自动机可以定义为六元组 (L,l_0,C,A,I,E)，其中 L 是一个有穷的位置集合，$l_0 \in L$ 为初始位置，C 为有限的时钟变量集合，A 是有限的动作集合，I 是一个映射，它为 L 中的每个位置 l 指定 $B(C)$中的某些时钟约束，$E \subseteq L \times A \times 2^C \times B(C) \times L$ 是一个状态变换集合，一个变换$\langle l,a,g,\lambda,l' \rangle$表示一条从 l 到 l'的一条边，该边的动作标签为 a，g 是定义在 C 上的时钟约束，在转移发生时 g 必须被满足，λ 表示该转移发生时必须被重置的所有时钟集合。

以下用 $u:C \rightarrow R_{\geqslant 0}$ 表示对时钟变量 c 赋予非负实数，$\forall c \in C, u_0(c)=0$。

定义 8（时间自动机语义）[21] 时间自动机 $TA=(L,l_0,C,A,I,E)$是一个时间转换系统 $TTS_A=\langle S,s_0,\rightarrow \rangle$，其中 $S \subseteq L \times R_{\geqslant 0}^C$ 是状态集合，$s_0=(l_0,u_0)$是初始状态，$\rightarrow \subseteq S \times \{R_{\geqslant 0} \cup A\} \times S$ 由以下转移关系组成：

i. $(l,u) \xrightarrow{d} (l,u+d)$，如果 $\forall d': 0 \leqslant d' \leqslant d \Rightarrow u+d' \in I(l)$，其中 $d \in R_{\geqslant 0}$；

ii. $(l,u) \xrightarrow{a} (l',u')$，若存在 $e=(l,a,g,\lambda,l') \in E$，使得 $u \in g, u'=u[\lambda \mapsto 0], u' \in I(l')$。其中 $u[\lambda \mapsto 0]$表示将 u 中变量 $c \in \lambda$ 全部置 0。

复杂系统通常需要多个时间自动机的组合来描述，用多个自动机的乘积来描述组合时间自动机。给定 n 个自动机 $TA_i=(L_i,l_{i,0},C_i,A_i,I_i,E_i), 1 \leqslant i \leqslant n$，它们组成的自动机组合为 $TA=(L,l_0,C,A,I,E)$，其中 $L=L_1 \times L_2 \times \cdots \times L_n, l_0=l_{1,0} \times l_{2,0} \times \cdots \times l_{n,0}, C=C_1 \cup C_2 \cup \cdots \cup C_n, A=A_1 \cup A_2 \cup \cdots A_n, I(L)=\wedge I_i(L_i), E=E_1 \cup E_2 \cup \cdots E_n$。

定义 9（组合时间自动机语义）[21] 组合时间自动机可以用时间转换系统 $TTS_{CA}=(S,s_0,\rightarrow)$来描述，其中 $S=(L,u)$是状态集合，$s_0=(l_0,u_0)$为初始状态，变换 $\rightarrow \subseteq S \times S$ 由以下关系定义：

i. $(l,u) \xrightarrow{d} (l,u+d)$，如果 $\forall d': 0 \leqslant d' \leqslant d \Rightarrow u+d' \in I(l)$；

ii. $(l,u) \xrightarrow{a} (l'[l_i'/l_i], u')$，如果存在 $(l_i,a,g,\lambda,l_i') \in E$，使得 $u \in g, u'=u[\lambda \mapsto 0], u' \in I(l')$，其中 $l[l_i'/l_i]$表示向量 l 中元素 l_i 用 l_i'替换；

iii. $(l,u) \rightarrow (l'[l_j'/l_j, l_i'/l_i], u')$，如果存在 $l_i \xrightarrow{ch?g_i\lambda_i} l_i'$，并且 $l_j \xrightarrow{ch!g_j\lambda_j} l_j'$，使得 $u \in (g_i \wedge g_j), u'=u[\lambda_i \cup \lambda_j \rightarrow 0], u' \in I(l')$，其中 $ch!$ 表示发送同步信号，$ch?$ 表示接收同步信号。

为了满足复杂系统模型检测的需要，UPPAAL 中对 TA 进行了适当的扩展。其中紧急位置（Urgent Location）表示当系统位于此位置时，时间不允许流逝；紧急通道（Urgent Channel）表示当边的控制条件满足时，同步动作马上执行；控制标签（guard）可以是任何布尔表达式；边上的赋值标签使得系统具有数据运算能力。

2.2 模型检测

针对 TA 已经有成熟的模型检测工具，如 Kronos，UPPAAL 等。为了利用这些工具验证 TCPNIA 模型的正确性、安全性等，需要将 TCPNIA 转换成等价的 TA，以验证其是否满足用 CTL 或 TCTL 公式描述的需求规范。

UPPAAL 以 TA 和 CTL 作为输入，验证 TA 描述的功能模型是否满足 CTL 描述的需求规范。它包含可视化的编辑器、模拟器和验证器，支持一组带有整型变量、时钟和控制条件的 TA 运行，TA 之间可以通过共享变量与通道进行通信。它可以用来检测系统安全性、可达性和受限活性等，并且其建模语言提供了复杂运算的能力，因此能满足复杂系统的模型检测需要。

UPPAAL 中采用简化的 CTL 来描述系统属性。该 CTL 公式由路径算子（A-所有，E-存在）、时间算子（[]-未来全部，<>-未来某一时刻）和谓词公式组成。其中路径算子与时间算子必须成对出现组成路径公式。路径公式不允许嵌套。

对一个实时系统来说，一些属性必须在严格的时间范围内得到满足。有界活性(Bounded Liveness)是指性质能够在时间上限以内得到满足。TCTL 可以很好地描述这种性质。UPPAAL 通过观察自动机来实现对 TCTL 描述的有界活性检测。

3 TCPNIA 到 TA 的转换

3.1 转换算法

本算法在文献[15]中方法的基础上增加了冲突调解机制，该机制通过引入冲突矩阵和冲突判断函数，消除了转换得到的 TA 并发执行过程中可能出现的错误状态。具体算法描述如下：

i. 设置一个全局时钟 tc。

ii. 定义一个全局的紧急通道(urgent channel) go，该通道保证接收者在满足其它条件的情况下，接收到同步信号后立即进行状态转换；定义一个名为 Dummy 的自动机如图 2 所示。

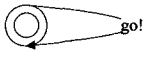

图 2 自动机 Dummy

iii. 定义一个结构

```
typedef struct {
    bool token;
    int v;
} P;
```

iv. 如果 TCPNIA 模型中有 m 个库所，则定义数组 P $p[m]$表示库所是否有 token，如果有 token，其值为 v，根据初始标记确定 $p[m]$的初始值；定义数组 clock $ts[m]$，当 token 更新时置为 0。

v. 如果 TCPNIA 模型中有 n 个变迁，则定义数组 int[0,1] conf[13][13]，表示变迁在激发过程中的冲突矩阵，1 表示冲突；定义 int[0,1] tag[13]，1 表示变迁处于激发过程中。

vi. 定义冲突判断函数 bool hasCollision(int j)。根据冲突矩阵，如果第 j 个变迁是某个正在执行变迁的冲突变迁，则返回值为真，表示第 j 个变迁此时不能执行；否则第 j 个变迁可以开始执行。

```
bool hasCollision(int j){
int i;
for(i=0;i<m;i++){
    if(conf[i][j]==1 and tag[i]==1)
        return true;
}
return false;}
```

vii. 划分变迁等价类：假设 TCPNIA 中每个变迁 t 有 a 个输入库所，b 个输出库所和 c 个抑制输入库所，每个变迁有一个变迁函数。我们使输入库所、输出库所和抑制输入库所个数分别相同，并将具有相同变迁函数的变迁划分为一个等价类。

viii. 针对每个变迁等价类定义一个 TA 模板：

(a) 定义模板的形式参数分别为：输入库所列表 $p_1^i, p_2^i, \cdots, p_a^i$，抑制弧列表 $p_1^{ih}, p_2^{ih}, \cdots, p_c^{ih}$，输出库所列表 $p_1^o, p_2^o, \cdots, p_b^o$，变迁最小执行时间 lb，变迁最大执行时间 ub，输出库所对应的时钟 $ts_1, ts_2, \cdots, ts_b$，变迁索引号 j；定义局部时钟变量 c 和变迁函数 $f_t(p_1^i.v, p_2^i.v, \cdots, p_a^i.v)$。

(b) 添加 disabled 和 firing 两个位置，初始位置为 disabled，在 firing 上添加不变量条件 $c \leqslant ub$。

(c) 添加一条从位置 disabled 到位置 firing 的边，其同步标签为 $go?$，控制条件为 $(\bigcap_{x=1}^{a} p_x^i.\text{token})$ and $(\bigcap_{y=1}^{c} p_y^{ih}.\text{token}==0)$ and $(\bigcap_{z=1}^{b} p_z^o.\text{token}==0)$ and not $hasCollision(j)$，赋值标签 $c=0, tag[j]=1, p_x^i.\text{token}=0$，其中 $x=1,2,\cdots,a$。

(d) 添加一条从位置 firing 到位置 disabled 的边，控制条件为 $c \geqslant lb$，赋值标签为 $p_z^o.\text{token}=1, p_z^o.v=f_t, ts_z=0, tag[j]=0$，其中 $z=1,2,\cdots,b$。

ix. 在过程声明部分，为 TCPNIA 中每个变迁实例化一个 TA 实例 $ta_k(k=0..n-1)$，如：$t0=Pt0(p[0], p[2], p[3], p[5], 1, 2, ts[3], ts[5], 0)$。

x. 定义 TCPNIA 模型对应为自动机 Dummy、$ta_k(k=0..n-1)$的组合。

图 3 显示了具有一个输入库所、一个抑制库所、一个输出库所和变迁函数为 $f_t(x)=x-1$ 的变迁等价类(a)转换为 TA(b)的情况。控制条件为 $pi.\text{token}$ and $pi.v>1$ and $pih.\text{token}==\text{false}$ and $po.\text{token}==\text{false}$ and not $hasCollision(j)$；紧急通道 go 保证了在满足控制条件的情况下，TA 一旦收到同步信号就将状态从 disabled 转换到 firing，在转换之前完成赋值操作 $pi.\text{token}=\text{false}, \text{tag}[j]=1$，也就是从输入库所中移去 token，并将变迁置于激发过程中。firing 中的不变量 $c \leqslant ub$ 保证 TA 位于 firing 状态，也就是变迁激发的时间不超过 ub。从 firing 到 disabled 的控制条件 $c \geqslant lb$ 保证 TA 位于 firing 状态的时间至少为 lb。这两个时钟限制使得变迁至少持续 lb 个时间单位，最多持续 ub 个时间单位，在此期间结束变迁。在变迁结束时刻，设置输出库所的 token 中的值 $po.\text{token}=1, po.v=ft(pi.v)$，让输出库所时钟开始计时 $ts=0$，解除变迁处于激发过程中的标志 $\text{tag}[j]=0$。TCPNIA 模型中库所 i 的 token 更新时间为 $ts_M[p_i]=tc-ts[i]$。

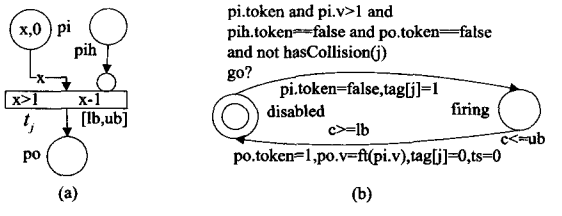

图 3 TCPNIA 中一个变迁等价类(a)到 TA(b)的转换

按上述过程构建的 TA 还不能进行有界活性的验证。为了验证有界活性，需要构建观察自动机。为了验证完成某一任务(TCPNIA 中一组变迁序列的激发)至少需要 lb 个时间单位，但是又不会超过 ub 个时间单位，按以下步骤构建观察自动机：

xi. 定义全局通道 chan arrive, leave。

xii. 定义观察自动机参数(const int lb, const int ub)和局部时钟 clock c。

xiii. 添加 begin, end 和 error 三个位置，初始位置为 begin；引一条从 begin 到 end 的边，同步标签为 $arrive?$，赋值标签为 $c=0$。分两种情况继续完成观察自动机的构建：

(a) 对于时间限制为$[lb, ub]$的情况，引一条从 end 到 begin 的边，控制条件为 $c \geqslant lb$ and $c \leqslant ub$，同步标签为 $leave?$；引

一条从 end 到 error 的边，控制条件为 $c>ub$；引一条从 end 到 error 的边，控制条件为 $c<lb$，同步标签为 *leave*?。如图 4(a)所示，其中模板参数为(const int *lb*，const int *ub*)。

(b) 对于至少需要 *lb* 个时间单位的情况，引一条从 end 到 begin 的边，控制条件为 $c\geqslant lb$，同步标签为 *leave*?；引一条从 end 到 error 的边，控制条件为 $c<lb$，同步标签为 *leave*?，如图 4(b)所示。

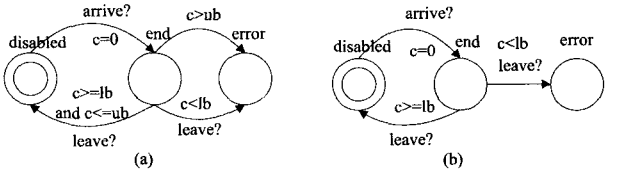

图 4　观察自动机

xiv. 为了与观察自动机进行通信，需要修改某一任务开始变迁的 TA 模板，如图 5(a)所示，在 disabled 与 firing 位置之间增加 nowait 位置，类型为 uegent；原先 disabled 到 firing 边更改为 disabled 到 nowait，标签上的信息不变；引一条从 nowait 到 firing 的边，同步标签为 *arrive*!。在任务结束变迁的 TA 模板中从 firing 到 disabled 的边上添加同步标签 *leave*!，如图 5(b)所示。

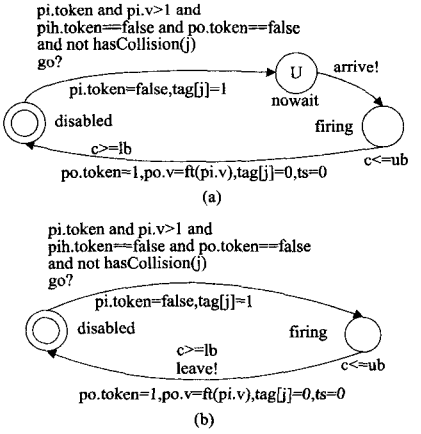

图 5　发送任务同步信息的 TA 模板

当特定任务开始时，发送 *arrive*! 同步信号，观察自动机接收到 *arrive*? 信号后立刻将局部时钟置 0，重新开始计时。当任务结束时发送 *leave*! 同步信号，观察自动机接收到 *leave*? 后，根据当前位置和控制条件来决定下一位置。模型检验时，如果观察自动机会进入 error 状态，说明系统不满足该性质的有界活性。

3.2　转换的正确性

TCPNIA 的变迁过程可以分为 3 个阶段(定义 6 中的 i，ii，iii)，TCPNIA 模型转换为 TA 后，刚好对应 TA 从 disabled→firing的变迁 a^{pre}、firing 中时间的流逝、firing→disabled 的变迁 a^{pos}。这里从每个执行步骤的系统前后状态等价性、时间流逝区间的等价性来证明 TCPNIA 到 TA 转换的正确性。

TCPNIA 的状态用 $S^N=(tk^N,v^N,tag^N,u^N)=(\{0,1\})^m\times R^m\times\{0,1\}^m\times(R_{\geqslant 0})^n$ 表示，初始状态为 $s_0^N=(tk_0^N,v_0^N,0,0)$。TA 状态用 $S^A=(L^A,u^A)$表示，初始状态为 $s_0^A=(l_0^A,u_0^A)$。设 TCPNIA 中变迁 t_i 对应一个时间自动机 TA_i。

定理 1　如果 TCPNIA 和相应的 TA 在初始状态下满足 $p_0[j].\text{token}=tk_0[j]$，$p_0[j].v=v_0[j]$，$j=1,2,\cdots,m$，那么 TCPNIA 经过变迁 t_i 得到的系统状态和 TA 经过与 t_i 对应的时间自动机 TA_i 运行后得到的状态相同，也就是 $p_k[j].\text{token}=tk_k[j]$，$p_k[j].v=v_k[j]$，$k=1,2,\cdots,m$。

证明：根据 TCPNIA 和 TA 的语义，一个变迁 t_i 及对应 TA 执行分两个阶段完成。

i. TCPNIA 中的变迁 t_i，如果有 $tk(^\circ t_i)==1\wedge tk(t_i^\circ)==0\wedge tk(t_i^\circ)==0\wedge\neg hasCollision(t_i)\wedge g_i(v)$，那么有 $(tk,v,tag,u)\xrightarrow{t_i^{pre}}(tk'',v,tag'',u'')$，执行完成后 $tk''(^\circ t_i)=0$，$tag''=tag[tag_i\mapsto 1]$，$u''=u[u_i\mapsto 0]$。

在条件 $tk(^\circ t_i)==1\wedge tk(t_i^\circ)==0\wedge tk(t_i^\circ)==0\wedge\neg$hasCollision$(t_i)\wedge g_i(v)$下，又由于在 a^{pre} 上没有 $u\in g$ 的条件，这些条件就满足了 3.1 节中步骤 viii(c)的控制条件，并且根据 3.1 节中步骤 viii(c)赋值标签定义有 $u'=u[u_i\mapsto 0]$，所以一定有 $u'\in I(l')$，从而有 $(l_i,a^{pre},g,u_i,l_i'')\in E$，使得 $(l,u)\xrightarrow{a^{pre}}(l''[l_i''/l_i],u'')$。又根据 3.1 节中步骤 viii(c)赋值标签定义有 $p''[^\circ t].\text{token}=0$，$tag''=tag[tag_i\mapsto 1]$。

在相同的状态下，TCPNIA 中 t_i 变迁第一阶段与相应的 TA_i 第一阶段运行产生相同的中间结果。此时，TCPNIA 中 t_i 变迁过程中，TA_i 处于 firing 状态；

ii. $\forall d'\in R_{\geqslant 0}$，$lb\leqslant d'\leqslant ub$，那么有 $(tk'',v,tag'',u''+d')\xrightarrow{t_i^{pos}}(tk',v',tag',u')$，从而得到 $tk'(t_i^\circ)=1$，$v'(t_i^\circ)=f_{t_i}(v(^\circ t_i))$，$tag'=tag''[tag''_i\mapsto 0]$。

根据 3.1 节定义 viii(d)，当 $lb\leqslant c\leqslant ub$ 时，满足 $u\in g$，从而有 $(l,u)\xrightarrow{a^{pos}}(l'[l_i'/l_i],u')$，根据其赋值定义有 $p'[t_i^\circ].\text{token}=1$，$p'[t_i^\circ].v=f_{t_i}(v(^\circ t_i))$，$tag_i'[j]=0$。

初始状态相同条件下，TCPNIA 中变迁 t_i 执行完成后与相应 TA_i 执行完成后得到相同的状态。

冲突矩阵避免了组合时间自动机的并发性错误，保证组合时间自动机中冲突自动机不会并发，从而使得 TCPNIA 中变迁的并发对应于相应 TA 的并发，所以在并发条件情况下，TCPNIA 的变迁与相应的 TA 执行在功能上等价。从而定理 1 成立。

定理 2　TCPNIA 中变迁 t_i 执行完成所需的时间区间与相应的 TA_i 从 disabled 到 firing，再到 disabled 所需总时间的区间相同。

证明：在 TCPNIA 中，对 $\forall d\in R_{\geqslant 0}$，$d\leqslant ub$ 有 $(tk'',v,tag'',u'')\xrightarrow{d}(tk'',v,tag'',u''+d)$，根据 3.1 中定义 viii(b)，$TA_i$ 中的 firing 位置满足不变量条件，即 $\forall d'$：$0\leqslant d'\leqslant d\Rightarrow u+d'\in I(l)$，使得 $(l,u)\xrightarrow{d}(l,u+d)$。两者均满足流逝时间小于等于 *ub*。它们的执行时间上限都是 *ub*。

另一方面，如果 $\forall d'\in R_{\geqslant 0}$，$lb\leqslant d'\leqslant ub$，那么 $(tk'',v,tag'',u''+d')\xrightarrow{t_i^{pos}}(tk',v',tag',u')$，同时根据 3.1 节节定义 viii(d)，当 $lb\leqslant c\leqslant ub$ 时，满足 $u\in g$，从而有 $(l,u)\xrightarrow{a^{pos}}(l'[l_i'/l_i],u')$。它们的执行时间下限都是 *lb*。

从而两者的执行时间区间均为$[lb,ub]$，定理 2 成立。

定理 1 和定理 2 说明，TCPNIA 到 TA 的转换算法保持了系统的功能语义和时间限制区间，从而，该转换算法是正确的。

3.3 算法复杂度分析

TCPNIA 模型到 TA 模型的转换算法包括划分等价类和每个等价类转换成 TA 时往输出文件中写数据。每个等价类转换为一个时间自动机所需的时间为常量，其时间复杂度为 $O(1)$。包含 n 个变迁的 TCPNIA 模型在最坏情况下有 n 个等价类，划分等价类的时间复杂度为 $O(n^2)$，n 个等价类到 TA 模型转换算法的时间复杂度为 $O(n\times1)$，此时整个 TCPNIA 模型到 TA 模型的转换算法时间的复杂度为 $O(n^2+n\times1)=O(n^2)$。最好情况下只有一个等价类，此时划分等价类的时间复杂度为 $O(n)$，整个 TCPNIA 模型到 TA 模型的转换算法的时间复杂度为 $O(n+1\times1)=O(n)$。

假设每个等价类有 m 个输入库所、k 个与抑制弧关联的库所和 l 个输出库所，那么该等价类需要 $m+k+l$ 个库所的存储空间。最坏情况下有 n 个等价类，此时算法的空间复杂度为 $O(n\times(m+k+l))$。最好情况下只有一个等价类，此时算法的空间复杂度为 $O(m+k+l)$。

4 实例

某火车站有一进站站台，每次只能停靠一列火车。进站之前有两条铁轨通向站台，分别通行快车和慢车。当有快车等待进站时，慢车必须等待。火车进入站台后一方面要进行上下客，另一方面要补充能源。假设一列火车能够存放 100 个单位的能量，只有当剩余能量小于 50 时才需要补充能量。系统需要记录自运行以来已经为进站火车总共补充的能源数量。系统的 TCPNIA 模型如图 6 所示。

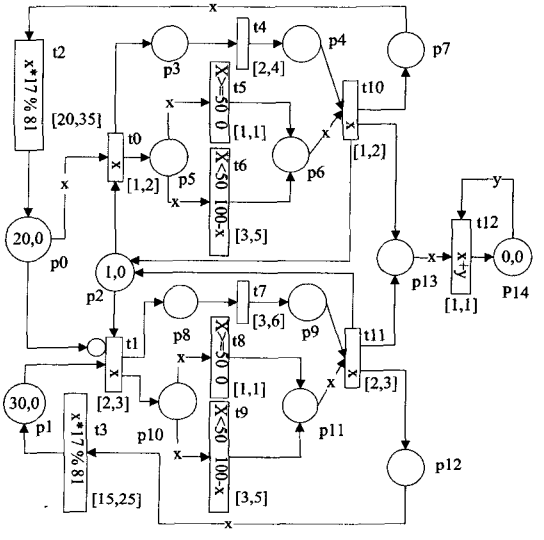

图 6 火车进站系统的 TCPNIA 模型

其中 p_0 表示快车正在等待进站，p_1 表示慢车正在等待进站，p_2 有 token 表示允许列车进站，p_{14} 表示自系统运行以来该站台已经为进站列车所添加的能源总量。t_0，t_1 分别表示快车和慢车进站，t_4，t_7 表示旅客上下车，t_5，t_8 表示当剩余能量大于等于 50 时不需要添加能量，t_6，t_9 表示当剩余能量小于 50 时需要添加能量，t_{10}，t_{11} 表示火车出站，t_{12} 表示对添加的能源总量进行累计，t_2，t_3 分别表示一列快车和一列慢车的到来。

该系统需要进行安全性、可达性、系统活性和有界活性验证。为了使火车安全进站，需要满足：i) 如果有列车正停靠在站台上，就不能允许其他列车进站，UPPAAL 中表达为：($p[3]$.token+$p[4]$.token)-->$p[2]$.token==false，($p[8]$.token+$p[9]$.token)-->$p[2]$.token==false；ii) 不能有两列火车同时停靠站台：($p[3]$.token+$p[4]$.token)-->($p[8]$.token+$p[9]$.token==0)，($p[8]$.token+$p[9]$.token)-->($p[3]$.token+$p[4]$.token==0)。可达性指火车的到来总是能在未来某一时刻离开：$p[0]$.token-->$p[7]$.token，$p[1]$.token-->$p[12]$.token。系统活性指系统的运行过程中不会出现死锁：A[]not deadlock。有界活性：一列快车到来时可能正好有一列慢车进入站台，其需要等待慢车离开后才能进站，这列快车最多需要 21 个时间单位就能离开站台；当快车到来时，刚好站台有空，直接进站离开至少需要 4 个时间单位。由于 UPPAAL 不能直接对 TCTL 进行验证，因此需要构建观察自动机来实现。

将变迁划分为 $\{t_0\}$，$\{t_1\}$，$\{t_2,t_3\}$，$\{t_4,t_7\}$，$\{t_5,t_8\}$，$\{t_6,t_9\}$，$\{t_{10},t_{11}\}$，$\{t_{12}\}$ 8 个等价类；为每个等价类建立一个自动机模板、一个 Dummy 模板和一个观察自动机模板。其中观察自动机模板如图 4(a)所示。快车进站并发送同步信号的自动机如图 7(a)所示；列车离开并发送同步信号的自动机如图 7(b)所示。有界活性就是验证任何路径、任何时刻观察自动机不进入 error 状态。

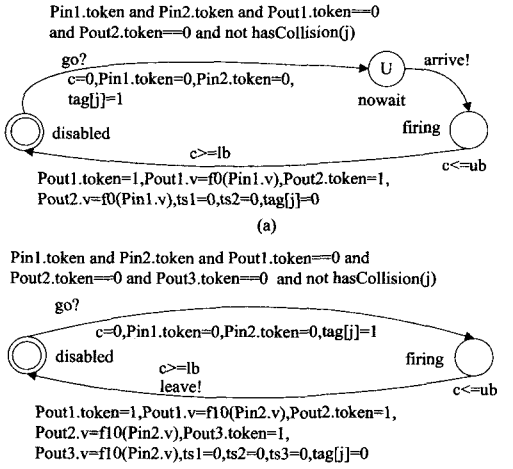

图 7 被观察的自动机

利用 UPPAAL 进行模型检测，结果表明上述安全性、可达性、系统活性和有界活性表达式在该组合自动机模型下为真。由于我们已经证明 TCPNIA 到 TA 的语义转换是正确的，因此这些属性在图 6 的 TCPNIA 模型下也为真。

大型复杂系统需要层次化方法来降低模型复杂性，提高重用性。层次化的 TCPNIA 模型是元组 $HTCPNIA=\langle P,T,S,In,Out,I,G,F,D,TS,M_0\rangle$，其中 $S=\{s_1,s_2,\cdots,s_u\}$ 是有限的抽象变迁集合，$In\subseteq P\times(S\cup T)$ 是一组有限的输入弧集，$Out\subseteq(S\cup T)\times P$ 是一组有限的输出弧集，库所容量与弧的权均为 1，在定义中省略，其他符号的定义与 TCPNIA 模型相同。TCPNIA 模型是 HTCPNIA 模型在 $S=\emptyset$ 时的特例。

这里采用文献[16]中的方法，在模型等价的基础上利用简单模型或已经验证的模型来替换复杂的等价模型，以达到提高验证效率的目的。

火车进站系统的层次化模型如图 8 所示。其中图 8(a)表示系统的顶层模型，s_0 和 s_1 分别表示快车与慢车进站并停靠站台的抽象过程。图 8(b)表示慢车进站并停靠站台 s_1 的

细化模型。这里假设 s_0 和 s_1 所对应的细化模型已经经过正确性验证，我们只需验证图 8(a)中的模型。

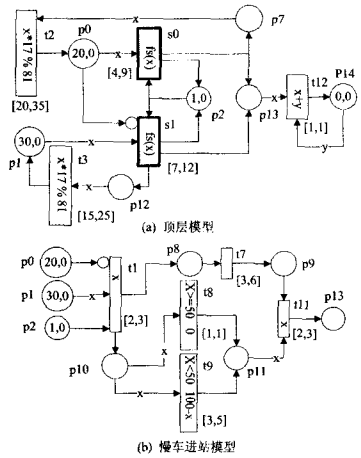

图 8　层次化模型

图 6 中的模型与图 8(a)中模型部分属性检测效率分别如表 1 和表 2 所列。其中运行环境为 Intel(R) Core(TM)2 Duo CPU，1.79G 内存，Windows XP SP3 操作系统，UPPAAL 4.1.1 模型检测软件。

表 1　非层次化验证结果

验证的属性	耗时(s)	常驻内存(kB)	虚拟内存(kB)
无死锁	62.266	237,716	484,236
快车总是会进站并离开	136.907	255,172	518,632
慢车总是会进站并离开	50.016	255,172	518,632

表 2　层次化验证结果

验证的属性	耗时(s)	常驻内存(kB)	虚拟内存(kB)
无死锁	2.188	20,704	47,016
快车总是会进站并离开	2.015	20,556	46,712
慢车总是会进站并离开	2.25	20,704	47,016

从表 1 和表 2 的比较来看，就时间与空间效率而言，层次化验证的效率比非层次化验证的效率高，并且这种方法提高了重用性。

结束语　为了对 TCPNIA 模型进行检测，本文给出了一个结构化的从 TCPNIA 到 TA 的语义等价转换算法。该算法引入了冲突调解机制来消除由于变迁时延可能引起的时间自动机并发性错误，从而导致的错误状态。文章给出了算法的语义正确性证明和复杂性分析，通过实例说明了该方法的可行性。文章还给出了 TCPNIA 的层次化模型。该实例利用已有层次化验证方法来提高验证效率，通过与非层次化验证结果的比较来说明层次化验证带来的时间与空间效率的提高。

下一步需要开发一个图形化的 TCPNIA 建模工具，该工具能够利用本文的算法，直接将 TCPNIA 模型转换成相应的 TA，然后利用 UPPAAL 工具进行模型检测。另一方面需要进一步研究 TCPNIA 模型在语义等价下的抽象与精化方法，以增强模型的层次化、模块化建模与验证能力。

参 考 文 献

[1] Murata T. Petri nets: Properties, analysis and applications[J]. Proceedings of the IEEE, 1989, 77(4): 541-580

[2] Cortés L A, Eles P, Peng Z. Modeling and formal verification of embedded systems based on a Petri net representation[J]. Journal of Systems Architecture, 2003, 49(12-15): 571-598

[3] Varea M, Al-Hashimi B M, Cortés L A, et al. Dual flow nets: Modeling the control/data-flow relation in embedded systems [J]. ACM Transactions on Embedded Computing Systems, 2006, 5(1): 54-81

[4] Liu S, Mu C. An extended Petri net epres for embedded system modeling[A] // Proceedings of the Fifth IEEE International Symposium on Embedded Computing[C]. Beijing, China: IEEE Computer Society, 2008: 9-13

[5] 杨年华，虞慧群. 基于带抑制弧的时延着色 petri 网的嵌入式系统建模与验证[J]. 华东理工大学学报，2010，36(3)：97-103

[6] Clarke E M, Emerson E A, Sistla A P. Automatic verification of finite-state concurrent systems using temporal logic specifications[J]. ACM Transactions on Programming Languages and Systems(TOPLAS), 1986, 8(2): 244-263

[7] 王小兵，段振华. 时序逻辑程序的模型检测[J]. 计算机科学，2009，36(10)：164-167

[8] Alur R, Courcoubetis C, Dill D. Model-checking for real-time systems[A] // Proceedings of Fifth Annual IEEE Symposium on Logic in Computer Science(LICS)[C]. Philadelphia, PA, USA: IEEE Computer Society, 1990: 414-425

[9] UPPAAL. http://www.uppaal.com/. 2009

[10] KRONOS. http://www-verimag.imag.fr/TEMPORISE/kronos/. 2002

[11] Alur R, Dill D L. A theory of timed automata[J]. Theoretical Computer Science, 1994, 126(2): 183-235

[12] Cortés L A, Eles P, Peng Z. Verification of embedded systems using a Petri net based representation[A] // Proceedings of the 13th international symposium on System synthesis[C]. Madrid, Spain: IEEE Computer Society, 2000: 149-155

[13] Gu Z, Shin K G. Analysis of event-driven real-time systems with time Petri nets: A translation-based approach[A] // Proceedings of the IFIP 17th World Computer Congress-TC10 Stream on Distributed and Parallel Embedded Systems: Design and Analysis of Distributed Embedded Systems[C]. Deventer, The Netherlands: Kluwer, B. V., 2002: 31-40

[14] Cassez F, Roux O-H. Structural translation from time Petri nets to timed automata[J]. Electronic Notes in Theoretical Computer Science, 2005, 128(6): 145-160

[15] Gu Z, Shin K G. An integrated approach to modeling and analysis of embedded real-time systems based on timed Petri nets[A] // Proceedings of 23rd International Conference on Distributed Computing Systems[C]. Providence, Rhode Island, USA: IEEE Computer Society, 2003: 350-359

[16] Cortés L A, Eles P, Peng Z. Hierarchical modeling and verification of embedded systems[A] // Proceedings of the Euromicro Symposium on Digital Systems Design[C]. Warsaw, Poland: IEEE Computer Society, 2001

[17] Zheng H, Mercer E, Myers C. Modular verification of timed circuits using automatic abstraction[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2003, 22(9): 1138-1153

[18] 夏传良，焦莉，陆维明. Petri 网精细化操作及其在系统设计中的应用[J]. 软件学报，2006，17(1)：11-19

(下转第 209 页)

法分析中可能存在扫描不能启动和/或扫描不完全的现象。

针对第二种现象，我们递归省略输入句子的当前未覆盖首终结符，直到能够输出一棵分析子树为止。我们以(→·S,[i,i]),0<i<l,为参数，调用子程序 Back-trace；如果没有得到分析子树，则令 i=i+1，并继续调用 Back-trace(→·S,[i,i])。

2.3 补充未覆盖子树

以上两种方法能够有效地解决 Earley 句法分析的空树问题，但只能为输入句子输出部分子树，不能够覆盖整个句子，因此补充未覆盖子树可以进一步提高句法分析的整体性能。

对于第一种情况下省略的句子结尾的子序列，如果该子序列长度等于一，则将其作为已输出子树的右兄弟节点连接到当前根节点；如果该子序列的长度大于等于二，我们可以再次调用已经扩展了的概率 Earley 句法分析器，得到未覆盖子树，并作为已输出子树的右兄弟子树连接到当前根节点；如果反复调用 Earley 句法分析器仍不能得到任何子树，则将省略子序列按原有顺序作为已输出子树的右兄弟节点逐一连接到当前根节点。

对于第二种情况下省略的句子首部的子序列，在理论上来说是无法构造任何子树的，因为当前文法不包含任何以该子序列为右部左角的产生式规则。然而，根据我们对具体语料的观察分析，发现造成这种情况的大部分原因是句首存在两个以上的 wLB 标记，这种标记的单词一般为左括号(、【、「、〖、〔,左引号‘、“,左书名号《、〈等等。如果输入句子没有语法错误，则当前已输出子树的右侧也必然省略了相应的 wRB 标记。

在句法树库中，大部分这种情况的外层节点都与内部子树的顶层节点相同，因此我们在句法分析时就以同时进行左右扩展的方法补充这种未覆盖子树，并标记完整句法树的根节点为已输出子树的根节点。

3 句法分析实验

基于以上理论，我们在中文信息学会句法评测 CIPS-ParsEval-2009 的实验语料上进行了一系列句法分析实验。

3.1 语料与任务

此次评测句法分析任务的语料由清华大学模式识别国家重点实验室自然语言处理组改编自清华树库 TCT，该句法树库的标记体系共包括 70 个词性标记和 16 个句法标记[4]。

用于评测的数据集合共包括约 45 万词汇，约 4 万句子，其中新闻类占 36%、百科学术类占 56%、文学类占 8%。以上数据分割为训练集和测试集，训练集合大约包括 35 万词汇，3.3 万句法树，句子平均长度为 11 个单词；测试集合大约包括 10 万词汇，7000 汉语句子，句子平均长度 12 个单词(此处没有计算那些只包括一个子节点的平凡句法树)。

评测任务的输入是经过正确切分和词性标注处理的汉语句子；输出是相应句子的完整句法结构树，各个节点以 np，vp，ap 等句法标记标注；评测指标基于括号匹配，包括标记准确率(Precision)、标记召回率(Recall)和 F1 值。

3.2 实验结果与分析

在句法分析实验中，我们直接从评测语料训练集中抽取文法规则，去除那些只出现一次的规则，未做任何概率平滑。实验包括：原始 Viterbi 路径句法分析，扩展启始状态，省略未覆盖句首和补充未覆盖子树。原始 Viterbi 路径句法分析在测试集合上共产生 684 棵空树，接近整个测试集的 10%，扩展启始状态的方法去除了大约 91%的空树，省略未覆盖句首的方法又去除了 8%，剩余的空树大多是由测试数据的文法错误造成。各部分实验的句法分析性能如表 1 所列。

表 1 句法分析实验结果

括号匹配(%)	Precision	Recall	F1
原始 Viterbi 路径	74.35	73.21	73.78
扩展启始状态	77.06	75.78	76.41
省略未覆盖句首	77.17	76.20	76.68
补充未覆盖子树	77.95	77.47	77.71

扩展启始状态的方法使得句法分析的 F1 值提高了 2.63%，是最有效的 Viterbi 扩展；省略未覆盖句首的方法使得 F1 值提高了 0.27%；补充未覆盖子树的方法使得 F1 值提高了 1.03%。总体上来看，我们对原始 Viterbi 路径的扩展使得概率 Earley 句法分析的整体性能提高了 3.93%。

结束语 本文针对概率 Earley 算法在句法分析中产生的空树问题，提出了扩展 Viterbi 路径的改进方法，包括扩展启始状态、省略未覆盖句首和补充未覆盖子树。在清华树库 TCT CIPS-ParsEval-2009 句法评测任务上的实验结果表明：扩展的 Viterbi 路径排除了绝大多数空树问题，有效地提高了 Earley 句法分析的整体性能。此后的研究工作将主要侧重于如何为已构造子树和未覆盖成分选择最优根节点。

参 考 文 献

[1] Han Xi-wu, Zhao Tie-jun, et al. Cross-lingual Syntactic Subcategorization Analysis Based on Chinese and English Sentence Pairs [C] // Proceedings of the First International Conference on Global Interoperability for Language Resources. 2008: 97-104

[2] Liu Yang, Liu Qun, Lin Shou-xun. Tree-to-String Alignment Template for Statistical Machine Translation [C] // Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL. 2006: 609-616

[3] Stolcke A. Parsing Algorithm that Computes Prefix Probabilities [J]. Computational Linguistics, 1995, 21(2): 1-36

[4] Zhou Qiang. Build a Large-Scale Syntactically Annotated Chinese Corpus [C] // Proceedings of 6th International Conference of Text, Speech and Dialogue (TSD2003). 2003: 106-113

[5] Earley J. *An efficient context-free parsing algorithm* [J]. Communications of the Association for Computing Machinery, 1970, 13(2): 94-102

(上接第 176 页)

[19] Henzinger T A, Manna Z, Pnueli A. Timed transition systems [A] // Real-time: Theory in practice [C]. Volume 600 of Lecture Notes in Computer Science. Berlin/Heidelberg: Springer, 1992: 226-251

[20] Alur R. Timed automata [A] // Computer aided verification [C]. Volume 1633 of Lecture Notes in Computer Science. Berlin/Heidelberg: Springer, 1999: 8-22

[21] Behrmann G, David A, Larsen K G. A tutorial on UPPAAL [A] // Formal methods for the design of real-time systems [C]. Volume 3185 of Lecture Notes in Computer Science. Berlin/Heidelberg: Springer, 2004: 200-236