并发程序 时序分析 安全性

计算机科学1998 Vol. 25 №. 1

13-17

并发程序性质的时序分析

Temporal Analysis of Properties of Concurrent Programs

贾国平 孟旭东

(上海贝尔电话公司 R&D 上海201206)

郑国梁

(南京大学计算机科学系 南京210093)

TP31

摘 要 In this paper, various properties of concurrent programs are analyzed and discussed. In particularly, two kinds of primary important properties, safety properties and liveness properties are considered. Formal definitions for safety and liveness are firstly proposed. Then a topological characterization of these properties is given. For properties that are expressible by temporal logic, a syntactic characterization of temporal formulas that correspond to properties in the different classes is finally provided. These conclusions are very helpful in specification and verification of concurrent programs and it provides a base for the application of temporal logic.

关键词 Concurrent programs, Safety property, Liveness property, Fairness property, Specification, Verification, Temporal logic.

1 引言

不同于顺序程序,并发程序并不能完全由简单的输入/输出关系来说明。并发程序的一个执行行为可以视为一个无穷状态序列 $\sigma: s_0, s_1, s_2, \cdots$,其中状态 s_0 之后的每一状态都是通过在前一状态下执行程序中的一个原子转换(或活动)而得到的。一个性质即是这样的一个无穷状态序列子集。由于一个程序也定义了一个状态序列集合,因此可以定义一个程序 P 满足一性质 p,如果由程序 P 所定义的状态序列集合包含于性质 p 集合之中。

从顺序程序推理到并发程序推理需要作许多扩充。首先,由于并发程序相互间的交互及干扰,必须对它们的交互作某一原子性假设[1]。其次,对并发程序中感兴趣的性质类比顺序程序中更加复杂。顺序程序中的不变式性及终止性不再足以用于并发程序,并发程序中出现了许多特有的性质,如互斥性、无死锁性、响应性及公平性等等。

Lamport 首先考虑了并发程序的安全性-活性分类[2],这一分类最大的优点在于,每一性质类中都包含了相同特征的性质。安全性表示了系统应该连续保持的需求,除了包含顺序程序中的部分正确性外,还包含并发程序中的互斥性及无死锁性等。活性表示了系统不必一直保持但系统一定要保证最终有的需求,除了包含顺序程序中的终止性外,还包含并发程序中诸如无活锁性、保证服务性、响应性等有趣的性质。这一分类的另一个优点在于,两类性质的正确性证明通常须使用不同的证明技术,对这两类性质进行分类有助于选择最合适的证明方法。

由于对时序逻辑的广泛兴趣及目前时序逻辑在并发程序规范及验证中的广泛应用,本文对并发程序性质进行时序分析。为了下面讨论方便起见,假设执行均为无穷状态序列,对有穷状态序列 $\sigma: s_0, s_1, \cdots, s_n$,定义 $s_m = s_n$(对所有 $m > n$),也即通过重复最后一状态来得到终止执行序列的无穷表示。

2 程序性质定义

首先给出一些定义、记号及约定,然后给出各种不同性质类的形式定义。

对集合 S,记 S^* 为所有 S 上的有穷序列集合,S^ω 为所有 S 上的无穷序列集合。特别对每一元素 $t \in S$,记 t^* 及 t^ω 分别表示由单元素 t 构成的有穷和无穷序列。一个性质是集合 S^ω 的一个子集。对序列 σ,$|\sigma|$ 表示序列的长度。如果 σ 为有穷序列 $s_0, s_1, \cdots, s_k$,

贾国平 博士,主要研究领域为软件工程、形式化方法、智能网。孟旭东 现为南京邮电学院讲师,主要研究领域为软件工程、计算机与通信、CSCW。郑国梁 教授,博士生导师,主要研究领域为软件工程、软件开发环境。

则$|\sigma|=k+1$，否则$|\sigma|=\omega$。对每一序列$\sigma:s_0,s_1,\cdots$及一整数$k<|\sigma|$，$\sigma|_k$表示序列σ的有穷序列$s_0,s_1,\cdots,s_{k-1}$，即σ的前k个元素构成的有穷前缀。$\sigma_1\cdot\sigma_2$为序列σ_1和序列σ_2的合并运算。如果σ_1为有穷序列，则$\sigma_1\cdot\sigma_2$为σ_1后续σ_2而成的序列，否则$\sigma_1\cdot\sigma_2=\sigma_1$。

如果取集合S为程序状态集合，则程序的每一执行都可模型化为集合S^ω中的一元素。因此，下面也称S^ω中元素为执行。

下面给出程序性质的形式定义。

2.1 安全性(Safety)

安全性最初在文[2]中给出，文[3]及[4]中给出了其形式定义。非形式地，安全性说明某一"坏事"在执行中永不会发生。安全性的例子包括部分正确性，无死锁性及互斥性等等。现有文献中已出现了许多不同的安全概念。下面给出它们的形式定义。

定义2.1 一个性质p是安全性，是指它满足：如果对序列$\sigma\in S^\omega$及任意$n\geqslant 0$，总存在序列$\beta\in S^\omega$，使得$(\alpha|_n)\cdot\beta\in p$，则$\sigma\in p$成立。

直观上，安全性是一性质p，如果一无穷序列σ的任意有穷前缀均可扩充为p中的序列，则序列σ也属于p。由此可知，安全性定义正好对应于文[5]中的极限封闭性质。

另一种等价地定义安全性的方式为：一个性质p为完全性，如果p满足：对任意$\sigma\in S^\omega$，$\sigma\notin p$，则对任意序列$\beta\in S^\omega$，存在整数i，使得$(\sigma|_i)\cdot\beta\notin p$。

由此等价定义知道，如果p是安全性，则对每一不属于p的序列σ，总存在某一有穷状态位置，在此位置"坏事"一定出现。因此安全性是有穷可反驳的。

需要注意的是，安全性并不要求某事最终一定发生，它仅仅要求无条件地阻止"坏事"的出现。一旦"坏事"发生，则一定存在一个位置，在此位置处，可判别它的出现。

定义2.2 一个性质p是哑元封闭的，是指如果序列$\sigma:s_0,s_1,\cdots,s_i,s_{i+1},\cdots$属于p，则序列$\sigma':s_0,s_1,\cdots,s_i,s_i,s_{i+1}\cdots$也属于p。

由此定义可知，如果p是哑元封闭性，则对每一序列$\sigma\in p$，连续重复σ中的任意状态有穷多次所得序列仍属于p。下面给出哑元封闭安全性定义。

定义2.3 一个性质p是哑元封闭安全性当且仅当下述条件成立：对任意序列$\sigma\in S^\omega$，$\sigma\in p$当且仅当对任意整数$k\geqslant 0$，$(\sigma|_k)\cdot S_k^\omega\in p$，其中$S_k$为$\sigma$中第$k+1$个状态，$S_k^\omega$为由单个状态$S_k$组成的无穷序列。

上述哑元封闭安全性最初由Lamport给出，因此也称为L-安全性。显然，定义2.1中的安全性要比L-安全性更一般。一个性质是L-安全性，则它是安全性。对一哑元封闭性质，两种安全性是等价的，即性质p是L-安全性当且仅当p是安全性。

定义2.4 一个性质p是强安全性当且仅当p满足：(1)p是哑元封闭安全性；(2)对任一p中序列$\sigma:s_0,s_1,\cdots,s_{k-1},s_k,s_{k+1},\cdots$及任一整数$k>0$，序列$s_0,s_1,\cdots,s_{k-1},s_{k+1},\cdots$仍属于p，即对p中序列$\sigma$，除$\sigma$的初始状态$s_0$外，消去$\sigma$中任意状态所得到的序列仍属于p。

直观上，强安全性说明，如果在某一时间没有对系统进行观察，则其后观察得到的行为仍然满足性质。显然，强安全性是最特殊的安全性。

2.2 活性(Liveness)

活性在许多文献中也称为最终性(eventuality)或进展性(progress)，最初在文献[2]中给出，其形式定义在文[4]中给出。非形式上，一个活性说明"好事"在执行过程中一定发生。活性的典型例子包括终止性、无活锁性、保证服务性等。

活性的一个明显特征即为任何一有穷序列都可以进行修正，即它总可能在将来满足"好事"出现这一要求。这一特征显然是区别于安全性的本质特征。如果一有穷序列是不能够进行修正的，则一定是"坏事"出现了。活性不能描述"坏事"的出现，它只能描述"好事"的发生。下面给出各种活性的定义。

定义2.5 性质p是一活性当且仅当满足对任意序列$\sigma\in S^*$，存在序列$\beta\in S^\omega$，使得$\alpha\cdot\beta\in p$成立。

直观上，p为活性，则每一有穷序列均可扩充为p中的一无穷序列。由上述定义可以看出，活性并不要求"好事"总是发生，它只说明"好事"最终发生。

定义2.6 性质p是一致活性当且仅当存在序列$\alpha\in S^\omega$，使得任何$\beta\in S^*$均有$\beta\cdot\alpha\in p$成立。

由定义可知，p是一致活性当且仅当有一无穷序列(α)，它附加到任一有穷序列(β)之后得到的序列都属于p。

定义2.7 性质p是绝对活性当且仅当p非空且对任意$\alpha\in p$及任意$\beta\in S^*$，均有$\beta\cdot\alpha\in p$成立。

直观上，p是绝对活性当且仅当p非空且p中每一序列(α)加到任一有穷序列(β)之后所得序列仍属于p。

由上述三个活性定义知道，任何绝对活性是一致活性；任何一致活性是活性，即绝对活性$\subseteq$一致活性$\subseteq$活性，且这一包含是严格的。

2.3 公平性

另一类重要的活性：公平性，在并发程序性质的

验证中，特别是活性证明中起着至关重要的作用。此处，只给出其定义，有关详细讨论可参见文[6]。

定义2.8 性质 p 是稳定性，如果对 p 中每一序列 σ，其所有后缀也属于 p，即 p 是后缀封闭的。

对稳定性 p，有下述结论。

命题2.1 对一性质 $p\neq S^{\omega}$，p 是稳定性当且仅当 p 的补集是一绝对活性。

证明：设 p 是稳定性且 $p\neq S^{\omega}$，令 $q=S^{\omega}-p$，则 q 非空。对 $\sigma\in q$ 及任一 $\beta\in S^{*}$，如果 $\beta\cdot\sigma\in p$，则由稳定性定义，$\sigma\in p$。显然与 $\sigma\in q$ 矛盾。因此对任意 $\beta\in S^{*}$，$\beta\cdot\sigma\in q$，由定义2.7知 q 是绝对活性的。

反之，$q=S^{\omega}-p$ 且 q 是绝对活性。令 $\sigma\in p$，σ' 为 σ 的任一后缀。如果 $\sigma'\in q$，则由绝对活性定义知，$\sigma\in q$。显然与 $\sigma\in p$ 矛盾。因此 $\sigma'\in p$，即 p 是后缀封闭的。得证。

定义2.9 性质 p 是公平性当且仅当 p 是稳定性且 p 是绝对活性。

直观上，性质 p 是公平性，如果对 p 中序列 σ 增加或删除任一前缀所得序列仍属于 p，即公平性对前缀的增加及删除运算是封闭的。

3 程序性质的拓扑分析

本节从拓扑结构上对不同性质进行分析，给出不同性质的拓扑特征。

3.1 拓扑结构

下面给出一些拓扑概念。首先给出 S^{ω} 中距离的概念。

定义3.1 对任何两个无穷序列 $\sigma,\sigma'\in S^{\omega}$，距离 $distance(\sigma,\sigma')=|\sigma-\sigma'|$ 定义如下：如果 σ 与 σ' 相同，即 $\sigma=\sigma'$，则 $|\sigma-\sigma'|=0$，否则 $|\sigma-\sigma'|=2^{-j}$，其中 $j=\max\{j|\sigma|_j=\sigma'|_j\}$，即 j 为 σ 与 σ' 最大相同前缀的长度。

不难证明，上述所定义的距离 distance 满足度量所要求的所有性质，因此 $(S^{\omega},distance(\sigma,\sigma'))$ 构成一度量空间。

定义3.2 S^{ω} 中一无穷序列 $\sigma_0,\sigma_1,\sigma_2\cdots$ 收敛于 σ，如果 $\lim_{i\to\infty} distance(\sigma,\sigma_i)=0$。$\sigma$ 称为无穷序列 $\sigma_0,\sigma_1,\sigma_2,\cdots$ 的极限，记为 $\sigma=\lim_{i\to\infty}\sigma_i$。

定义3.3 给定一集合 $A\subseteq S^{\omega}$，称 $\sigma\in S^{\omega}$ 为集合 A 的极限点，如果存在 A 中元素构成的一个无穷序列 $\sigma_0,\sigma_1,\sigma_2\cdots$ 收敛于 σ。

显然，由上述定义可知，集合 A 中每一元素 $\sigma\in A$ 均为集合 A 的极限点。

定义3.4 集合 A 的所有极限点集合称为集合 A 的闭包。记集合 A 的闭包为 Closure(A)。

显然，由定义，有 $A\subseteq Closure(A)$。

定义3.5 一个集合 A 是闭集，如果集合 A 由它所有的极限点构成，即 $Closure(A)\subseteq A$。

由上述定义可知，集合 A 为闭集，则 $A=Closure(A)$ 成立。集合 A 的闭包 Closure(A) 为包含集 A 的最小闭集。

定义3.6 一个集合 A 是开集，如果集合 A 中每一元素 $\sigma\in A$，存在 $\varepsilon>0$，对每一 $\sigma'\in S^{\omega}$，满足 $distance(\sigma,\sigma')<\varepsilon$，均有 $\sigma'\in A$。

定义3.7 一个集合 A 是稠密集，如果对任意一 $\sigma\in S^{\omega}$ 及 $\varepsilon>0$，总存在 A 中元素 $\sigma'\in A$，满足 $distance(\sigma,\sigma')<\varepsilon$。

3.2 安全性

命题3.1 p 是一安全性当且仅当 p 是一闭集。

证明：设 p 是一安全性。为了证明 p 是一闭集，只需证明 $p=Closure(p)$。因为 $p\subseteq Closure(p)$，因此只需证明 $Closure(p)\subseteq p$。

对任意 $\sigma\in Closure(p)$，总存在一 p 中的无穷序列 $\sigma_0,\sigma_1,\sigma_2\cdots$ 收敛于 σ。假设 $\sigma\notin p$。由于 p 是安全性，由安全性定义知，存在一整数 k，有穷前缀 $\sigma|_k$ 的任何扩充均不属于 p。又由距离定义可知，不存在 $\sigma'\in p$，使得 $distance(\sigma,\sigma')<2^{-k}$ 成立。特别，对任意 j，$distance(\sigma,\sigma_j)\geqslant 2^{-k}$。显然，这与 σ 是序列 $\sigma_0,\sigma_1,\sigma_2\cdots$ 的极限矛盾。因此，$\sigma\in p$，即 p 为闭集。

反之，设 p 为闭集且 p 不为安全性。由安全性定义可知，对一序列 $\sigma\in S^{\omega}$ 及任意整数 k，存在一 $\rho\in S^{\omega}$，使得 $(\sigma|_k)\cdot\rho\in p$，但 $\sigma\notin p$。下面可以构造一 p 中无穷序列 $\sigma_0,\sigma_1,\sigma_2,\cdots$，其中，每一 σ_i 就取作 $(\sigma|_k)\cdot\rho\in p$。由此构造，显然 $\lim_{i\to\infty}\sigma_i=\sigma$。由于 p 是闭集，由定义 $\sigma\in p$。矛盾。因此，p 为安全性。得证。

由于闭集对于有穷并及任意交运算均是封闭的，因此由命题3.1知道，安全性的有穷并集和安全性的任意交集仍是安全性。

由前节稳定性定义及本节命题3.1立即有下述结论：

命题3.2 p 是一稳定性，则 p 是开集。

3.3 活性

命题3.3 p 是一活性当且仅当 p 是稠密集。

证明：设 p 是一活性，对任一序列 $\sigma\in S^{\omega}$ 及 $\varepsilon>0$，由活性定义，总存在 $\rho\in p$ 及一整数 k，满足 $\rho|_k=\sigma$

$|_k$ 且$2^{-k}<\epsilon$。因此由距离定义，有 distance$(\sigma,\rho)<\epsilon$。p 为稠密集。

反之，设 p 为稠密集且 p 不是活性，则存在有穷序列前缀 σ'，它不是 p 中任何序列的前缀，即对 σ' 的任何扩充 σ，都不会存在 $\sigma''\in$p，满足 distance$(\sigma,\sigma'')<2^{-k}$，其中 $k=|\sigma'|$。取此 σ 及 $\epsilon=2^{-|\sigma'|}$，则对任何 $\rho\in$p，均有 distance$(\sigma,\rho)\geqslant\epsilon$ 成立。由此知，p 不是稠密集。矛盾。得证。

由于活性是稠密集，因此活性在并集运算下是封闭的，即活性与活性的并仍然是活性。

3.4 其它性质

有许多其它的性质既不是安全性，也不是活性，如大家熟知的程序的完全正确性。然而，完全正确性有一个特点，即它可以写为部分正确性和终止性的合取，其中部分正确性是一安全性，终止性是一活性。文[4]中给出如下一个更一般的结论：

命题3.4 每一性质 P 均是一安全性 S 和一活性 L 的交集，即 $P=S\cap L$。

证明：令 S=Closure(P)，即 S 为包含性质 P 的最小安全性。令 $L=\sim(S-P)$，即 L 为差集 S−P 的补集。由于 $L\cap S=\sim(S-P)\cap S=(\sim S\cup P)\cap S=(\sim S\cap S)\cup(P\cap S)=P\cap S=P\cap$Closure$(P)=P$，因此，只需证明 L 是稠密集。

反证。假设 L 不为稠密集，则存在一非空开集 $A\subseteq\sim L$，即 $A\subseteq S-P$。因此 $P=S-A=$Closure$(P)-A$。由于 A 为开集，则 Closure(P)−A 为闭集，即为安全性。显然，这与 Closure(P)为包含 A 的最小安全性矛盾。因此，L 为稠密集。得证。

4 程序性质的时序分析

本节，对并发程序的活性及安全性进行时序分析，所使用的时序逻辑是线性将来时间命题时序逻辑。其语法及语义可参见文[7]。特别，一个解释为(σ,i)，其中 σ 是 $s_0,s_1,\cdots$ 是一无穷状态序列；i 是一非负整数。$\models$ 为解释(σ,i)和时序公式之间的可满足性关系。如果$(\sigma,0)\models p$，则称无穷序列 σ 满足时序公式 p。对一时序公式 p，由 p 所表达的性质即为集合：$\{\sigma\in S^{\omega}|(\sigma,0)\models p\}$。

对一时序算子集合 O_{π}，称公式集合 F 为算子集合 O_{π} 的正公式集，如果 F 中每一公式仅由 O_{π} 中时序算子及命题联接词 $\vee$，$\wedge$ 及 $\sim$ 组成且 O_{π} 中算子不出现在联接词的辖域中。

4.1 安全性分析

命题4.1 (1)每一命题公式是一安全性。(2)如果 p,q 是安全性，则 $p\wedge q$，$\bigcirc p$，pWq 及 $\Box p$ 均是安全性。

证明：由安全性定义，不难看出，每一命题公式表达了一安全性，即结论(1)成立。对结论(2)，不难由安全性定义及对时序公式的构成结构进行归纳可得，此处略去，详细可参见文[7]。

由上述命题知，所有由时序算子 W、$\Box$及$\bigcirc$得到的正公式均是安全性。正如文[8]中所讨论的，安全性的上述结论也是完备的。

命题4.2 命题时序逻辑中可表达的安全性集合正是只使用时序算子 W、$\Box$及$\bigcirc$而得到的正公式所表达的性质集合。

命题4.3 只使用时序算子 W 的每一正公式是哑元封闭安全性公式。

证明：对只使用算子 W 的每一公式结构进行归纳，不难得到结论。

同样，由文[8]中讨论，有下述更强的结论。

命题4.4 命题时序逻辑中可表达的哑元封闭安全性正是仅使用时序算子 W 而得到的正公式所表达的性质集。

命题4.5 只使用时序算子$\Box$的每一正公式是一强安全性。

证明：由强安全性定义，命题4.2及对正公式结构进行归纳，不难得出上述结论。

文[8]中给出了有穷状态自动机定义的强安全性类与时序公式的关系。

命题4.6 如果 p 是一由有穷状态自动机定义的强安全性，则存在一个由时序算子$\Box$构成的正公式，它表达了性质 p。

证明：见文[8]中引理3.6。

命题4.7 有穷状态自动机定义的强安全性正是只使用时序算子$\Box$的正公式所表达的性质类。

证明：见文[8]中引理3.3的构造性证明。

由文[9]可知，对每一命题时序逻辑可表达的性质均存在一无穷字串上的自动机，它定义了这一性质。因此不难得出下述结论。

命题4.8 命题时序逻辑公式表达的强安全性，正是由只使用时序算子$\Box$的正公式所表达的性质类。

由上述给出的各种安全性时序语法特征可知，

强安全性是最特殊的安全性定义。

4.2 活性分析

命题4.9 命题时序逻辑中的公式 p 表达了绝对活性当且仅当 p 是可满足的且 p 和◇p 是表示等价的，即它们表达了相同性质集合。

证明：设 p 是一绝对活性，则由定义2.7知，p 是可满足的(非空)。因此，只需证明 p 和◇p 是表示等价的。由于 p 是绝对活性，显然，任一满足公式 p 的无穷序列也满足◇p。对任一满足公式◇p 的无穷序列 σ，存在某一 $i\geqslant 0$，$(\sigma,i)\models p$。由于 p 为绝对活性，$(\sigma|_{i-1})\cdot(s_i,s_{i+1},\cdots)\in p$，即 σ 也满足 p。因此 p 和◇p 是表示等价的。

反之，设 p 是可满足的且 p 和◇p 是表示等价的。对任一满足 p 的无穷序列 σ 及序列 $\beta\in S^*$，显然 $\beta\cdot\alpha$ 满足◇p。由于 p 和◇p 是表示等价的，因此 $\beta\cdot\alpha$ 也满足 p。由定义知，p 是绝对活性。证毕。

对命题 p 和 q，由命题4.9知，◇p 表达了绝对活性；而□(p→◇q)并不是绝对活性公式，但它是活性公式。

命题4.10 命题时序逻辑中的公式 p 表达了一稳定性当且仅当 p 和□p 是表示等价的。

上述结论不难由稳定性定义得到。由命题4.9，命题4.10及公平性定义2.9，立即可得下述结论。

命题4.11 公式 p 表达了公平性当且仅当 p 是可满足的，且 p 和◇p 及 p 和□p 均是表示等价的。

对命题公式 p，由上述命题知道，公式□◇p→□◇q 及◇□p→□◇q 均表示了公平性，它们常常被称为强公平性和弱公平性。有关详细讨论可见文[6]。

结束语 本文对并发程序的两类重要性质：安全性和活性进行了分析和讨论，可以得到以下一些主要结论：(1)除了平凡性质(空集或整个序列空间)外，任何安全性与活性都是不相交的。每一性质类都包含了许多熟悉的并发程序性质。(2)任何性质都能够表示为一个安全性和一个活性的交。(3)安全性是有穷可反驳的。本质上，它们能够通过使用不变式原理进行证明。活性的证明需要使用结构化归纳，即对某一状态函数进行显式归纳。其代表方法是证明格(proof lattice)或良序集原理(Well-founded set principle)[10]。(4)表达安全性及活性的时序逻辑公式具有不同的时序语法特征。

上述结论的应用可归纳为：

·规范。任何程序的性质可分为安全性及活性两部分，说明并发程序可以通过对这两部分分别加以说明，以获得规范的完备性。

·验证。本文给出的表达程序性质的时序公式的语法特征可以帮助判定每一程序性质所属的性质类。根据程序性质所属的不同类(安全性类或活性类)，可以选择合适的证明规则对其加以证明推理。

·规范语言的选择。对某些特殊问题，有时并不需要去说明活性，只需对问题的安全性进行表达。此时，只需使用一般的谓词逻辑，亦即使用有穷行为上的谓词语言，并不需使用更复杂的规范语言。时序逻辑语言在无穷行为序列上定义谓词，它主要用于表达程序的活性，其时序推理比一般逻辑推理更复杂。这一结论非常有助于在规范语言的复杂性及表达能力之间作出选择，以适合不同的实际问题需要及便于实际应用。

参考文献

[1] Manna Z., Pnueli A., The Temporal Logic of Reactive and Concurrent System: Specification, New York, Spinger-Verlag, 1991

[2] Lamport L., Proving the Correctness of Multiprocess Programs, IEEE Trans. on Soft. Eng. SE-3(2) 1977

[3] Lamport L., Basic Concepts, In: Distributed Systems-Methods and Tools for Specification, Lec. Notes in Comp. Sci. 190, Spring-Verlag, Berlin, 1985

[4] Alpern B., Schneider F. B., Defining Liveness, Infor. Proc. Letters, 21(4)1985

[5] Emerson E. A., Alternative Semantics for Temporal Logic, Theor. Comp. Sci., 26, 1983

[6] Francez N., Fairness, New York, Springer-Verlag, 1986

[7] 贾国平，时序逻辑研究：模型、方法及应用[博士学位论文]，南京大学研究生院，1996

[8] Sistla A. P., Safety, Liveness and Fairness in Temporal Logic, Formal Aspects of Computing, 6, 1994

[9] Vardi M. Y., Wolper P., An Automata-theoretic Approach to Automatic Program Verification, In: Proc. 1st IEEE Symp. on Logic in Comp. Sci. 1986

[10] Owicki L., Lamport L., Proving Liveness Properties of Concurrent Programs, ACM Trans. on Prog. Lang. and Syst. 4(3)1982