计算机科学 2005Vol. 32№. 11

接口自动机——一种用于组件组合的形式系统*)

张 岩 胡 军 于笑丰 李宣东 郑国梁
(南京大学计算机软件新技术国家重点实验室 南京 210093)
(南京大学计算科学与技术系 南京 210093)

摘 要 接口自动机是描述基于组件系统中组件及组件间交互行为的形式化工具。接口自动机在处理组件组合问题时所使用的"乐观方法"和博弈思想是区别于其它形式化工具的关键点。本文对接口自动机、时间接口自动机和资源接口及其中的博弈思想进行综述。在同其它形式化方法比较的基础上,指出了接口自动机的长处和局限。文中总结了接口自动机在理论上和实际中的意义并对其应用前景做了展望。

关键词 接口自动机,时间接口自动机,资源接口,乐观方法,博弈

Interface Automata——A Formal System for Components Composition

ZHANG Yan HU Jun YU Xiao-Feng LI Xuan-Dong ZHENG Guo-Liang
(State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093)
(Department of Computer Science & Technology, Nanjing University, Nanjing 210093)

Abstract Interface automata is a new formal system used to specify components and interaction among them in the component-based system. Optimistic approach and game-theoretic foundations that deal with composition are the prominent characteristics of interface automata and are distinct from other formal methods. This paper surveys interface automata, timed interface automata, resource interface and game-based idea in them. By contrast with other formal methods, merits and limits of interface automata are summarized. This paper also gives significances of interface automata in theory and in practice, and the future of the application of interface automata.

Keywords Interface automata, Timed interface automata, Resource interface, Optimistic approach, Game

1 介绍

基于组件的软件开发(Component-based Software Development, CBSD)方法已被业界广泛采用,它带来的好处是:便捷、可靠、灵活和高效。组件的粒度可大可小。小到一个过程、一个对象或一个模块,大到一个代理(agent)、一个服务或一个应用软件包均可看作是组件。组件仅通过接口来与环境进行交互,因此常采用对接口进行形式化描述的方法来刻画组件的特性。关于这方面的研究长期以来一直受到学术界和产业界的重视[1~5]。

2001年由 Luca de Alfaro 和 Thomas A. Henzinger 正式提出一种新的用于描述接口的理论——接口自动机(Interface Automata)[6]。与已有的各种方法不同,在该理论中有两个突出的特点:"乐观方法"(optimistic approach)和博弈(game)思想。前者用于定义开放式系统中的接口兼容问题;后者用于描述这一问题的语义。

接口自动机描述了组件接口的时序特性,即接口交互活动间的先后顺序关系。接口自动机虽然基于传统的自动机,但它将接口的输出保证(output guarantee)和输入假设(input assumption)整合在一起,纳入到该形式化系统中加以处理。输出保证反映了本接口请求的各外部服务间的先后顺序关系——相当于对环境所作的假设;输入假设反映了本接口对外提供的各服务间的先后顺序关系——相当于自身行为的一种描述。此外,接口自动机还引入博弈思想来处理接口与环境间的关系,即判别接口兼容性的新方法——"乐观方法"(optimistic approach)。它与传统的所谓"悲观方法"(pessimistic approach)不同,认为只要存在一个可保证组合后的接口能在其中正确运行的环境,那么原接口就是可兼容的。同时,由于采用了"乐观方法"来处理接口组合问题,使得组合后的接口状态大幅度减少,因而接口自动机又是一种轻量级的形式化系统。这也是接口自动机优于那些基于"悲观方法"的形式化工具的所在。

在接口自动机理论中,将接口与环境始终看作是一个博弈中的对立双方。环境一方的目标就是要尽一切可能找到一个策略来满足接口的所有输入假设。判别两个接口是否兼容就是求解博弈中环境一方的获胜策略。若环境有获胜策略则两个接口是兼容的,否则两个接口就是不兼容的。

接口自动机的提出者还在接口自动机的基础上增加时间约束和资源约束,从而扩充称为时间接口自动机(Timed Interface Automata, TIA)[7]和资源接口(Resource Interface)[8]。它们分别用于描述实时环境和有限资源环境中接口的交互和组合特性。在这两种接口自动机中,也使用了"乐

*)本论文工作得到国家自然科学基金项目编号:60273036、国家重大基础研究计划973项目编号:2002CB312001、江苏省自然科学基金项目编号:BK2004080的资助。张 岩 博士生,主要研究方向为软件工程,形式化方法,软件度量;胡 军 博士生,主要研究方向为软件工程,嵌入式软件建模,形式化方法;于笑丰 博士生,主要研究方向为软件工程,模型驱动转换与验证;李宣东 教授,博士生导师,主要研究软件工程、形式化方法、模型检验;郑国梁 教授,博士生导师,主要研究软件工程、形式化方法。

观方法"和博弈思想来处理接口组合问题。

博弈的思想起源于对数字电路的合成问题的研究。Church 最先给出了博弈中一些核心问题的形式化描述[9]。博弈的基本思想很简单:在一个有限有向图上一个标记(token)被博弈中的双方沿着边的方向从一个结点移到另一个结点。每一方只有在标记落在属于它们控制的结点上时才能移动它。当某个条件满足时,则认为其中一方获胜。双方的目标是使自己获胜。博弈的思想虽然简单但其应用却很广泛,包括描述程序设计语言和逻辑系统的语义[10]、模型检验[11,12]、软件模块的合成及离散事件系统的行为描述[13,14]等。这主要是因为博弈理论反映了系统元素与外界环境互相影响的这样一类开放式系统的本质特征。我们认为博弈理论的作用有:一、作为一种语义工具描述待解的问题。利用博弈理论可以将诸多不直接相关的问题纳入到同一个语义描述框架中加以解释。二、用博弈理论给出待解问题的复杂度。三、用于模型检测,可给出不满足条件时的出错记录。

在接口自动机中就是用博弈理论来描述接口可兼容的语义并给出判断接口可兼容算法的复杂度。

本文将对接口自动机及其中的博弈思想进行介绍,第 2 节中简要介绍博弈理论和接口自动机的定义;第 3 节中分别描述了接口自动机、接口自动机的组合以及时间接口自动机和资源接口中的博弈思想;第 4 节在对比其它方法的基础上指出接口自动机的长处和局限;第 5 节总结了接口自动机在理论上和实际中的意义;第 6 节给出对接口自动机的研究前景的展望。

2 博弈和接口自动机

2.1 博弈的介绍[15]

本节主要介绍双人无限博弈(infinite two-person game)的一些基本概念。

一个博弈由两部分组成:场界(arena)和获胜条件(winning condition)。

定义 1 场界是一个三元组 $\mathcal{A}=(V_0,V_1,E)$,其中 V_0 是 0-结点集,V_1 是 1-结点集,$V_0\cap V_1=\emptyset$。$E\subseteq(V_0\cup V_1)\times(V_0\cup V_1)$ 为边的集合,也称作步(move)集合。记 $V=V_0\cup V_1$,则 $v\in V$ 的后继集为:$vE=\{v'\in V\mid(v,v')\in E\}$。

将博弈中的双方称作局中人 0 和局中人 1。若令 $\sigma\in\{0,1\}$,则可用局中人 σ 泛指任一方,并用局中人 $\bar{\sigma}$ 指其对手方。

一个博弈的进行可这样理解:有一个标记放在某个初始结点 v 上,若该结点为 σ-结点则由局中人 σ 将该标记移动到 $v'\in vE$。如此进行下去,一直到某结点 $\bar{v}$ 没有后继节点,即 $\bar{v}E=\emptyset$ 为止,结点 $\bar{v}$ 称为僵局(dead end)。

定义 2 场界 $\mathcal{A}$ 中的一局(play)是一条无限路径 $\pi=v_0v_1v_2\cdots\in V^\omega$,其中对任意的 $i\in\omega$ 有 $v_{i+1}\in v_iE$,并称为无限局;或是一条有限路径 $\pi=v_0v_1v_2\cdots v_l\in V^+$,其中对任意的 $i<l$ 有 $v_{i+1}\in v_iE$,并称为有限局。

定义 3 令 $\mathcal{A}$ 为场界且 $\mathrm{Win}\subseteq V^\omega$,则称 $\mathcal{G}=(\mathcal{A},\mathrm{Win})$ 为一个博弈。其中 $\mathcal{A}$ 是博弈的场界,Win 是获胜集。局中人 σ 在 $\mathcal{G}$ 的某一局 π 中获胜,当且仅当 $\pi=v_0v_1v_2\cdots v_l\in V^+$ 是有限局且 v_l 是 $\bar{\sigma}$-结点,或 π 是无限局且 $\pi\in\mathrm{Win}$。

定义 4 令 $\mathcal{A}$ 为场界,设 $\chi:V\to C$ 为一函数,它将场界中的结点映射到一个称作 ‖ 卜颜色 ┤ 的有限集合上。若 $\pi=v_0v_1v_2\cdots$ 是博弈中的一局,则 $\chi(\pi)=\chi(v_0)\chi(v_1)\chi(v_2)\cdots$。因此,若将 C 视作一个有限 ω-自动机[16]的状态空间,Acc 是其接受条件,并将所有无限局 π 构成的获胜集记为 $W_\chi(\mathrm{Acc})$,其中 $\chi(\pi)$ 是在 Acc 下可接受的。我们可以定义不同的获胜条件,这里仅介绍 Büchi 条件:$\pi\in W_\chi(\mathrm{Acc})$ 当且仅当 $\mathrm{Inf}(\chi(\pi))\cap F\neq\emptyset$,其中 $\mathrm{Acc}=F\subseteq C$,$\mathrm{Inf}(\chi(\pi))$ 表示在 $\chi(\pi)$ 中出现无穷多次的状态。对应 Büchi 条件的博弈称作 Büchi game。

此外还有 Muller game、Rabin game 等等[15],他们分别对应不同的获胜条件,在此不一一介绍了。Büchi game 在资源接口自动机中要用到。下面给出在接口自动机和时间接口自动机中要用到的另一类博弈--可达性博弈(Reachability game)。

定义 5 给定场界 $\mathcal{A}=(V_0,V_1,E)$ 和集合 $X\subseteq V$。若一旦 X 中的某个结点或局中人 $\bar{\sigma}$ 的僵局在一局 π 中出现就认为局中人 σ 在 π 中获胜,这样的博弈称作可达性博弈,记为 $R(\mathcal{A},X)$。

部分函数 $f_\sigma:V^*V_\sigma\to V$ 表示根据局中人 σ 在博弈中所走步的历史记录为其选择下一步。若对局 $\pi=v_0v_1\cdots v_l$ 的每个前缀 $v_0\cdots v_i(0\leqslant i<l$ 且 $v_i\in V_\sigma)$,f_σ 在其上均有定义且 $v_{i+1}=f_\sigma(v_0\cdots v_i)$,则称 π 符合 f_σ(对于无限局有类似的定义)。一局 π 从 $U\subseteq V$ 中结点开始且不会终止于 σ 的僵局上且符合 f_σ,若 f_σ 在 π 的每个前缀上均有定义则称 f_σ 是 σ 在 U 上的策略。设 f_σ 是 σ 在 U 上的一个策略(strategy),若 σ 在所有符合 f_σ 且从 U 中结点开始的局中均能获胜则称 f_σ 是 σ 在 U 上的获胜策略。特别地,当 U 只含一个元素 v 时,称 f_σ 是 σ 在 v 上的获胜策略。

在博弈理论中主要解决这样三类问题:一、在一个博弈中,局中人 σ 是否可以在不考虑 $\bar{\sigma}$ 采取何种策略的情况下获胜。这涉及到博弈是否为"确定的"问题。二、在有限图上进行的博弈中,能否判定哪一方获胜;若能,复杂度如何。已有结论显示,判定的复杂性与博弈类型密切相关[17]。三、能否给出博弈中获胜方的获胜策略。由于获胜策略中的每一步是根据以前所走各步的历史记录来确定的,这就又产生了 memoryless 策略——不依赖历史记录和 forgetful 策略——仅依赖有限数量的历史记录。它们的存在与否与特定的博弈类型有关[15]。

2.2 接口自动机的介绍[6]

定义 6 接口自动机 P 是一个六元组,$P=\langle V_P,V_P^{in},\mathcal{A}_P^I,\mathcal{A}_P^O,\mathcal{A}_P^H,\mathcal{T}P\rangle$,其中

(1)V_P 是状态集合;

(2)$V_P^{in}\subseteq V_P$ 是初始状态集,且 V_P^{in} 中至少包含一个元素;若 $V_P^{in}=\emptyset$ 则称 P 为空;

(3)$\mathcal{A}_P^I$,$\mathcal{A}_P^O$,$\mathcal{A}_P^H$ 分别为输入活动、输出活动和内部活动集合,且两两互不相交;记所有活动的集合为 $\mathcal{A}_P$,$\mathcal{A}_P=\mathcal{A}_P^I\cup\mathcal{A}_P^O\cup\mathcal{A}_P^H$;

(4)$\mathcal{T}_P\subseteq V_P\times\mathcal{A}_P\times V_P$ 是迁移的集合。

定义 7 接口自动机 P 的一个执行片段是其状态与活动交替排列的有限序列:$v_0a_0v_1a_1\cdots v_n$,其中 $(v_i,a_i,v_{i+1})\in\mathcal{T}_P$,$0\leqslant i<n$。任给两个状态 $v,u\in V_P$,若存在一个执行片段,其第一个状态为 v 且最后一个状态为 u,则称 u 从 v 可达。若 u 从某个初始状态 $v\in V_P^{in}$ 可达,则称 u 在 P 中是可达的。

定义 8 两个接口自动机 P 和 Q 若满足如下条件则称它们是可组合的:

$$\mathcal{A}_P^H\cap\mathcal{A}_Q=\emptyset\quad\mathcal{A}_P^I\cap\mathcal{A}_Q^I=\emptyset$$
$$\mathcal{A}_P^O\cap\mathcal{A}_Q^O=\emptyset\quad\mathcal{A}_Q^H\cap\mathcal{A}_P=\emptyset$$

两个接口自动机是可组合的要满足两个条件，一是它们不能有相同的输入或输出活动，二是其中一方的内部活动不能与另一方的任何活动相同。

设有任意两个可组合的接口自动机 A 和 B，其组合记为 $A \| B$。显然 $A \| B$ 中可能会有可达的非法状态（或称不兼容状态）存在。非法状态是这样产生的：在该状态上 A 为 B 提供的输入（即 A 的输出作为 B 的输入）不满足 B 的输入假设或 B 为 A 提供的输入（即 B 的输出作为 A 的输入）不满足 A 的输入假设—总之，在该状态上 A、B 的输出保证和输入假设不匹配。这样，就存在 A 和 B 的组合是否兼容的问题。

若存在一个环境可以保证 $A \| B$ 在该环境中能正常运行，且 $A \| B$ 中的非法状态永远不可到达，则称接口自动机 A 和 B 是兼容的，并将这样的环境称为合法环境。这里，环境实质上也可看作是一个接口自动机，不妨记为 E。$A \| B$ 在环境 E 中运行，实质就是 $A \| B$ 与环境 E 进行组合，即 $(A \| B) \| E$。

时间接口自动机是接口自动机的一种扩充，即为输入/输出活动增加了时间约束。这种扩充相当于时间自动机对传统自动机的扩充。利用时间接口自动机可以描述组件组合的一些实时特性[7]。

时间接口自动机在接口自动机的基础上增加了时钟变量，每个状态上均有一组由时钟变量构成的谓词，称为时钟条件。时钟条件分别对输入/输出活动在该状态上的触发时间进行约束，对应地分别称为输入不变式和输出不变式。同时为接口自动机中的每个输入/输出活动增加一个卫式（guard），它规定了该输入/输出活动的发生所必须满足的时间约束条件。注意：时钟条件是指自动机要一直处于某个状态时所必须满足的时间约束条件；卫式是指引起自动机从一个状态迁移到另一个状态的活动的发生时间所必须满足的约束条件。这两个概念与时间自动机中相应的概念是一致的，唯一的区别是将时钟条件进一步区分为输入不变式和输出不变式（各自使用不同的时钟变量）。这样做是为了避免用同一个不变式约束输入和输出活动会给处理接口组合问题带来一些麻烦[7]。

由于输入/输出活动本身是不占用时间的，即自动机从一个状态迁移到另一个状态所用的时间均为 0。因此可以通过执行一个输入/输出活动的无限序列，使得自动机的时间永远停在某一时刻不变，即所谓的齐诺行为（Zeno behavior）。这与实际情况是相违背的。为此，给出良构标准（well-formedness criterion），即在时间接口自动机中的任何一个可达状态上，均应保证约束输入/输出活动的时间变量不会停止变化，除非是因输入活动的齐诺行为使约束输出活动的时间变量保持不变，或因输出活动的齐诺行为使约束输入活动的时间变量保持不变的情况发生。

若两个时间接口自动机之间没有相同的输出活动且没有相同的时钟变量，则称这两个时间接口自动机是可组合的。

设有任意两个可组合的时间接口自动机 A 和 B，其组合记为 $A \| B$。与接口自动机的组合相似，$A \| B$ 中也会有可达的非法状态存在。不同的是非法状态的产生有两种情况：一种情况与接口自动机中非法状态的产生原因相同，即 A、B 的输出保证和输入假设不匹配造成的（注意，此时的匹配还包括输入/输出活动上卫式的满足），称为即时非法状态；另一种情况是该状态不满足良构标准，称为时间非法状态。

与接口自动机类似，若存在一个环境可以保证 $A \| B$ 在该环境中能正常运行，且 $A \| B$ 中的非法状态永远不可到达，则称时间接口自动机 A 和 B 是兼容的。

资源接口用来描述这样的问题：环境中有一系列资源，而资源的数量都是有限的。组件的行为要消耗环境中一定数量的资源（在某些情况下，组件的行为也可以生成一定数量的资源）。利用资源接口可以判别两个组件组合后它们所消耗的资源是否会超过资源总量，也可求解为使两个组件组合后能正常运行所需的最少资源数量是多少等一系列问题[8]。

资源接口在接口自动机的基础上为每个状态增加一组资源标志（resource label），它显示了系统处在该状态时所消耗或产生的相应资源的数量。

对两个资源接口自动机的组合要考虑 Δ-可兼容问题。有任意两个资源接口 A 和 B，其组合记为 $A \| B$。若存在一个环境使得 $A \| B$ 在其中能正常运行，且所消耗的资源数量不超过 Δ，则称资源接口 A 和 B 是 Δ-可兼容的。

由于篇幅的限制，本节没有给出时间接口自动机和资源接口自动机的形式化定义，以及它们各自的可组合和可兼容的形式化定义。相关内容可参考文[7,8]。

此外，接口自动机理论中用到的求精关系，即 alternating refinement，此处也未作介绍。alternating refinement 源自 alternating transition system（ATS）中的求精定义[18]，在接口自动机中只是借用。alternating refinement 与传统的求精定义不同，它采用“逆变”的形式。一般地讲，两个接口自动机 P 和 Q，若 Q 可以模拟 P 的所有输入活动且 P 可以模拟 Q 的所有输出活动，则称 Q 是 P 的求精（或细化）。换句话说，Q 是通过对 P 的输入假设进行放宽，同时对 P 的输出保证进行约束而得到的。形式化的定义可参考文[6]。

3 接口自动机中的博弈思想

在第 2 节中介绍了接口自动机、时间接口自动机和资源接口中关于两个接口是可兼容的定义。从定义中可以看出它们的共同点是：均需判断是否存在一个能使组合后的接口中的非法状态不可到达的合法环境。如何判断合法环境是否存在呢？在接口自动机理论中引入了博弈思想，通过求解组合后的自动机的状态集合上的博弈来检验合法环境是否存在，从而判断两个接口是否兼容。下面将对三种接口中的博弈思想进行详细介绍。

以下在不引起混淆的情况下，将三种接口均简称为接口。A、B 分别代表可组合的接口，$A \| B$ 代表它们的组合。

3.1 接口自动机及其组合的博弈解释

用博弈方法对接口进行建模，其思想是这样的：参与博弈的双方分别为接口和环境。在博弈的进行中，接口一方所走的每一步代表接口所产生的输出；环境一方所走的每一步代表环境提供给接口的输入。接口可能产生的输出以及它能够接受的输入均依赖于接口的内部状态，即接口仅在某些内部状态上产生相应的输出，称作输出保证，同时仅在某些内部状态上接受相应的输入，称作输入假设。这样一个接口便可看作是在接口的内部状态集上反复进行的一个博弈。环境一方的获胜条件是在博弈的进行中使输入假设永远得到满足。若对于环境一方存在一个获胜策略，则称该接口是良构的。

由上述的描述可以看出，“环境一方存在一个获胜策略”的含义是：环境为接口提供的输入可以满足接口的所有输入假设，即环境不会在接口不能接受某个输入的内部状态上为其提供该输入。因此，所谓“接口是良构的”系指存在某个可

以满足接口所有输入假设的环境。这也就是说，该接口可以在一定环境中正确运行；反之，则该接口在任何环境中均不能正确运行，这时接口就是非良构的。

进而，可将博弈方法用于接口的组合上，其思想如下：应用博弈理论，将 $A \| B$ 和环境视作参与博弈的两方。同样，$A \| B$ 一方所走的每一步代表 $A \| B$ 所产生的输出（输出到环境）；环境一方所走的每一步代表环境提供给 $A \| B$ 的输入。于是构成在 $A \| B$ 的内部状态集上的一个博弈。若环境一方存在一个策略使得 $A \| B$ 中的所有非法状态永远不可到达——该策略称为环境的获胜策略，则认为 A 与 B 是兼容的，否则认为 A 与 B 是不兼容的。

由此可以看出，在博弈意义下两个接口是兼容的实际意味着存在这样一个环境，它可以按某种方式来适当选择其为组合后的接口所提供的输入，以此来避免到达所有非法状态。实质上，这样的环境保证了两个接口的输入假设在该环境中能够互相得到满足。接口自动机的提出者将这种判别组合兼容性的方法称为“乐观方法”。它与传统的判别组合兼容性的方法，即“悲观方法”相对立。在“悲观方法”中一旦两个组件组合后有可达的非法状态存在就认为这两个组件是不兼容的。这种方法实质上是基于这样的一个前提，即只有组合后的接口可以在任何环境中正确运行才认为两个组成部分之间是兼容的。与此相对，“乐观方法”的前提是只要存在一个环境使得组合后的接口能够在该环境中正确运行就认为两个组成部分之间是兼容的。

“悲观方法”适用于封闭的系统（即只有内部活动而无输入和输出活动），“乐观方法”适用于开放的系统。可以证明，若 $A \| B$ 为封闭的，即没有输入和输出活动，则用于判别 A 和 B 是否兼容的“乐观方法”便退化为“悲观方法”。

3.2 时间接口自动机和资源接口中的博弈解释

在时间接口自动机中，接口和环境被看作是博弈中的两方，接口被看作是在其内部状态集上反复进行的一个博弈。接口一方的每一步在满足时间约束的条件下可有两种选择：要么停在某个状态上不动，而让约束输出活动的时间变量增加，称为时间步（timed move），要么执行一个输出活动，称为即时步（immediate move）。与此相同，环境一方的每一步也有这样两种选择，只不过是对输入活动而言。无论接口和环境中的哪一方，若在某个状态上没有满足时间约束的时间步和即时步可选择，则判该方为负，而对方获胜。环境一方的获胜条件是：在满足良构标准的同时能使（带时间约束的）输入假设始终得到满足。若环境一方存在这样一个获胜策略则称接口是良构的。

当两个接口 A 和 B 组合时，将 $A \| B$ 和环境视作参与博弈的两方。$A \| B$ 在时间约束条件满足的前提下可选择代表输出的时间步或即时步，环境可选择代表输入的时间步或即时步。于是构成在 $A \| B$ 的内部状态集上的一个博弈。若环境一方存在一个策略使得 $A \| B$ 中的所有非法状态，包括即时非法状态和时间非法状态永远不可到达——该策略称为环境的获胜策略，则认为 A 与 B 是兼容的，否则认为 A 与 B 是不兼容的。

在资源接口中，依然用博弈思想来描述接口与环境的关系。为了能突出资源接口的特点，这里仅以两个接口组合的情况为例来加以说明。

将 $A \| B$ 看作是博弈中的一方，博弈中的另一方看作是个调度器。$A \| B$ 的每一步代表从 $A \| B$ 中的一个状态迁移到另一个状态；调度器的每一步代表调度器确定让 A 和 B 中的哪个接口来决定 $A \| B$ 中的状态迁移。于是构成一个在 $A \| B$ 的状态集合上的博弈，调度器一方的获胜条件是：使 $A \| B$ 在博弈的进行过程中所消耗的资源总量始终不超过某一给定阈值 Δ，当然同时也要保证 A 和 B 的输入假设能互相得到满足。由于调度器一方可以通过不让 A 和 B 中的任何一个接口来决定 $A \| B$ 中的状态迁移，即阻止 $A \| B$ 中的状态迁移来获胜，而这显然不是我们需要的结果。为此在博弈中引入 Büchi 条件——调度器必须无限多次地选中 A 和 B，于是这就构成一个 Büchi game。若调度器一方存在一个获胜策略，则 A 和 B 是 Δ-可兼容的；否则，A 和 B 就不是 Δ-可兼容的。

4 接口自动机的特点

形式化描述并研究组件接口及组件组合问题的方法已有很多，如基于自动机[19~22]、基于进程代数[23~25]、基于时序逻辑[26,27]和基于 Petri 网[28,29]等等。接口自动机与这些方法相比，突出的特点是：采用“乐观方法”来定义组合问题，并用博弈理论来解释“乐观方法”下组合问题的语义，及相关算法的时间复杂度。

在语法上接口自动机与 I/O 自动机[30]非常相似，但它们之间有着本质的不同。I/O 自动机是基于“悲观方法”，它要求在自动机的每个状态上都可以接受来自环境的任何输入活动；而接口自动机是基于“乐观方法”的。所以，接口自动机比 I/O 自动机应用起来更灵活更方便，因为其中少了许多不必要的迁移。Wen 等人在文[31]中给出了接口自动机到 I/O 自动机的转换方法，通过转换前后的对比我们可以更清楚地看到这点。

进程代数强调了进程间的交互，这虽然可以描述与“输入假设”相类似的约束条件，但在这种模型中没有体现博弈的思想。进程代数将进程组合中的兼容性问题看作是求解图的问题。两个进程组合后不会产生死锁（deadlock）—进程可执行的所有活动均被环境拒绝，便认为它们是兼容的。而接口自动机则将接口间的兼容性问题看作是求解博弈，即组合后的接口和环境是博弈中的对立双方，前者的目标是要达到不兼容状态，后者的目标是尽一切可能使不兼容状态不可达。若后者获胜则原来的两个接口是兼容的。这看起来更适合描述开放式系统中的组件与环境间的关系。

传统的基于逻辑的方法，用前置条件和后置条件来描述接口，这种方法已不适用于处理具有安全性（safety）和活性（liveness）的反应式系统（reactive system），因而产生了用时序逻辑来描述的方法，如 LTL、CTL 和 PDL 等。时序逻辑公式的可满足性问题可以转换成相应的自动机可接受问题，而后者处理起来往往比前者简单。博弈理论将时序逻辑和自动机统一到一个语义框架中并将二者相互联系起来。对于接口自动机来说它的描述能力应该不比时序逻辑弱。

接口自动机理论也存在一定的局限性：

利用输入假设能够描述环境可以产生某个输入，但不能描述环境必须产生某个输入。这就导致了存在这样一个环境使得任何组件或任何组件的组合在其中都可正确运行。这个环境可以这样构造：它接受组件产生的所有输出，同时提供组件所需的一切输入接口，但却永远不产生这些输入活动。

文[32]中所给出的“BOTTOM automata”就是这样的环境。该环境在构造组件的类型系统和求解接口自动机的组合

时是有理论价值的。但在我们的研究过程中发现，由于存在这样的环境，对想通过已知的两个接口来构造能使它们组合后正确运行的环境造成一定困难。接口自动机的提出者也曾指出可以通过增加实时约束、公平性和同步性等条件来扩展接口自动机，使其具有能描述环境必须产生某个输入的能力[6]。但到目前为止，除了时间接口自动机外未看到其它相关的研究成果公布。

一般的接口自动机是基于有限状态博弈(finite-state game)的，无法用于描述组件间存在递归调用的组合情况。Chakrabarti 等人利用 Stateful Software Module 接口(SM Interface)来解决这一问题并将 pushdown game 的思想应用于其中[33]。SM Interface 虽然不能算作是接口自动机的扩充，但其基本思想与接口自动机是完全一致的，特别是其中也运用了博弈理论。

当两个接口自动机组合时，相互匹配的输入/输出活动(即一方的输出恰是另一方的输入)间是同步的，而各自的内部活动之间是交错异步的(interleaved asynchronous)。这就带来两个问题：一、若两个组件间的交互是异步的，则用接口自动机来为组件建模就显得不合适[21]；二、由于接口自动机将输入和输出活动分开来处理，给一些原子活动的描述带来麻烦。例如：有原子活动 writefile()，在它被调用(作为输入活动)和写文件结束并返回(作为输出活动)之间是不允许中断的。但用接口自动机描述时输入活动和输出活动之间一定有个中间状态存在。于是在与其它接口组合时，原子活动可能被打断，从而产生非法状态。其结果往往导致直觉上可以组合的两个接口，计算结果却显示不能组合。Lee 等人在实际应用中发现了这个问题，并给出了相应的解决办法[32]。

5 接口自动机的意义

从理论上看，接口自动机理论是一个类型系统。

传统的类型系统中，是通过对接口上交互的数据的值域(即数据值的允许范围)进行形式化描述来表达输入假设和输出保证的。而在接口自动机中，输入假设和输出保证包含的内容不再是数据值的允许范围，而变成输入/输出活动的先后顺序；在时间接口自动机中则是输入/输出活动在时间上的约束；在资源接口自动机中则是输入/输出活动在资源上的约束。因此接口自动机理论实质上也构成一个类型系统[32]，它是传统类型系统在“行为类型”、“实时类型”和“资源类型”上的变异。

从实际应用的角度看，接口自动机为基于组件的开放式系统的研究提供了有力的描述工具。

基于组件的软件系统已成为软件应用的主流。CORBA、EJB、COM 均可看作是基于组件的。以电子商务(E-commerce)和 Web Services 为代表的面向服务的计算(Service Oriented Computing，SOC)也可看作是基于组件的，因为服务(service)本身就可看作是一个组件[34]。而以上这些系统又均是开放式的系统，即用户请求的功能是通过组件与组件间交互、协作来完成的。通过了解组件间协作的特性、从而把握系统的整体特性，这是一条切实可行的技术路线。如果我们把组件间的协作看作是接口的组合，把一个组件同其它组件的交互看作是该组件同环境的交互，那么上述的这些系统均可以用接口自动机来加以描述和研究。如果我们将接口自动机中的合法环境与基于组件系统中的 connector[35,36] 以及所谓的“glue code”作类比就不难发现它们在本质上是一致的。

Berkeley 大学的 Lee 和 Xiong 在他们的 Ptolemy 项目中用接口自动机为 PtolemyII 系统(一种用 Java 语言书写的软件开发环境)中的组件建立行为类型系统，并用接口自动机中的兼容性判定来做类型检查[32]。Chakrabarti 等人在一个传感器网络的项目 PicoRadio 中使用资源接口来优化路由算法，以保证在电池能量有限的环境中让网络能始终正常运行[8]。

展望 随着技术的发展，软件形态正在向一种基于 Internet 计算环境的新型软件形态演化[37]。这种新型软件系统是由一组通过 Internet 连接的，具有模块性、可组合性、能动性和反应性的软件实体组成的开放式系统。这种软件系统的结构不再是静态的，而呈现出自组织、自适应以及动态演化等特性。系统呈现给用户的整体功能也是通过软件实体间的自组织来完成的。我们要深入研究这种开放式系统，就必须描述其组成元素(即软件实体)与环境之间的关系，包括对环境的感知和使用。

接口自动机因其特有的“乐观方法”和博弈思想，将软件组件对环境的要求及其自身的行为描述集成到一个形式化系统之中，从而避免了单独为环境建模所带来的困难。因此接口自动机成为形式化描述软件实体的有力工具。我们认为接口自动机将在研究这种新型软件系统中软件实体的可组合性以及整个系统的自组织性和自适应性上发挥一定作用。

此外，接口自动机还可用于基于模块及模块间组合的其它开放式系统，如嵌入式系统、传感器网络(Sensor Network)、多代理系统(Multi-Agent Systems，MAS)和网格等。

由于接口自动机提出的时间不长，相关的研究不多，特别是在应用方面研究还不广泛。因此其理论的完备性还有待于我们在利用接口自动机解决实际问题的过程中不断检验、不断完善。

参 考 文 献

1 Cox P T, Song Baoming. A Formal Model for Component-Based Software. In: Proc. of the IEEE 2001 Symposia on Human Centric Computing Languages and Environments (HCC'01). Los Alamitos: IEEE Computer Society, 2001. 304～311

2 Casati F, Shan M C. Dynamic and adaptive composition of e-service. Information Systems, 2001, 26(3): 143～163

3 Heisel M, Santen T, Souquières J. Toward a Formal Model of Software Components. In: Proc. of the 4th Intl. Conf. of Formal Engineering Methods (ICFEM2002). Berlin: Springer-Verlag, 2002. 57～68

4 Narayanan S, Mcilraith S. Simulation, verification and automated composition of web service. In: Proc. of the 11th Intl. World Wide Web Conf. (WWW2002). New York: ACM Press, 2002. 77～88

5 Shaparau D. Approaches to Web Service Composition. http://science.unitn.it/～tomasi/think/pdf/sdmi.pdf, 2004-7-17

6 de Alfaro L, Henzinger T A. Interface Automata. In: Proc. of 9th Annual ACM Symposium on Foundations of Software Engineering (FSE 2001). New York: ACM Press, 2001. 109～120

7 de Alfaro L, Henzinger T A, Stoelinga M. Timed Interfaces. In: Proc. of the 2nd Intl. Conf. on Embedded Software (EMSOFT 2002). Berlin: Springer-Verlag, 2002. 108～122

8 Chakrabarti A, de Alfaro L, Henzinger T A, et al. Resource Interfaces. In: Proc. of the 3rd Intl. Conf. on Embedded Software (EMSOFT 2003). Berlin: Springer-Verlag, 2003. 117～133

9 Church A. Logic, arithmetic, and automata. In: Proc. of the Intl. Congress of Mathematicians. Djursholm, Sweden: Institut Mittag-Leffler, 1963. 23～35

10 Abramsky S. Semantics of interaction: an introduction to game semantics. Semantics and Logics of Computation. Cambridge: Cambridge University Press, 1997. 1～32

11 Berwanger D, Gradel E. Games and Model Checking for Guarded Logics. In: Proc. of the 8th Intl. Conf. on Logic for Programming and Automated Reasoning (LPAR 2001). Berlin: Springer-Verlag, 2001. 70～84

12 Walukiewicz I. Pushdown Processes: Games and Model Checking. Information and Computation, 2001, 164(2): 234～263

13 Pnueli A, Rosner R. On the synthesis of a reactive module. In: Proc. of the 6th Annual ACM Symposium on Principles of Pro-

gramming Languages (POPL'98). New York: ACM Press, 1989. 179～190
14 Thistle J G, Wonham W M. Supervision of infinite behavior of discrete-event system. SIAM Journal on Control and Optimization, 1994, 31(4): 1098～1113
15 Mazala R. Infinite Games. In: Grädel E, Wolfgang T, Wilke T, eds. Automata, Logics, and Infinite Games-A Guide to Current Research. Berlin: Springer-Verlag, 2000. 23～38
16 Büchi J R. On a decision method in restricted second order arithmetic. In: Proc. of the Intl. Congress on Logic, Methodology and Philosophy of Science. Stanford: Stanford University Press, 1962. 1～11
17 Klauck H. Algorithms for Parity Games. In: Grädel E, Wolfgang T, Wilke T, eds. Automata, Logics, and Infinite Games-A Guide to Current Research. Berlin: Springer-Verlag, 2000. 107～129
18 Alur R, Henzinger T A, Kupferman O, et al. Alternating Refinement Relations. In: Proc. of the 9th Intl. Conf. On Concurrency Theory (CONCUR'98). Berlin: Springer-Verlag, 1998. 163～178
19 Berardi D, Calvanese D, Giacomo G D, et al. Automatic Composition of e-Services that Export their Behavior. In: Proc. of the 1st Intl. Conf. on Service Oriented Computing (IC-SOC03). Berlin: Springer-Verlag, 2003. 43～58
20 Bultan T, Fu X, Hull R, et al. Conversation Specification: A New Approach to Design and Analysis of E-Service Composition. In: Proc. of the 12th Intl. World Wide Web Conf. (WWW2003). New York: ACM Press, 2003. 403～410
21 Fu X, Bultan T, Su J. Conversation Protocols: A Formalism for Specification and Verification of Reactive Electronic Services. In: Proc. of the 8th Intl. Conf. on Implementation and Application of Automata (CIAA 2003). Berlin: Springer-Verlag, 2003. 188～200
22 Moisan S, Ressouche A, Rigault J-P. Towards Formalizing Behavioral Substitutability in Component Frameworks. In: Proc. of the 2nd Intl. Conf. on Software Engineering and Formal Methods (SEFM'04). Los Alamitos: IEEE Computer Society, 2004. 132～141
23 Rajamani S K, Rehof J. A behavioral module system for the pi-calculus. In: Proc. of Static Analysis Symposium (SAS'01). Berlin: Springer-Verlag, 2001. 375～394
24 Lumpe M, Achermann F, Nierstrasz O. A formal Language for Composition. In: Leavens G T, Sitaraman M, eds. Foundations of Component-based System. Cambridge: Cambridge University Press, 2000. 69～90
25 Salaün G, Bordeaux L, Schaerf M. Describing and Reasoning on Web Services using Process Algebra. In: Proc. of the IEEE Intl. Conf. on Web Services (ICWS'04). Los Alamitos: IEEE Computer Society, 2004. 43～51
26 Lano K, Bicarregui J, Maibaum T. Composition of Reactive System Components. In: Leavens G T, Sitaraman M, eds. Foundations of Component-based System. Cambridge: Cambridge University Press, 2000. 267～283
27 Han J. Interaction Compatibility: An Essential Ingredient for Service Composition. In: Proc. of the 2nd Intl. Workshop on Grid and Cooperative Computing (GCC2003). Berlin: Springer-Verlag, 2003. 59～66
28 Hamadi R, Benatallah B. A Petri Net-based Model for Web Service Composition. In: Proc. of the 14th Australasian Database Conf. (ADC2003). New South Wales: Australian Computer Society Inc, 2003. 191～200
29 Tao X, Jiang C. Formalizing Web Service and Modeling Web Service-Based System Based on Object Oriented Petri Net. In: Proc. of the 2nd Intl. Workshop on Grid and Cooperative Computing (GCC2003). Berlin: Springer-Verlag, 2003. 1008～1011
30 Lynch N, Tuttle M. An introduction to input/output automata. CWI Quarterly, 1989, 2(3): 219～246
31 Wen Y, Wang J, Qi Z. Bridging Refinement of Interface Automata to Forward Simulation of I/O Automata. In: Proc. of the 6th Intl. Conf. Formal Engineering Method (ICFEM2004). Berlin: Springer-Verlag, 2004. 259～273
32 Lee E A, Xiong Y. Behavioral Type for Component-Based Design. University of California, Berkeley, CA 94720, USA, Technical Memorandum UCB/ERL: M02/29, 2002
33 Chakrabarti A, de Alfaro L, Henzinger T A, et al. Interface Compatibility Checking for Software Modules. In: Proc. of the 14th Intl. Conf. on Computer-Aided Verification (CAV2002). Berlin: Springer-Verlag, 2002. 428～441
34 Bennet K, Layzell P, Budgen D, et al. Service-Based Software: The Future for Flexible Software. In: Proc. of the 7th Asia-Pacific Software Engineering Conf. (APSEC'00). Los Alamitos: IEEE Computer Society, 2000. 214～221
35 Allen R, Garlan D. A Formal Basis for Architectural Connection. ACM Transactions on Software Engineering and Methodology, 1997, 6(3): 69～90
36 Min H G, Choi S W, Kim S D. Using Smart Connectors to Resolve Partial Matching Problems in COTS Component Acquisition. In: Proc. of the 7th Intl. Symposium on Component-Based Software Engineering. Berlin: Springer-Verlag, 2004. 40～47
37 杨芙清，梅宏，吕建，等. 浅论软件技术发展. 电子学报，2002, 30(12A): 1901～1906
38 Henzinger T A. Automata for Specifying Component Interfaces. In: Proc. of the 8th Intl. Conference on Implementation and Application of Automata (CIAA2003). Berlin: Springer-Verlag, 2003. 1～2
39 de Alfaro L, Henzinger T A. Interface theories for component-based design. In: Proc. of the First Intl. Workshop on Embedded Software (EMSOFT). Berlin: Springer-Verlag, 2001. 148～165
40 de Alfaro L, Stoelingal M. Interfaces: A Game-Theoretic Framework for Reasoning About Component-Based Systems. In: Proc. of the 2nd Intl. Workshop on Foundations of Coordination Languages and Software Architectures (FOCLASA 2003). New York: Elsevier, 2004. 3～23
41 de Alfaro L. Game Models for Open Systems. In: Proc. of Intl. Symposium on Verification (Theoryin Practice). Berlin: Springer-Verlag, 2003. 269～289
42 Henzinger T A. Rich Interfaces for Software Modules. In: Proc. of the 18th European Conf. on Object-Oriented Programming (ECOOP 2004). Berlin: Springer-Verlag, 2004. 516～517
43 Griss M L. Software Agents as Next Generation Software Components. In: Heineman G T, Councill W T, eds. Component-Based Software Engineering: Putting the Pieces Together. Boston: Addison-Wesley, 2001. 641～657

(上接第 201 页)

些公共的资源抽取出来，单独定义，以便全局引用。

行为(Behavior)— 行为部分定义控件的事件集合 Event，操作集合 A，以及联结这两者的 R_{CA}。

数据(Datamodel)— 即数据模型定义。它将 XML 格式的数据，根据数据依赖定义，使用第三节给出的数据依赖图相关算法构造其拓扑结构。

综合以上五个部分，GUI XML 可以很自然地定义各种 GUI 实例，并通过解释引擎实例化得到运行时的实例。

其他基于 XML 的 GUI 描述语言还有 XUL[4]，Bamboo-kit[5]等。与 GUI XML 相比，它们只针对某种 GUI 工具包进行描述，而且缺乏对数据描述模型的定义。GUI XML 由于采用了抽像描述模型以及相关理论，在扩展性、数据集成能力等方面都具有明显的优势。

结论 针对目前 GUI 设计和开发过程中由多 GUI 工具包和复杂数据所造成的困难，本文提出 GUI 抽象描述模型以及建立在该模型之上的数据和绑定模型。该模型采用实体和关系集合的方式定义 GUI，满足了描述的通用性。数据模型和绑定模型将异构的数据组织成有向图结构，并在该图与控件树之间建立多种绑定关系，从而解决了数据维护以及数据与控件间的同步问题。GUI XML 从实际应用的角度证明了以上模型的良好描述能力和通用性。

参 考 文 献

1 Buschmann F, Meunier R, Rohnert H, Sommerland P, Stal M. Pattern-Oriented Software Architecture: A System of Patterns. John Wiley and Sons, 1996
2 Hussey A, Carrington D. Comparing the MVC and PAC architectures: a formal perspective, Software Engineering. IEE Proceedings, 1997, 144(4)
3 Clark J, DeRose S. XML Path Language (XPath) Version 1.0 http://www.w3.org/TR/xpath/ Nov. 1999
4 Deakin Neil. XUL Tutorial 15 April 2004
5 Bambookit Tutorial. http://www.bambookit.com
6 清华大学—IT Frontier 株式会社知识工程联合实验室. GUI XML for JFC Swing 规范. Aug. 2004
7 张鹏，徐鹏. Java 图形用户界面的 XML 描述方法. 见：第七界全国 Java 大会，北京：2004