ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.1, January 2007, pp.50−61
DOI: 10.1360/jos180050

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

场景驱动的构件行为抽取*

张 岩[1,2+], 胡 军[1,2], 于笑丰[1,2], 张 天[1,2], 李宣东[1,2], 郑国梁[1,2]

[1](南京大学 计算机科学与技术系,江苏 南京 210093)

[2](计算机软件新技术国家重点实验室(南京大学),江苏 南京 210093)

Scenario-Driven Component Behavior Derivation

ZHANG Yan[1,2+], HU Jun[1,2], YU Xiao-Feng[1,2], ZHANG Tian[1,2], LI Xuan-Dong[1,2], ZHENG Guo-Liang[1,2]

[1](Department of Computer Science and Technology, Nanjing University, Nanjing 210093, China)

[2](State Key Laboratory for Novel Software Technology (Nanjing University), Nanjing 210093, China)

+ Corresponding author: Phn: +86-25-83593671, Fax: +86-25-83594683, E-mail: zhangyan@seg.nju.edu.cn

Zhang Y, Hu J, Yu XF, Zhang T, Li XD, Zheng GL. Scenario-Driven component behavior derivation. *Journal of Software*, 2007,18(1):50−61. http://www.jos.org.cn/1000-9825/18/50.htm

Abstract: Components with redundant functionalities, especially with undesired functionalities, can not be used properly by users. Therefore, the scenario-based behavior derivation of components is a significant problem that needs to be solved, where the scenario specifies the user's desired behavior. An approach is proposed to derive the desired behavior specified by a scenario specification from components. The main idea of this approach is that by constructing a special environment, i.e., supremum-inclusive environment (SIE), for a component, all behavior specified by a scenario specification can be extracted from the component to the composition of the component and its SIE, and other behavior of the component is discarded to the most extent. This paper uses interface automata to model the behavior of components and a set of action sequences to abstract the scenario specification in Message Sequence Charts (MSCs). The composition of the components is modeled by the product of interface automata. This paper gives the relevant algorithm in this approach and illustrates it by an example.

Key words: interface automaton; MSC (message sequence chart); supremum-inclusive environment; component; behavior derivation

摘 要: 如果构件含有冗余的功能,特别是含有用户不想要的功能,则无法被用户正确使用.因此,如何从构件中提取场景规约中所描述的用户想要的行为便是一个亟待解决的问题.给出了解决该问题的一种方法.该方法通过为构件构造一个环境,即极大包含环境,使得场景规约中所描述的所有行为可以从构件中抽取出来,并保留到该构件与其极大包含环境的组合中.同时,构件中的其他行为,即不在场景规约中的行为,被尽可能地舍弃.用接口自动机为构件的行为建模,并将用消息序列图描述的场景规约抽象为一组活动序列.构件的组合描述为接口自动机的乘积.给出了基于场景进行构件行为抽取的相关算法,并用一个实例对文中所述方法进行了说明.

* Supported by the National Natural Science Foundation of China under Grant Nos.60233020, 60425204, 60673125 (国家自然科学基金); the National Grand Fundamental Research 973 Program of China under Grant No.2002CB312001 (国家重点基础研究发展规划(973)); the Natural Science Foundation of Jiangsu Province of China under Grant No.BK2004080 (江苏省自然科学基金)

Received 2006-02-14; Accepted 2006-05-11

关键词: 接口自动机;消息序列图;极大包含环境;构件;行为抽取
中图法分类号: TP311 文献标识码: A

基于构件的软件开发(component based software development,简称 CBSD)为我们提供了一种利用已有构件通过“即插即用”的方式来开发复杂的软件系统的有效手段.利用这种开发方法,用户只需从构件库中提取所需构件并将它们组合在一起,从而构建出新的软件系统.在基于构件的软件开发中经常遇到的问题是已有构件不能恰好满足用户的需求.这常常干扰了用户有效地对构件进行复用.上述问题通常呈现为以下两种类型:(1) 单个构件的功能不能全部满足用户需求;(2) 单个构件中除了含有用户所需功能以外,同时还含有其他冗余的功能.第(1)类问题可以通过构件的组合来解决,而且目前在学术界已有不少这方面的研究[1−3].然而,第(2)类问题至今仍未见到有很好的解决方法.本文试图解决这一问题.

通常,用户使用场景规约(scenario specification)来给出需求.一个场景规约描述了一个构件在与其他构件交互时所应具有的行为,即用户想要的行为.如果能够从构件中抽取出用户所给场景规约中描述的所有行为,即对构件进行基于场景的行为抽取,那么第(2)类问题也就解决了.换句话说,就是要使得构件中所有包含场景规约中所描述的行为这一部分得到保留,而对构件中的其他部分要尽可能舍弃.

本文给出一种场景驱动的构件行为抽取的方法.该方法通过为构件构造一个环境(环境也可以被看作是一个构件),使得构件在该环境中运行时可以从构件中提取出场景规约所描述的所有行为.也就是说,场景规约所描述的所有行为被保留在构件和该环境的组合中.构件的行为模型用接口自动机(interface automata,简称 IA)[4]来描述.场景规约用消息序列图(message sequence chart,简称 MSC)[5]来描述.消息序列图进而被抽象成一组活动序列.构件的组合用接口自动机的乘积来描述.我们扩展了接口自动机理论中的环境概念,并引入极大包含环境(supremum-inclusive environment,简称 SIE).只要在已知的活动序列集合 L 下为给定接口自动机 R 构造极大包含环境 E_L,就可以使 R 中包含 L 中元素的所有行为保留在复合构件 $R\otimes E_L$ 中;同时,R 中的其他行为最大程度地不被保留在 $R\otimes E_L$ 中.

接口自动机将环境假设[4]和构件行为整合到同一个模型中,因此,它非常适合描述开放系统中的构件行为.正因为接口自动机具有这样的特点,所以我们选择其作为建模工具.

本文第 1 节简要介绍接口自动机和消息序列图.第 2 节引入场景驱动的构件行为抽取方法中的一些相关概念.第 3 节详细说明场景驱动的构件行为抽取方法并给出相关算法.第 4 节进行相关工作的比较.第 5 节总结全文并对今后工作进行展望.此外,用一个实例作为对文中所述方法的说明贯穿全文.同时,通过对该实例的研究,也反映了本文所述方法的有效性.

1 接口自动机与消息序列图

本节对接口自动机和消息序列图的有关概念进行简要介绍,其中多数概念参考自文献[4,5].

1.1 接口自动机

定义 **1**(接口自动机). 接口自动机 P 是一个六元组,

$$P=\left\langle V_P, V_P^{init}, A_P^I, A_P^O, A_P^H, T_P\right\rangle,$$

其中:

- V_P 是状态的有穷集合.
- $V_P^{init}\subseteq V_P$ 是初始状态集,且 V_P^{init} 中至少包含一个元素.若 $V_P^{init}=\varnothing$,则称 P 为空.
- A_P^I, A_P^O, 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 A_P,

$$A_P=A_P^I\cup A_P^O\cup A_P^H.$$

- $T_P\subseteq V_P\times A_P\times V_P$ 是迁移的集合.将 $(v,a,v')\in T_P$ 记为 $v\xrightarrow{a}_P v'$,并记

$$label(v\xrightarrow{a}_P v')=a,\ head(v\xrightarrow{a}_P v')=v,\ tail(v\xrightarrow{a}_P v')=v'.$$

若 $a \in A_P^I$,则称 $v \xrightarrow{a}_P v'$ 为输入迁移,记输入迁移的集合为

$$T_P^I = \{v \xrightarrow{a}_P v' \mid a \in A_P^I\}.$$

类似地,有输出迁移和内部迁移,它们的集合分别记为

$$T_P^O = \{v \xrightarrow{a}_P v' \mid a \in A_P^O\},\ T_P^H = \{v \xrightarrow{a}_P v' \mid a \in A_P^H\}.$$

若有 $v \xrightarrow{a}_P v'$ (其中 $v,v' \in V_P, a \in A_P$),则称活动 a 在状态 v 上是使能的(enabled).记状态 $v \in V_P$ 上使能的输入活动的集合为

$$A_P^I(v) = \{a \in A_P^I \mid \exists v' \in V_P.v \xrightarrow{a}_P v'\}.$$

类似地,有 $A_P^O(v) = \{a \in A_P^O \mid \exists v' \in V_P.v \xrightarrow{a}_P v'\}$ 和 $A_P^H(v) = \{a \in A_P^H \mid \exists v' \in V_P.v \xrightarrow{a}_P v'\}$.

令 $A_P(v) = A_P^I(v) \cup A_P^O(v) \cup A_P^H(v)$.

对于接口自动机 P,若有 $A_P^I = A_P^O = \varnothing$,则称 P 是封闭的;否则,称 P 是开放的.若 P 满足 $\left|V_P^{init}\right| = 1$ 且 $\forall v \xrightarrow{a}_P u$, $v \xrightarrow{a}_P u'.u = u'$,则称 P 是确定的;否则,称 P 是不确定的.为简单起见,约定本文中所涉及的接口自动机均是确定的.图 1 是一个接口自动机的示例(符号“?”,“!”和“;”分别表示该活动为输入活动、输出活动和内部活动.初始状态有一个无源的箭头指向它).

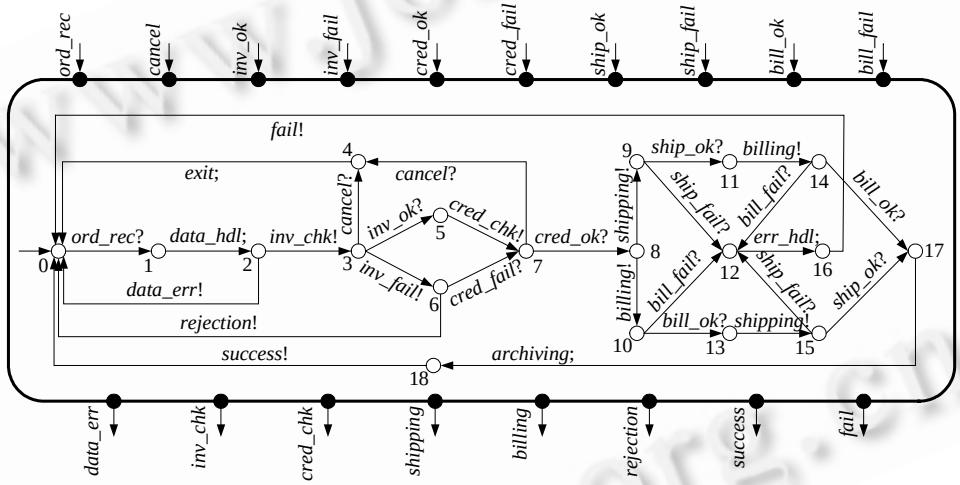

Fig.1 Interface automaton *Seller*

图 1 接口自动机 *Seller*

例 1:在一个电子商务系统中,销售方是用一个构件来实现的,我们用如图 1 所示的接口自动机 *Seller* 为该构件建立了行为模型.销售方接受客户发来的订单(*ord_rec*)并对订单中的数据进行相关处理(*data_hdl*),如数据格式转换等.如果在数据处理的过程中发现订单中的数据有问题,则向客户报告错误(*data_err*);如果数据处理正常完成,销售方将继续向供应商查询存货清单(*inv_chk*)并向银行查询客户的存款(*cred_chk*).当得知供应商可提供客户所订货物(*inv_ok*)并且客户在银行的存款可支付所订货物(*cred_ok*)时,销售方将通知发货人向客户发送货物(*shipping*)并同时通知银行结算相关帐目(*billing*).如果供应商的存货不能满足客户所订货物的要求(*inv_fail*),或客户的银行存款不能支付所订货物(*cred_fail*),销售方则拒绝客户本次的订单(*rejection*);客户也可以向销售方发送消息(*cancel*)来取消本次订购活动(*exit*).当发货(*ship_ok*)和结帐(*bill_ok*)顺利完成后,销售方会作相应的记录(*archiving*)并通知客户订单已完成(*success*);否则(*ship_fail* 或 *bill_fail*),销售方在处理完异常(*err_hdl*)后,会通知客户订单未能完成(*fail*).

定义 **2**(执行片段). 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穷序列:$v_0 a_0 v_1 a_1 \ldots a_{n-1} v_n$,其中,$v_i \xrightarrow{a}_P v_{i+1}$,$0 \le i < n$.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 $v \in V_P^{init}$ 可达,则称 u 在 P 中是可达的;否则,称 u 在 P 中是不可达的.

定义 **3**(可组合). 两个接口自动机 P 和 Q 若满足条件:

$$A_P^H \cap A_Q = \varnothing,\ A_P^I \cap A_Q^I = \varnothing,\ A_Q^H \cap A_P = \varnothing \text{ 且 } A_P^O \cap A_Q^O = \varnothing,$$

则称它们是可组合的.令 $shared(P,Q) = A_P \cap A_Q = (A_P^I \cap A_Q^O) \cup (A_P^O \cap A_Q^I)$.

定义 **4**(接口自动机乘积). 若 P 和 Q 是可组合的接口自动机,则它们的乘积 $P \otimes Q$ 定义为

$V_{P\otimes Q}=V_P \times V_Q$;

$V_{P\otimes Q}^{init} = V_P^{init} \times V_Q^{init}$;

$A_{P\otimes Q}^I = (A_P^I \times A_Q^I) \setminus shared(P,Q)$;

$A_{P\otimes Q}^O = (A_P^O \times A_Q^O) \setminus shared(P,Q)$;

$A_{P\otimes Q}^H = A_P^H \cup A_Q^H \cup shared(P,Q)$;

$$T_{P\otimes Q} = \{(v,u) \xrightarrow{a} (v',u) \mid v \xrightarrow{a}_P v' \wedge a \notin shared(P,Q) \wedge u \in V_Q\} \cup$$
$$\{(v,u) \xrightarrow{a} (v,u') \mid u \xrightarrow{a}_Q u' \wedge a \notin shared(P,Q) \wedge v \in V_P\} \cup$$
$$\{(v,u) \xrightarrow{a} (v',u') \mid v \xrightarrow{a}_P v' \wedge u \xrightarrow{a}_Q u' \wedge a \in shared(P,Q)\}.$$

定义 **5**(非法状态). 接口自动机 P 和 Q 是可组合的,$P \otimes Q$ 中的非法状态集合 $Illegal(P,Q) \subseteq V_P \times V_Q$ 为

$$Illegal(P,Q) = \{(v,u) \in V_P \times V_Q \mid \exists a \in shared(P,Q).((a \in A_P^O(v) \wedge a \notin A_Q^I(u)) \vee (a \in A_Q^O(u) \wedge a \notin A_P^I(v)))\}.$$

非法状态的含义是:在两个接口自动机的乘积中的某个状态上,一个接口自动机产生的输出活动恰是另一个接口自动机的输入活动,但那个接口自动机在此时却不能接受该输入活动,即在当前状态上该输入活动不是使能的.

定义 **6**(环境). 若接口自动机 E 满足如下条件,则称其为接口自动机 R 的环境:(1) E 和 R 是可组合的;(2) E 是非空的;(3) $A_E^I = A_R^O$;(4) 如果 $Illegal(R,E) \neq \varnothing$,则 $Illegal(R,E)$中的状态在 $R \otimes E$ 中均是不可达的.

上述定义中,条件(3)要求环境必须接受来自接口自动机的所有输出活动;条件(4)要求在接口自动机与其环境的乘积中不能有可达的非法状态.

1.2 消息序列图

消息序列图刻画了构件与其环境间消息交互的顺序,它可以形象地展示出被描述系统的行为轨迹.每个消息序列图都有等价的图形表示和文本表示,尤其是其图形表示可以直观地描述出系统的行为轨迹.因此,消息序列图被广泛用于书写场景规约.

消息序列图的基本元素是构件和消息流.带有题头(用矩形表示)的垂直直线代表构件.题头上方书写构件的名称.垂直直线上,自上而下依次排列有一系列事件.这些事件可以是消息发送事件、消息接收事件、定时器等.消息序列图中,从消息发送事件指向消息接收事件的带箭头的水平直线或斜线表示消息.根据事件间的序关系(例如,消息发送事件必须发生在同一消息的接收事件之前),可以从消息序列图中得到一组由该图所描述的消息序列.图 2 是一个消息序列图的示例.

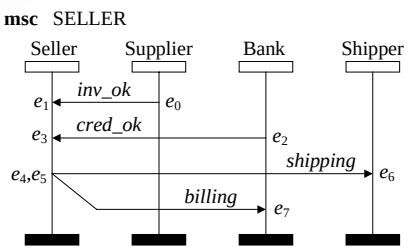

Fig.2 MSC 'SELLER' specifying a scenario of interactions between the seller component and other components

图 2 描述销售方构件与其他构件交互场景的消息序列图 SELLER

下面给出消息序列图及消息序列的形式化定义.

定义 **7**(消息序列图). 消息序列图 $Ch=\langle C,N,M,F,D\rangle$是一个 5 元组,其中:

- C 是构件的有穷集合;
- N 是事件的有穷集合.每个事件对应着消息的发送或接收;
- M 是消息的有穷集合.对任意消息 $m\in M$,用 $m!$和 $m?$分别表示发送和接收消息 m.任一事件 $e\in N$,它或者是某一消息 m 的发送事件或者是某一消息 m 的接收事件,分别记为$\lambda(e)=m!$和$\lambda(e)=m?$;
- $F:N\to C$ 是一个函数,它将每一个事件 $e\in N$ 映射到唯一一个构件 $F(e)\in C$ 上;
- $D\subseteq N\times N$ 是事件集合上的一个偏序关系.每个$(e,e')\in D$ 均满足 $e\neq e'$.(e,e')代表了消息序列图 Ch 中显示的事件 e 和 e'之间的先后顺序.

定义 **8**(消息序列). $Ch=\langle C,N,M,F,D\rangle$是一个消息序列图,$e_0e_1\ldots e_n$是 Ch 中的一个事件序列当且仅当它满足如下条件:

- $\{e_0,e_1,\ldots,e_n\}=N$;
- $e_i\neq e_j(0\le i,j\le n,i\neq j)$;
- 对于任意 $e_i,e_j(0\le i,j\le n)$,若$(e_i,e_j)\in D$,则有 $i<j$.

对于 Ch 中的任一事件序列 $e_0e_1\ldots e_n$,按如下方法可得到相应的消息序列:如果有$\lambda(e_i)=m!$且$\lambda(e_{i+1})=m?$(其中,$m\in M,0\le i\le n$),则用 m 替换 e_ie_{i+1};对于该事件序列中的其他事件 $e_i(0\le i\le n)$,用 m 替换之(其中,$m\in M;\lambda(e_i)=m!$或$\lambda(e_i)=m?$).

注意到消息序列图中的消息对应着接口自动机中的活动,因此,称消息序列图的消息序列为从消息序列图导出的活动序列,并记为$\rho=\rho(0)\rho(1)\ldots\rho(n)$,其中,$\rho(i),0\le i\le n$ 是消息序列中的一个消息.

例 2:用户用消息序列图 SELLER(如图 2 所示)给出了一个关于例 1 中销售方构件的场景规约.SELLER 描述了销售方构件与其他构件交互时,该用户希望其具有的行为:当销售方收到 *inv_ok* 和 *cred_ok* 消息后,就应该同时发送 *shipping* 和 *billing* 消息.从 SELLER 中,我们可以导出活动序列的集合

$$L_S=\{inv\_ok\wedge cred\_ok\wedge shipping\wedge billing, cred\_ok\wedge inv\_ok\wedge shipping\wedge billing,$$
$$inv\_ok\wedge cred\_ok\wedge billing\wedge shipping, cred\_ok\wedge inv\_ok\wedge billing\wedge shipping\}.$$

2 基本概念及符号表示

定义 **9**(运行). 对于接口自动机 P 中的任意执行片段

$$\eta=v_ia_iv_{i+1}a_{i+1}\ldots a_{j-1}v_j(i<j),$$

其中: $v_i\in V_P^{init}$, $v_k\notin V_P^{init}$, $k=i+1,\ldots,j-1$,如果 $v_i=v_j$ 或 $A_P(v_j)=\varnothing$,则称η为 P 中的运行.

如果 $v_i=v_j$ 或 $A_P(v_j)=\varnothing$,则称η为 P 中的运行.

令Γ_P和Σ_P分别表示接口自动机 P 中所有执行片段组成的集合和所有运行组成的集合.显然,$\Sigma_P\in\Gamma_P$.对于任一执行片段$\eta=v_ia_iv_{i+1}a_{i+1}\ldots a_{j-1}v_j(i<j)$,称执行片段$\eta'=v_sa_sv_{s+1}a_{s+1}\ldots a_{t-1}v_t(i\le s<t\le j)$在$\eta$上,记为 $\eta'\lhd\eta$.

特别地,如果$\eta'=v_sa_sv_{s+1}(i\le s<j)$,则称迁移 $\tau=v_s\xrightarrow{a_s}_P v_{s+1}$ 在η上,记为 $\tau\lhd\eta$.

对任意$\eta\in\Gamma_P$,记η的第一个状态为 $first(\eta)$,最后一个状态为 $last(\eta)$,η中所有状态组成的集合为 $V(\eta)$.

定义 **10**(迹和投影). 由执行片段$\eta=v_0a_0v_1a_1\ldots a_{n-1}v_n$中所有活动组成的子序列称为$\eta$的迹,记为

$$trace(\eta)=a_0a_1\ldots a_{n-1}.$$

若有执行片段$\eta\in\Gamma_{P\otimes Q}$ 且 $trace(\eta)=a_0a_1\ldots a_{n-1}$,则$\eta$的迹在 P 上的投影为删除 $trace(\eta)$中所有满足条件$a_i\in A_Q\backslash shared(P,Q),0\le i\le n-1$ 的活动后得到的子序列,记作$\pi_P(trace(\eta))$.

定义 **11**(覆盖、对应迁移和对应状态). 给定两个可组合的接口自动机 P 和 Q,有$\eta=v_0a_0v_1a_1\ldots a_{n-1}v_n\in\Gamma_P$和$\alpha\in\Sigma_{P\otimes Q}$.若存在执行片段 $\zeta\lhd\alpha$ 满足如下条件:

$$\pi_P(trace(\zeta))=trace(\eta),\text{且对于任意 } v_ia_iv_{i+1}\lhd\eta \text{ 存在 } (v_i,u_i)a_i(v_{i+1},u_{i+1})\lhd\alpha,$$

其中,$u_i,u_{i+1}\in V_Q$ 且 $0\le i<n$,则称 α覆盖 η.同时称 $u_i\xrightarrow{a_i}_Q u_{i+1}$ 是 $v_i\xrightarrow{a_i}_P v_{i+1}$ 在 Q 中的对应迁移(其中,$a_i\in$

$shared(P,Q)$);称 u_i,u_{i+1} 分别为 v_i,v_{i+1} 在 Q 中的对应状态.

接口自动机 P 中的一个执行片段被 $P\otimes Q$ 中的一个运行覆盖,意味着 $P\otimes Q$ 中保留了 P 中该执行片段所代表的行为.值得注意的是:尽管 P 中的执行片段 $vav',a\notin shared(P,Q)$可被 $P\otimes Q$ 中的运行所覆盖,但在 Q 中只有 v 和 v'的对应状态,而没有 $v\xrightarrow{a}_P v'$ 的对应迁移.

定义 **12**(出现). 若活动序列ρ是接口自动机 P 中某个运行α的迹 $trace(\alpha)$的子序列,则称ρ在α上出现,并记为 $\rho\propto\alpha$.

我们可以认为活动序列和运行都表示的是行为.那么,一个活动序列在接口自动机的某个运行上出现,就意味着该接口自动机的某个行为包含了那个活动序列所表示的行为.

给定接口自动机 P 和一个活动序列的集合 L,函数 $\phi_L:2^{\Sigma_P}\to 2^{\Sigma_P}$ 可将任一集合$\Sigma\in\Sigma_P$ 划分成两个集合:

$$\phi_L(\Sigma)=\{\alpha\in\Sigma\mid\exists\rho\in L.\rho\propto\alpha\},\ \overline{\phi_L(\Sigma)}=\Sigma\setminus\phi_L(\Sigma).$$

于是,Σ_P 可被划分成$\phi_L(\Sigma_P)$和 $\overline{\phi_L(\Sigma_P)}$.对于集合$\phi_L(\Sigma_P)$中的每一个运行,至少有 L 中的一个活动序列在其上出现;而对于集合 $\overline{\phi_L(\Sigma_P)}$ 中的任何一个运行,都没有 L 中的任何活动序列在其上出现.

定义 **13**(极大包含环境). 有接口自动机 R 和活动序列集合 L,且 L 中有活动序列在 R 的运行上出现.若 R 的环境 E 满足条件:对任意$\rho\in L$,若ρ出现在 R 中的运行α上,那么α一定被 $R\otimes E$ 中的运行所覆盖,则称 E 为在 L 下 R 的一个包含环境.在 L 下 R 的一个包含环境 E 是在 L 下 R 的极大包含环境,当且仅当不存在 L 下 R 的任何其他包含环境 E',使得被 $R\otimes E$ 中运行所覆盖的 R 中的某个执行片段不被 $R\otimes E'$中任何运行所覆盖.记在 L 下 R 的极大包含环境为 E_L.

给定接口自动机R和活动序列集合L,如果我们将L看作是一组行为,那么,R中有两类行为会被保留到$R\otimes E$中,其中,E是在 L 下 R 的一个包含环境.第一类,R 中所有包含 L 中行为的行为;第二类,R 中所有不包含 L 中行为的行为.与其他包含环境相比,在 $R\otimes E_L$ 中保留的 R 中的第二类行为最少.

定理 **1**. 对于任意的接口自动机 R 和活动序列集合 L,存在 L 下 R 的极大包含环境,当且仅当存在 L 下 R 的包含环境.

3 极大包含环境的构造

一个构件 COMP 的行为可以用一个接口自动机 R 来描述.用户希望 COMP 具有的行为可以用 MSC 书写的场景规约来给出.这样,从 COMP 中抽取场景规约中描述的行为,就相当于在活动序列集合 L 下为 R 构造极大包含环境 E_L,其中,L 是从用 MSC 书写的场景规约中导出的活动序列集合.当 E_L 构造完成后,$R\otimes E_L$ 中就会保留从 R 中抽取出来的场景规约中给出的用户想要的所有行为,而 R 中的其他行为则尽可能地不被保留在 $R\otimes E_L$ 中.

为了构造 L 下 R 的极大包含环境 E_L,首先需要判定 E_L 是否存在.由定理 1 可知,这等价于判定 L 下 R 的包含环境是否存在.

本节将详细讨论 L 下 R 的包含环境存在性的判定、极大包含环境的构造方法,并给出具体的构造算法.

3.1 包含环境存在性的判定

由定义 13 可知,如果 L 中的所有活动序列在 R 中的任何运行上都不出现,那么就不存在 L 下 R 的包含环境.此外,我们还证明:对于任意接口自动机 P 来说,如果其中存在了满足一定形式的执行片段,那么,对于 P 的任何环境 E 而言,这种执行片段均无法被 $P\otimes E$ 中的任何运行所覆盖[7].可以设想:如果 R 中某个有 L 中的活动序列出现的运行中包含了这种形式的执行片段,那么,由于这个运行无法被 R 与其环境的乘积中的任何运行所覆盖,由定义 13 可知,此时,L 下 R 的包含环境也不存在.下面以定理形式给出判定包含环境存在性的方法.

定理 **2**. 对任意的接口自动机 R 和活动序列集合 L,如果下列条件满足其一,则不存在 L 下 R 的包含环境 E:

(1) 对任意的$\rho\in L$ 和$\alpha\in\Sigma_R$,ρ均不在α上出现;

(2) 存在$\rho\in L$ 且$\rho\propto\alpha$(其中,$\alpha\in\Sigma_R$),有 $\eta_1,\eta_2\in\Gamma_R$, $\eta_2\lhd\alpha$ 且η_1,η_2 满足下列条件之一:

a) $\eta_1=v_iav_j$ 且 $\eta_2=v_jbv_k$,其中:$i\neq j\neq k$; $a\in A_R^H$; $b\in A_R^I\cap shared(R,E)$ 且 $b\notin A_R(v_i)$;

b) $\eta_1=v_iav_j$ 且 $\eta_2=v_ibv_k$,其中:$i\neq j\neq k$; $a\in A_R^H$; $b\in A_R^I\cap shared(R,E)$ 且 $b\notin A_R(v_j)$.

c) $\eta_1=v_ia_iv_{i+1}a_{i+1}\ldots a_{j-1}v_j$ 且 $\eta_2=v_ibv_i'$,其中:$i<j$; $v_i'\notin V(\eta_1)$;$a_k\notin shared(R,E)$;$k=i,i+1,\ldots,j-1$; $b\in A_R^I\cap shared(R,E)$ 且存在 $v\in V(\eta_1)$使得 $b\notin A_R(v)$.

根据定理 1 可得到结论:如果在 L 下 R 的包含环境不存在,则在 L 下 R 的极大包含环境也不存在.

3.2 极大包含环境的构造算法

对于给定的接口自动机 R 和活动序列集合 L,构造 L 下 R 的极大包含环境 E_L 可分两步进行:首先需要确定 R 中的哪些行为保留在了 $R\otimes E_L$ 中,即 R 中的哪些运行和执行片段被 $R\otimes E_L$ 中的运行所覆盖;其次,在一个空的接口自动机 E 中,为 R 中的这些运行和执行片段上的每一个迁移构造其对应迁移.当对应迁移全部构造完成后,E 就是我们要得到的极大包含环境 E_L.对于第二步,在文献[7]中我们已经给出了一种方法来为一个接口自动机中的迁移在另一接口自动机中构造其对应迁移.因此,问题的关键就是确定 R 中的哪些运行和执行片段被 $R\otimes E_L$ 中的运行所覆盖.经分析可知,该问题等价于确定 R 中的哪些迁移在 E_L 中有其对应迁移.下面以定理形式来回答该问题.

定理 **3**. 接口自动机 R 的环境 E 是在活动序列集合 L 下 R 的极大包含环境,当且仅当 E 具有下述所有性质:

(1) 对任意的迁移 $\tau\triangleleft\alpha\in\phi_L(\Sigma_R)$,

若 $label(\tau)\in shared(R,E)$,则 E 中存在 τ的对应迁移;

若 $label(\tau)\notin shared(R,E)$,则 E 中存在 $head(\tau)$和 $tail(\tau)$的对应状态.

(2) 对任意的迁移 $\tau\triangleleft\eta\in\Gamma_R$, $\eta\triangleleft\alpha\in\overline{\phi_L(\Sigma_R)}$,其中,执行片段 η满足条件 $first(\eta)\in V_R^{init}$ 和对任意 $\tau'\triangleleft\eta$ 且 $\tau'\in T_R^I$,存在$\beta\in\phi_L(\Sigma_R)$,使得 $\tau'\triangleleft\beta$,

若 $label(\tau)\in shared(R,E)$,则 E 中存在 τ的对应迁移;

若 $label(\tau)\notin shared(R,E)$,则 E 中存在 $head(\tau)$和 $tail(\tau)$的对应状态.

(3) 对于除(1),(2)以外的 R 中的所有其他迁移 τ,

若 $label(\tau)\in shared(R,E)$,则 E 中不存在 τ的对应迁移;

若 $label(\tau)\notin shared(R,E)$,则 E 中不存在 $head(\tau)$和 $tail(\tau)$的对应状态.

定理 3 描述了在已知活动序列集合下任意接口自动机的极大包含环境所具有的性质.同时,定理 3 也告诉我们:对于接口自动机 R 和活动序列集合 L 下 R 的极大包含环境 E_L 而言,R 中的哪些执行片段可被 $R\otimes E_L$ 中的运行所覆盖,以及 R 中的哪些执行片段不被 $R\otimes E_L$ 中的运行所覆盖.注意到定理 3 中所列出的性质是充分必要的,因此,该定理还可用于判断 R 的任一环境 E 是否为 L 下 R 的极大包含环境.

3.2.1 构造极大包含环境的基本方法

由定理 3 可知,$\phi_L(\Sigma_R)$中的所有运行均被 $R\otimes E_L$ 中的运行所覆盖.但 $\overline{\phi_L(\Sigma_R)}$ 中的运行被覆盖的情况不很明确.按照定义 13,我们希望 $\overline{\phi_L(\Sigma_R)}$ 中的运行全部不被 $R\otimes E_L$ 中的运行覆盖才好.但是,这一点并不总能做到.由定义 6 可知,R 的环境必须接受 R 的所有输出活动,因此,E_L 只有通过不为 R 提供某个输入活动,即在 E_L 中不为 R 的某个输入迁移构造其对应迁移,才能使得 R 中该迁移后面的执行片段不被 $R\otimes E_L$ 中的运行所覆盖.此外, $\overline{\phi_L(\Sigma_R)}$ 中的运行与 $\phi_L(\Sigma_R)$中的运行可能存在公共部分,即可能存在某执行片段,它既在 $\overline{\phi_L(\Sigma_R)}$ 中的运行上,又在 $\phi_L(\Sigma_R)$中的运行上.因此,我们只能对那些只存在于 $\overline{\phi_L(\Sigma_R)}$ 中的运行上而不存在于$\phi_L(\Sigma_R)$中的运行上的输入迁移,不在 E_L 中为其构造对应迁移.

基于以上分析,可按下述方法得到 R 的极大包含环境 E_L.首先,对 $\overline{\phi_L(\Sigma_R)}$ 中的每个运行α,从状态 $first(\alpha)$开始,沿α对其上的迁移进行遍历.同时,找到α上第一个不在$\phi_L(\Sigma_R)$中任何运行上的输入迁移.然后,将所有这些输入迁移从 R 中删除,并删除由此产生的所有不可达状态;其次,在一个空的接口自动机 E 中为 R 中剩下的迁移构造对应迁移.当上述过程完成后,E 便是在 L 下 R 的极大包含环境 E_L.

如果 $first(\eta)=last(\eta)$且 $first(\eta),last(\eta)\notin V_R^{init}$,我们则称执行片段$\eta\in\Gamma_R$ 是一个环.特别注意到:如果接口自动

机 R 中存在环,则Σ_R 就是无穷集合.同时,Σ_R 中某些运行的长度,即该运行上迁移的个数也是无穷的.进而,$\phi_L(\Sigma_R)$, $\overline{\phi_L(\Sigma_R)}$ 和它们中某些运行的长度都是无穷的.因此,直接对 $\overline{\phi_L(\Sigma_R)}$ 中的运行进行遍历是不可行的.为了得到一个可行的办法,我们特别地引入简单运行和简单环的概念.

设活动序列$\rho=\rho(0)\rho(1)\ldots\rho(m)\in L$ 出现在运行$\alpha\in\phi_L(\Sigma_R)$上.对于执行片段 $\eta\lhd\alpha$,若ρ是 $trace(\eta)=a_0a_1\ldots a_n$ ($n\geq m$)的子序列且$\rho(0)=a_0$,$\rho(m)=a_n$,则称执行片段η是ρ在α上的真出现.设 $\eta_0,\eta_1,\ldots,\eta_n\lhd\alpha$ 分别是$\rho_0\rho_1\ldots\rho_n\in L$ 在α上的真出现.对执行片段 $\eta\lhd\alpha$, 若 $first(\eta)\notin V(\eta_i)$且 $last(\eta)\notin V(\eta_i)$,$i$=0,1,…,$n$,则称执行片段$\eta$是$\rho_0\rho_1\ldots\rho_n$ 在α上的真非出现.

定义 **14**(简单运行). 给定接口自动机 R 和活动序列集合 L,运行$\alpha=v_0a_0v_1a_1\ldots a_{n-1}v_n\in\Sigma_R$,若满足如下条件,则称其为简单运行:

(1) 如果 $\alpha\in\overline{\phi_L(\Sigma_R)}$,则有 $v_i\neq v_j(i\neq j,0<i,j<n)$;

(2) 如果$\alpha\in\phi_L(\Sigma_R)$,则

a) 对于α上的任何 L 中活动序列的真非出现$\eta=v_ia_iv_{i+1}a_{i+1}\ldots a_{j-1}v_j(0\leq i<j\leq n)$,有 $v_s\neq v_t(s\neq t,i\leq s,t\leq j)$,并且

b) 对于α上的活动序列$\rho=\rho(0)\rho(1)\ldots\rho(m)\in L$ 的真出现ς,如果存在 $\varsigma'=v_ia_iv_{i+1}a_{i+1}...a_{j-1}v_j\lhd\varsigma$ 并且 $a_i=\rho(k),a_{j-1}=\rho(k+1),0\leq k<m$,则有 $v_s\neq v_t(s\neq t,i<s,t<j)$.

上述定义中的第 1 个条件说明在没有 L 中活动序列出现的简单运行中不包含任何环;第 2 个条件说明简单运行上的 L 中活动序列的真非出现中不包含任何环;而在 L 中活动序列的真出现中,活动序列中相邻两个活动的出现之间不包含任何环.

在 L 下接口自动机 R 中的所有简单运行构成的集合记为 Ω_R^L .同理, Ω_R^L 可划分成 $\phi_L(\Omega_R^L)$ 和 $\overline{\phi_L(\Omega_R^L)}$ 两个集合.显然有

$$\Omega_R^L\subseteq\Sigma_R,\ \phi_L(\Omega_R^L)\subseteq\phi_L(\Sigma_R),\ \overline{\phi_L(\Omega_R^L)}\subseteq\overline{\phi_L(\Sigma_R)}.$$

注意到, $\phi_L(\Omega_R^L)$ 和 $\overline{\phi_L(\Omega_R^L)}$ 均是有穷集合,而且所有简单运行的长度也是有穷的.

定义 **15**(简单环). 给定接口自动机 R 和活动序列集合 L,如果执行片段$\eta=v_ia_iv_{i+1}a_{i+1}\ldots a_{j-1}v_j\in\Gamma_R(i<j)$满足条件:(1) $v_i=v_j$, $v_i,v_j\notin V_R^{init}$;(2) $v_s\neq v_t(s\neq t,i\leq s,t<j)$;(3) η不在 $\phi_L(\Omega_R^L)$ 中的任何简单运行上,则称η为简单环.

在上述定义中,条件(1)和条件(2)确保了在简单环中除第 1 个和最后一个状态外,再没有相同的状态.条件(3)确保了简单环不是 L 中某个活动序列的真出现中的环.

在 L 下接口自动机 R 中的所有简单环构成的集合记为 Λ_R^L .我们称简单环 $\eta\in\Lambda_R^L$ 与简单运行 $\alpha\in\Omega_R^L$ 相关联,如果 $V(\eta)\cap V(\alpha)\neq\varnothing$或 $V(\eta)\cap V(\eta')\neq\varnothing$,其中,$\eta'$是与$\alpha$相关联的简单环.令 $\psi_L(\Lambda_R^L)=\{\eta\in\Lambda_R^L\mid\exists\alpha\in\phi_L(\Omega_R^L)\}$.η与α相关联$\}$, $\overline{\psi}_L(\Lambda_R^L)=\{\eta\in\Lambda_R^L\mid\exists\alpha\in\overline{\phi_L(\Omega_R^L)}\}$.η与α相关联$\}$.注意到, $\psi_L(\Lambda_R^L)$ 和 $\overline{\psi}_L(\Lambda_R^L)$ 均是有穷集合,而且所有简单环的长度也是有穷的.

3.2.2 构造算法

通过引入简单运行和简单环的概念,将无穷集合Σ_R转换成了等价的有穷集合 Ω_R^L 和 Λ_R^L .这样便可将第 3.2.1 节中所述方法应用到 Ω_R^L 和 Λ_R^L 上,从而使该方法成为可行的.

构造极大包含环境的算法框架如下:首先根据定理 2 判定在 L 下 R 的包含环境是否存在.如果不存在,则在 L 下 R 的极大包含环境也不存在;其次,如果在 L 下 R 的包含环境存在,我们可以通过下述 3 个步骤来获得 L 下 R 的极大包含环境:第 1 步,对每个没有 L 中活动序列出现的简单运行以及与其关联的简单环上的迁移进行遍历,并分别找到其上的第 1 个不在任何有 L 中活动序列出现的简单运行以及与其关联的简单环上的输入迁移;第 2 步,从 R 中删除在上一步中找到的所有输入迁移,同时删除由此产生的所有不可达状态;第 3 步,为 R 中剩下的所有迁移在一个空的接口自动机中构造对应迁移.

令 $R\downarrow T$ 表示从接口自动机 R 中删除 $T\subset T_R$ 中的迁移及由此产生的所有不可达状态后所得到的新的接口自动机.约定 $A_{E_L}^H=\varnothing$ 和 $A_{E_L}^O=A_R^I$ [7].构造活动序列集合 L 下接口自动机 R 的极大包含环境 E_L 的算法见算法 1.

算法 **1**. 极大包含环境构造算法.

输入:接口自动机 R 和活动序列集合 L..

输出:在活动序列集合 L.下接口自动机 R 的极大包含环境 E_L.

1 遍历 R 得到集合 $\phi_L(\Omega_R^L)$, $\overline{\phi_L(\Omega_R^L)}$, $\psi_L(\Lambda_R^L)$ 和 $\overline{\psi}_L(\Lambda_R^L)$

2 **if** L 下 R 的包含环境不存在 **then return** E_L 不存在

3 **else**

4 T:=∅;

5 **for** 每一个 $\eta \in \overline{\phi_L(\Omega_R^L)} \cup \overline{\psi}_L(\Lambda_R^L)$ **do**

6 found:=**true**;

7 τ:=η上的第 1 个迁移; // $head(\tau) = first(\eta) \wedge \tau \lhd \eta$

8 **while** $(\tau \notin T_R^I \vee \exists \varsigma \in \phi_L(\Omega_R^L) \cup \psi_L(\Lambda_R^L).\tau \lhd \varsigma) \wedge$ found **do**

9 **if** τ不是η上的最后一个迁移 **then** // $\neg(tail(\tau) = last(\eta) \wedge \tau \lhd \eta)$

10 τ:=η上的下一个迁移

11 **else** found:=**false**

12 **endwhile**

13 **if** found **then** T:=$T \cup \{\tau\}$;

14 **endfor**

15 R':=$R \downarrow T$;

16 $V_{E_L} := V_{E_L}^{init} := \{u_0\}$; //初始化 E_L

17 **for** 每一个$\tau \in T_{R'}$ **do** 在 E_L 中构造τ的对应迁移;

18 **return** E_L

19 **endif**

3.2.3 算法分析

可以证明算法 1 的输出结果是在 L 下 R 的一个包含环境.同时,该包含环境还具有定理 3 中的全部性质.也就是说,算法 1 的输出结果是在 L 下 R 的极大包含环境.因此,算法 1 是正确的.

关于算法 1 中的第 1 行,在文献[6]中我们已给出了相关算法,以证明可以在已知接口自动机中找出具有某个活动序列出现的所有运行.关于算法 1 中的第 17 行,在文献[7]中我们也给出了构造对应迁移的具体方法.

设 R 中所有简单运行和所有简单环的最大长度为 $n = \max\{length(\eta) \mid \eta \in \Omega_R^L \cup \Lambda_R^L\}$,其中 $length(\eta)$为η上的迁移个数.设 $\phi_L(\Omega_R^L)$ 和 $\overline{\phi_L(\Omega_R^L)}$ 中简单运行的个数分别为 $m_1 = \left|\phi_L(\Omega_R^L)\right|$ 和 $m_2 = \left|\overline{\phi_L(\Omega_R^L)}\right|$.设 $\psi_L(\Lambda_R^L)$ 和 $\overline{\psi}_L(\Lambda_R^L)$ 中简单环的个数分别为 $k_1 = \left|\psi_L(\Lambda_R^L)\right|$ 和 $k_2 = \left|\overline{\psi}_L(\Lambda_R^L)\right|$.在最坏情况下,算法 1 中第 5 行~第 14 行的时间复杂度为 $O((m_1+k_1)(m_2+k_2)n^2)$.由文献[6,7]可知,算法 1 中第 1 行和第 17 行的时间复杂度分别为 $O((m_1+m_2)n)$和 $O(|V_{R'}|)$,其中,$|V_{R'}|$为接口自动机 R'中的状态数.通常地,对于 $\eta \in \Lambda_R^L$ 和 $\alpha \in \Omega_R^L$ 有 $length(\eta) << length(\alpha)$和$|V_{R'}| << (m_1+m_2)n$.因此,算法 1 的时间复杂度为 $O(m_1 m_2 n^2)$.

例 3:考察电子商务系统中销售方构件的实例(参见例 1 和例 2).按照算法 1,我们可以得到在由消息序列图 SELLER(如图 2 所示)导出的活动序列集合 L_S 下接口自动机 *Seller*(如图 1 所示))的极大包含环境 E_{L_S} (如图 3 所示).作为算法 1 的中间结果的接口自动机 R'(见算法 1 中第 15 行)如图 4 所示.*Seller* 和 E_{L_S} 的组合 $Seller \otimes E_{L_S}$ 如图 5 所示.从中我们可以看出:由消息序列图 SELLER 所描述的用户想要的所有行为全部从 *Seller* 中抽取出来并保留到了 $Seller \otimes E_{L_S}$ 中.而 *Seller* 中的其他行为在 $Seller \otimes E_{L_S}$ 中已被尽可能地丢弃掉了.由该实例可见,本文所给出的基于场景规约的构件行为抽取方法是有效的.

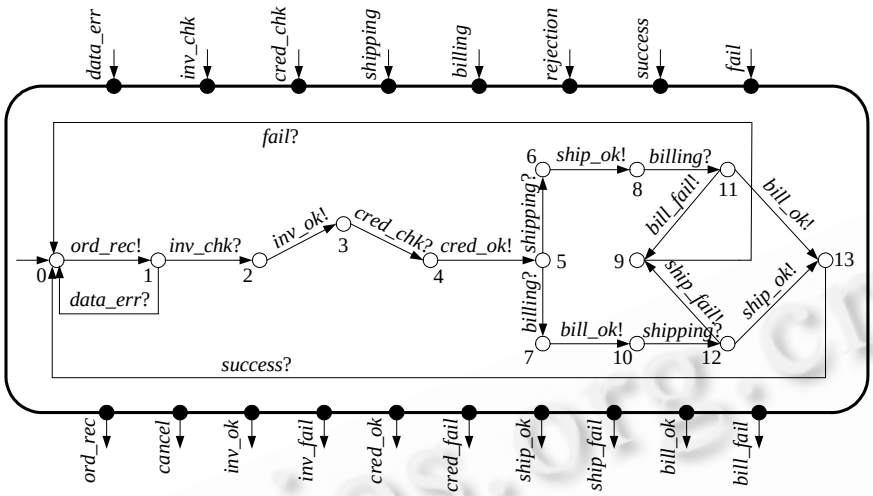

Fig.3 The SIE E_{L_S} of *Seller* under L_S

图 3 在 L_S 下 *Seller* 的极大包含环境 E_{L_S}

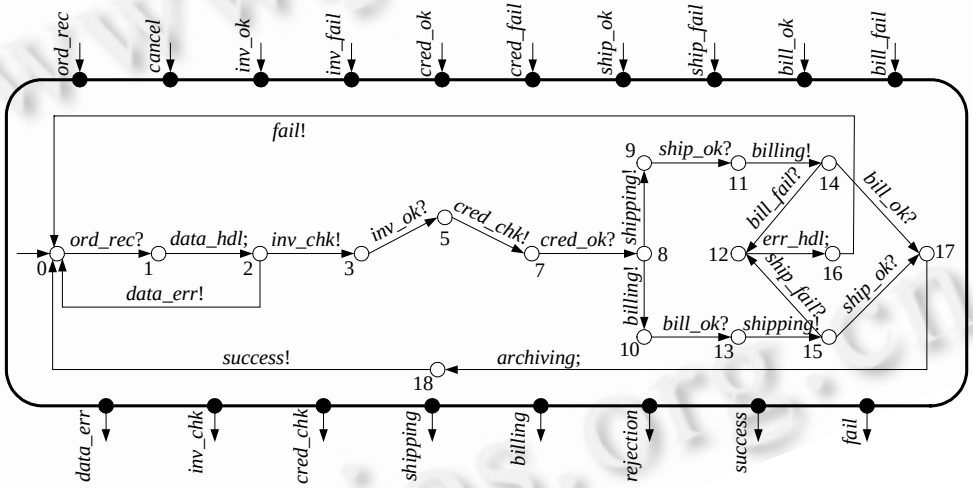

Fig.4 IA R', the intermediate result of algorithm 1 with inputs of *Seller* and L_S

图 4 接口自动机 R'.它是以 *Seller* 和 L_S 作为输入时算法 1 的中间结果

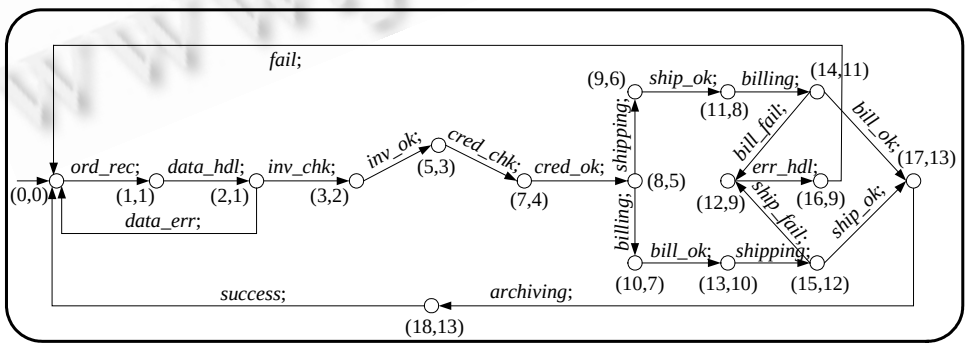

Fig.5 *Seller* $\otimes E_{L_S}$, the composition of *Seller* and E_{L_S}

图 5 *Seller* 和 E_{L_S} 的组合,即 *Seller* $\otimes E_{L_S}$

4 相关工作

基于构件的软件开发重点是解决构件获取和构件组合这两个问题.这两个问题均应在用户需求的指导下来完成.也就是说,无论是构件获取还是构件组合,最终得到的构件的功能应与用户的需求相一致.然而,现实中想做到这一点是很困难的.因此,人们从不同角度对这一问题进行了研究并提出许多灵活的方法来获取与用户需求最接近的构件(如果无法获得与用户需求完全吻合的构件)[8−10].特别地,Atkinson 从行为的角度对构件获取进行了研究[11],并给出一种方法可以找到行为上与用户需求最为接近的构件.但是,上述种种方法均忽略了一个问题,那就是如何处理找到的最接近用户需求的构件中冗余的那部分行为.而这些行为往往成为干扰用户正确使用这些构件的最大障碍.

在文献[2,3]中,作者主要解决了构件组合中的行为兼容问题,即构件间交互顺序的匹配问题.同样,他们也忽略了对构件中冗余行为的处理.如果是构件中冗余的行为影响了构件的兼容,或是在组合后的构件中含有冗余的行为,那么,他们提出的方法就不适用了.

本文给出的方法通过对环境的使用,根据用户需求从构件或构件的组合(因为构件的组合依然可以被看作是一个构件)中抽取用户想要的行为.这样就弥补了上述研究工作的不足.

在我们以前的工作中,仅对如何在已知构件中查找是否具有用户想要的行为进行了研究[6].本文则进一步研究了当已知构件中确实存在用户想要的行为时,如何从中抽取这部分行为.在文献[7]中,我们利用环境来解决构件行为兼容的问题,而没有考虑组合后的行为是否与用户需求相一致.在本文中,我们利用环境来按照用户需求从构件中抽取用户想要的行为.

5 总结及展望

本文研究了场景驱动的构件行为抽取.通过为构件构造一个特殊的环境,即极大包含环境,可以从构件中抽取出场景规约中所描述的用户想要的行为.我们使用接口自动机为构件的行为建模.用消息序列图描述场景规约并将消息序列图抽象为一组活动序列,并用接口自动机的乘积来刻画构件的组合.我们给出了在已知活动序列集合下构造给定接口自动机的极大包含环境的算法,并用实例对文中所述的方法进行了说明.

场景驱动的构件行为抽取方法可以提高 CBSD 中构件的复用率.利用该方法可以使构件库中那些由于含有冗余行为而原本无法在特定场景下使用的构件变为可以在该场景下使用的构件.由于极大包含环境的行为模型可以通过算法自动构造,从而摆脱了人工分析所带来的巨大工作量,因此也提高了开发效率.

在以服务为基本组成元素而构建起来的软件系统中(如 Web Services),服务与构件有许多共同特征,如模块性、可组合性和可复用性等.而且,服务也可以用构件技术来实现.因此,本文所述方法同样适用于服务.

本文只研究了如何从构件中抽取用户想要的行为.对于下一步工作,我们希望利用环境将用户不想要的行为从构件中过滤掉.

References:

[1] Allen R, Garlan D. A formal basis for architectural connection. ACM Trans. on Software Engineering and Methodology, 1997,6(3): 213−249.

[2] Bracciali A, Brogi A, Canal C. A formal approach to component adaptation. Journal of Systems and Software, 2005,74(1):45−54.

[3] Yellin DM, Strom RE. Protocol specifications and component adaptors. ACM Trans. on Programming Languages and Systems, 1997,19(2):292−333.

[4] de Alfaro L, Henzinger TA. Interface automata. In: Wermelinger M, Gall H, eds. Proc. of the 9th Annual ACM Symp. on Foundations of Software Engineering (FSE 2001). New York: ACM Press, 2001. 109−120.

[5] ITU-T. ITU-T recommendation Z.120: Message Sequence Chart (MSC). Geneva: ITU-T, 1999.

[6] Hu J, Yu XF, Zhang Y, Wang LZ, Li XD, Zheng GL. Checking component-based designs for scenario-based specification. Chinese Journal of Computers, 2006,29(4):513−525 (in Chinese with English abstract).

[7] Zhang Y, Hu J, Yu XF, Zhang T, Li XD, Zheng GL. Deriving available behavior all out from incompatible component compositions. In: Liu Z, Barbosa L, eds. Proc. of the 2nd Int'l Workshop on Formal Aspects of Component Software (FACS 2005). ENTCS 160. Netherlands: Elsevier, 2006. 349−361. http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B75H1-4KKPCPF-T-1&_cdi=13109&_user=2316225&_orig=browse&_coverDate

[8] Mili R, Mili A, Mittermeir RT. Storing and retrieving software components: A refinement based system. IEEE Trans. on Software Engineering, 1997,23(7):445−460.

[9] Sugumaran V, Storey VC. A semantic-based approach to component retrieval. ACM SIGMIS Database, 2003,34(3):8−24.

[10] Tansalarak N, Claypool K. Finding a needle in the haystack: A technique for ranking matches between components. In: Heineman G, Crnkovic I, Schmidt HW, Stafford JA, Szyperski CA, Wallnau KC, eds. Proc. of the 8th Int'l Symp. on Component-Based Software Engineering (CBSE 2005). LNCS 3489, Berlin, Heidelberg, New York: Springer-Verlag, 2005. 171−186.

[11] Atkinson S. Examining behavioural retrieval. In: Proc. of the 8th Annual Workshop on Institutionalizing Software Reuse (WISR8). 1997. http://www.umcs.maine.edu/~ftp/wisr/wisr8/papers/atkinson/atkinson.html

附中文参考文献:

[6] 胡军,于笑丰,张岩,王林章,李宣东,郑国梁.基于场景规约的构件式系统设计分析与验证.计算机学报,2006,29(4):513−525.

张岩(1974－),男,山东武城人,博士生,主要研究领域为软件工程,形式化方法,面向服务的计算,软件度量.

胡军(1973－),男,博士,主要研究领域为软件工程,形式化方法,嵌入式软件建模.

于笑丰(1976－),男,博士生,主要研究领域为软件工程,模型驱动开发.

张天(1978－),男,博士生,主要研究领域为软件工程,UML,模型转换.

李宣东(1963－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为软件工程,形式化方法,模型检验.

郑国梁(1936－),男,教授,博士生导师,主要研究领域为软件工程,软件开发环境.