基于有限状态自动机的服务组合模型

蒋运承[1,2],汤 庸[2],邓培民[1]
(1. 广西师范大学 计算机科学与信息工程学院,广西 桂林 541004; 2. 中山大学 计算机科学系,广东 广州 510275)

摘 要:分析了目前服务计算的研究现状和存在的问题,在 D Berardi 和 A Wombacher 的基础上提出了一种带条件的有限状态自动机模型 cFSA(Finite State Automata with condition),并给出了基于 cFSA 的服务理论模型. 在该服务理论模型的基础上提出了一种基于有限状态自动机的服务组合形式化模型,并给出了该模型的代数性质和实现方法.

关键词:有限状态自动机;带条件的有限状态自动机;服务计算;服务组合
中图分类号:TP18 **文献标识码**:A **文章编号**:1673-4785(2006)02-0048-10

A service composition model based on finite state automata

JIANG Yun-cheng[1,2], TANG Yong[2], DENG Pei-min[1]
(1. College of Computer Sciences and Information Engineering, Guangxi Normal University, Guilin 541004, China; 2. Department of Computer Sciences, Sun Ya-sen University, Guangzhou 510275, China)

Abstract: The existing conditions and problems of service computing are analyzed, and based on the work of D Berardi and A Wombacher, a kind of finite state automata with condition cFSA (Finite State Automata with condition) is presented, and the service theory model based on cFSA is presented too. Based on this model, the formal theory model of service composition based on finite state automata with condition cFSA is studied. The algebraic property and implementing method of the service composition model are studied.

Keywords: finite state automata; finite state automata with condition; service computing; service composition

面向服务的计算[1] (service-oriented computing)是一种新的分布式计算范型,它以服务作为开发应用或提供问题解决方案的基本元素. 而服务是一种自描述的、平台无关的计算元素,它支持分布式应用的快速和低价组合. 例如,目前的 Web 服务[2]、语义 Web 服务[3]或基于主体的服务[4]等都是面向服务的计算范型,文中统称为服务或 e-Service.

目前国内外对 e-Service 进行了深入研究,W3C 提出了 Web 服务描述语言 WSDL[5],D Martin 基于 OWL 提出了语义 Web 服务本体 OWL-S[6],史忠植利用描述逻辑理论来描述主体服务[7],蒋运承提出了主体服务描述语言 SDLSIN[4],并且这些服务描述语言都有相应的服务匹配算法. IBM 根据工作流的思想提出了 Web 服务流语言 WSFL[8],Microsoft 提出了服务组合语言 XLANG[9],IBM 和 Microsoft 将 WSFL 和 XLANG 结合起来提出了服务组合语言 BPEL4WS[10],蒋运承研究了面向服务质量的服务匹配算法[11]等.

但是,目前 e-Service 的研究中,主要将服务看成是一个无状态的动作,事实上这是不充分的,服务应该是一个有状态的动作序列[12-13],这方面已有研究. 例如,D. Berardi 用有限状态自动机来建立 Web 服务的理论模型,但 D. Berardi 的工作只研究了服务描述与有限状态自动机的转换问题,没有研究基于有限状态自动机的服务匹配和组合问题[12]. A Wombacher 研究了基于有限状态自动机的 Web 服务商业过程的匹配问题,但 A Wombacher 也没有研究基于有限状态自动机的服务组合问题[13]. 本文进一步研究了基于有限状态自动机的服务组合问题.

1 服务理论模型

为了用有限状态自动机来描述服务,下面先给

收稿日期:2006-02-23.
基金项目:国家自然科学基金资助项目(60373081);广东省自然科学重点基金资助项目(04105503).

出一些基本概念.

定义 1(有限状态自动机)　有限状态自动机(finite state automata, FSA)是一个五元组$(Q, \sum, \delta, q_0, F)$,其中 Q 是一个有穷集合,叫做状态集;$\sum$ 是一个有穷集合,叫做字母表;$\delta: Q \times \sum \to Q$ 是转移函数;$q_0 \in Q$ 是起始状态;$F \subseteq Q$ 是接收状态集.

e-Service 可以看成是一个软件"黑箱",它可以与客户交互,包括 3 步:1) 客户通过发送输入命令(input_command)调用 e-Service;2) 输入命令激发 e-Service 的内部计算(internal_computation);3) 客户得到 e-Service 的输出消息(output_message).因此,e-Service 交互可以用 3 元组 < input_command, internal_computation, output_message > 表示.由于采用"黑箱"的方法,交互可用二元组 < input_command, output_message > 来表示.

因而可用有限状态自动机来描述 e-Service,即 e-Service 的每次交互可以看成是有限状态自动机的状态转换,并且交互可用二元组 < input_command, output_message > 来表示.

定义 2(服务弱形式化模型)[12]　给定一个服务 e-Service,则 e-Service 可以表示为一个有限状态自动机 FSA,并且 FSA 是一个六元组$(Q, I, O, \delta, q_0, F)$,其中 Q 是一个有穷集合,是 FSA 的状态集,其中状态表示客户与 e-Service 交互序列中的历史记录;I 是输入命令 input_command 的有穷集合;O 是输出消息 output_message 的有穷集合,$I \times O$ 是 FSA 的字母表;$\delta: Q \times I \times O \to Q$ 是转移函数,即给定一个状态,根据 < input_command, output_message >,FSA 能够转移到另一个状态;$q_0 \in Q$ 是起始状态;$F \subseteq Q$ 是接收状态集,即用户能够与 e-Service 结束交互的状态集.

图 1 是一个基于有限状态自动机的股票交易服务[12].

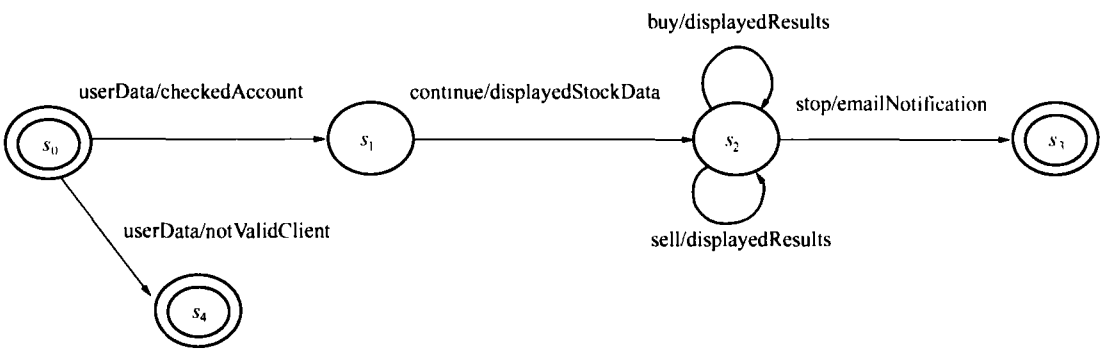


图 1　股票交易服务

Fig. 1　Stock service

股票交易服务的形式化模型可以定义为

FSA = $(Q, I, O, \delta, s_0, F)$,其中状态集 $Q = \{s_0, s_1, s_2, s_3, s_4\}$;输入命令集合 $I = \{$userData, continue, buy, sell, stop$\}$,输出消息集合 $O = \{$checkedAccount, notValidClient, displayedStockData, displayedResults, emailNotification$\}$,字母表 $I \times O = \{$userData/checkedAccount, userData/notValidClient, continue/displayedStockData, buy/displayedResults, sell/displayedResults, stop/emailNotification$\}$;转移函数 δ 为:$s_0 \times$userData/checkedAccount $\to s_1$, $s_0 \times$ userData/notValidClient $\to s_4$, $s_1 \times$continue/displayedStockData $\to s_2$, $s_2 \times$buy/displayedResults $\to s_2$, $s_2 \times$sell/displayedResults $\to s_2$, $s_2 \times$stop/emailNotification $\to s_3$;$s_0 \in Q$ 是起始状态;接收状态集 $F = \{s_3, s_4\}$.

服务组合是根据组合算子来实现的,而有些组合算子(如 If-Then-Else 算子)涉及到条件,因而需要一种新的自动机的定义.

定义 3(命题逻辑公式)　命题逻辑公式按下列规则生成:

1) 常量 true 和 false 是命题逻辑公式;

2) 命题变元是命题逻辑公式;

3) 如果 ϕ 是命题逻辑公式,则 $\neg\phi$ 也是命题逻辑公式;

4) 如果 ϕ 和 μ 是命题逻辑公式,则 $\phi \wedge \mu$ 和 $\phi \vee \mu$ 也是命题逻辑公式;

5) 命题逻辑公式只能由上述规则生成.

定义 4(条件)　条件是由状态和命题逻辑公式通过联结词组成的表达式,形式定义如下:

假设 Q 是有限状态自动机状态的集合,E 是命题逻辑公式的集合,or 和 and 是条件联结词,则条件按下列规则生成:

1) p、$q \in Q$,则 p or q 和 p and q 是条件;

2) $p \in Q$, $e \in E$,则 p and e 是条件(如果 e = true,则简写为 p);

3) 如果 c_1 和 c_2 是条件,则 c_1 and c_2 和 c_1 or c_2

也是条件;

4)条件只能由上述规则生成.

条件的语义解释是:条件 p or q 表示状态 p 和 q 只选择一个;条件 p and q 表示状态 p, q 都选择;条件 p and e 表示当 e 的值为真时才选择状态 p;对于复杂条件 c_1 and c_2 和 c_1 or c_2 类似进行解释.

定义5(带条件的有限状态自动机) 带条件的有限状态自动机(finite state automata with condition, cFSA)是一个六元组 $(Q, \sum, \delta, q_0, F, QC)$,其中 Q 是一个有穷集合,叫做状态集; $\sum$ 是一个有穷集合,叫做字母表; $\delta: Q \times \sum \times QC \to Q$ 是转移函数; $q_0 \in Q$ 是起始状态; $F \subseteq Q$ 是接收状态集; $QC: Q \times C$ 是状态的条件集.

定义6(服务形式化模型) 给定一个服务 e-Service,则 e-Service 可以表示为一个带条件的有限状态自动机 cFSA,并且 cFSA 是一个七元组 $(Q, I, O, \delta, q_0, F, QC)$,其中 Q 是一个有穷集合,是 cFSA 的状态集,其中状态表示客户与 e-Service 交互序列中的历史记录或条件判断记录; I 是输入命令 input_command 的有穷集合, O 是输出消息 output_message 的有穷集合, $I \times O$ 是 cFSA 的字母表; $\delta: Q \times I \times O \times QC \to Q$ 是转移函数,即给定一个状态,根据 <input_command, output_message> 和条件,cFSA 可以转移到另一个或几个状态; $q_0 \in Q$ 是起始状态; $F \subseteq Q$ 是接收状态集,即用户能够与 e-Service 结束交互的状态集; $QC: Q \times C$ 是状态的条件集.

图2是一个基于带条件的有限状态自动机的旅游购票服务,定义为:

$\text{cFSA} = (Q, I, O, \delta, s_0, F, QC)$,其中状态集 $Q = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$;输入命令集合 $I = \{$userData, continue, buy, stop$\}$,输出消息集合 $O = \{$checkedAccount, notValidClient, displayedTicket-Data, displayedResults, emailNotification$\}$,字母表 $I \times O = \{$userData/checkedAccount, userData/notValid-Client, continue/displayedTicketData, buy/displayed-Results, stop/emailNotification$\}$;转移函数 δ 为: $s_0 \times$userData/checkedAccount $\times$ hasAirTicket $\to s_1$, $s_0 \times$userData/checkedAccount $\times \neg$hasAirTicket $\to s_2$, $s_0 \times$userData/notValidClient $\times$ true $\to s_3$, $s_1 \times$continue/displayedStockData $\times$ true $\to s_4$, $s_4 \times$buy/displayed-Results $\times$true $\to s_4$, $s_5 \times$buy/displayedResults $\times$true $\to s_5$, $s_4 \times$stop/emailNotification $\times$true $\to s_6$, $s_5 \times$stop/emailNotification $\times$true $\to s_7$; $s_0 \in Q$ 是起始状态;接收状态集 $F = \{s_3, s_6, s_7\}$;状态条件集 QC 是:状态 S_0 的条件是 (S_1 and hasAirTicket) or (S_2 and $\neg$hasAirTicket) or S_3,状态 s_1, s_2, s_4, s_5 的条件都是 true,而 s_3, s_6, s_7 没有带条件.

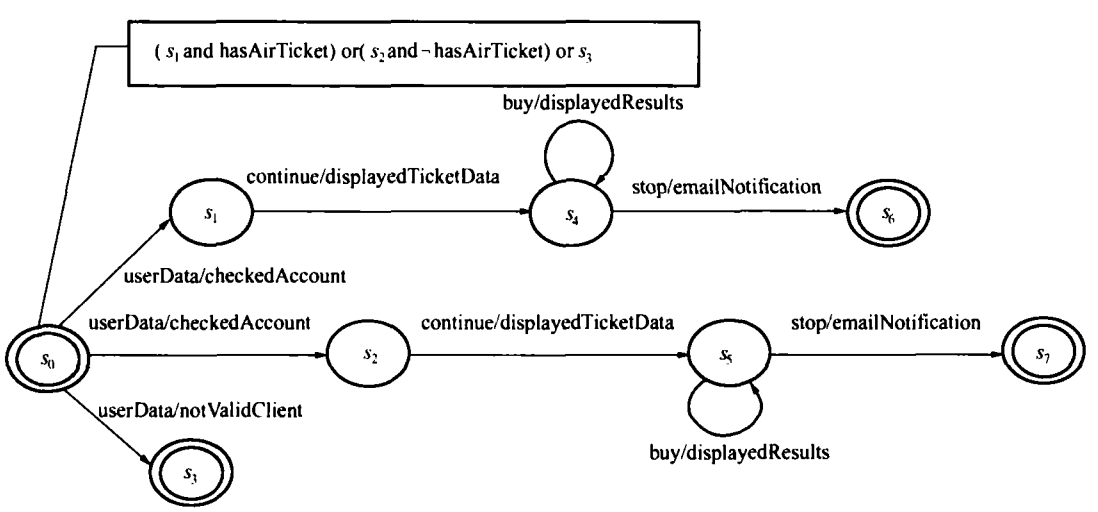


图2 旅游购票服务

Fig. 2 Travel book service

2 服务组合模型

2.1 简单结构的服务组合模型

假设服务 eS_i 的形式化模型 $cFSA_{eS_i} = (Q_{eS_i}, I_{eS_i}, O_{eS_i}, \delta_{eS_i}, S_{0_{eS_i}}, F_{eS_i}, QC_{eS_i})$,其中状态集 $Q_{eS_i} = \{s_{0_{eS_i}}, s_{1_{eS_i}}, \cdots, s_{n_{eS_i}}\}$,输入命令集 $I_{eS_i} = \{i_{0_{eS_i}}, i_{1_{eS_i}}, \cdots, i_{m_{eS_i}}\}$,输出消息集 $O_{eS_i} = \{o_{0_{eS_i}}, o_{1_{eS_i}}, \cdots, o_{k_{eS_i}}\}$,字母表 $I_{eS_i} \times O_{eS_i} = \{io_{0_{eS_i}}, io_{1_{eS_i}}, \cdots, io_{t_{eS_i}}\}$,转移函数 $\delta_{eS_i} = \{\delta_{0_{eS_i}}, \delta_{1_{eS_i}}, \cdots, \delta_{n_{eS_i}}\}$,接收状态集 $F_{eS_i} = \{s_{u_{eS_i}}, \cdots, s_{v_{eS_i}}\}$,状态条件集 QC_{eS_i} 为: $\forall s_{i_{eS_i}} \in Q_{eS_i} \wedge s_{i_{eS_i}} \notin$

F_{eS_i}, $s_{i_{eS_i}}$ 的条件是 $qc_{i_{eS_i}}$.

1) 组合算子 Sequence 表示 2 个服务可以连续执行,即第 1 个服务的输出是第 2 个服务的输入,因而服务 eS_1 和 eS_2 通过算子 Sequence 组合成一个新的服务的条件是 eS_1 的一个接受状态是 eS_2 的起始状态,即 $\exists s_{w_{eS_1}} \in F_{eS_1} = \{ s_{u_{eS_1}}, \cdots, s_{v_{eS_1}} \}$,使得 $s_{w_{eS_1}} = s0_{eS_2}$ 成立. 服务 Sequence(eS_1, eS_2) 的形式化模型 $cFSA_{S(eS_1,eS_2)}$ 定义如下:

$cFSA_{S(eS_1,eS_2)} = (Q_{S(eS_1,eS_2)}, I_{S(eS_1,eS_2)}), O_{S(eS_1,eS_2)}, \delta_{S(eS_1,eS_2)}, s0_{S(eS_1,eS_2)}, F_{S(eS_1,eS_2)}, QC_{S(eS_1,eS_2)}$, 其中状态集 $Q_{S(eS_1,eS_2)} = Q_{eS_1} \cup Q_{eS_2} - s0_{eS_2} = \{ s0_{eS_1}, s1_{eS_1}, \cdots, sn_{eS_1}, s1_{eS_2}, \cdots, sn_{eS_2} \}$, 输入命令集 $I_{S(eS_1,eS_2)} = I_{eS_1} \cup I_{eS_2} = \{ i0_{eS_1}, i1_{eS_1}, \cdots, im_{eS_1}, i0_{eS_2}, i1_{eS_2}, \cdots, im_{eS_2} \}$, 输出消息集 $O_{S(eS_1,eS_2)} = O_{eS_1} \cup O_{eS_2} = \{ o0_{eS_1}, o1_{eS_1}, \cdots, ok_{eS_1}, o0_{eS_2}, o1_{eS_2}, \cdots, ok_{eS_2} \}$, 字母表 $I_{S(eS_1,eS_2)} \times Q_{S(eS_1,eS_2)} = I_{eS_1} \times O_{eS_1} \cup I_{eS_2} \times O_{eS_2} = \{ io0_{eS_1}, io1_{eS_1}, \cdots, io_{eS_1}, io0_{eS_2}, \cdots, io_{eS_2} \}$, 转移函数 $\delta_{S(eS_1,eS_2)} = \delta_{eS_1} \cup \delta_{eS_2} = \{ \delta_{e_{eS_1}} \delta_{1_{eS_1}}, \cdots, \delta_{n_{eS_1}}, \delta_{0_{eS_2}}, \delta_{1_{eS_2}}, \cdots, \delta_{n_{eS_2}} \}$, 起始状态 $S0_{S(eS_1,eS_2)} = s0_{eS_1}$, 接收状态集 $F_{S(eS_1,eS_2)} = F_{eS_1} \cup F_{eS_2} - s_{w_{eS_1}} = \{ s_{u_{eS_1}}, \cdots, s_{v_{eS_1}}, s_{u_{eS_2}}, \cdots, s_{v_{eS_2}} \} - s_{w_{eS_1}} = \{ s_{u_{eS_1}}, \cdots, s_{w-1_{eS_1}}, s_{w+1_{eS_1}}, \cdots, s_{v_{eS_1}}, s_{u_{eS_2}}, \cdots, s_{v_{eS_2}} \}$, 状态条件集 $QC_{S(eS_1,eS_2)} = QC_{eS_1} \cup QC_{eS_2}$, 即 $\forall s_{i_{S(eS_1,eS_2)}} \in Q_{S(eS_1,eS_2)} \wedge s_{i_{S(eS_1,eS_2)}} \notin F_{S(eS_1,eS_2)}$, 如果 $s_{i_{S(eS_1,eS_2)}} = s_{i_{eS_1}}$ 则 $s_{i_{S(eS_1,eS_2)}}$ 的条件是 $qc_{i_{eS_1}}$; 如果 $s_{i_{S(eS_1,eS_2)}} = s_{i_{eS_2}}$ 则 $s_{i_{S(eS_1,eS_2)}}$ 的条件是 $qc_{i_{eS_2}}$.

图 3 是将服务 eS_1 和 eS_2 通过算子 Sequence 组合成一个新的服务的例子.

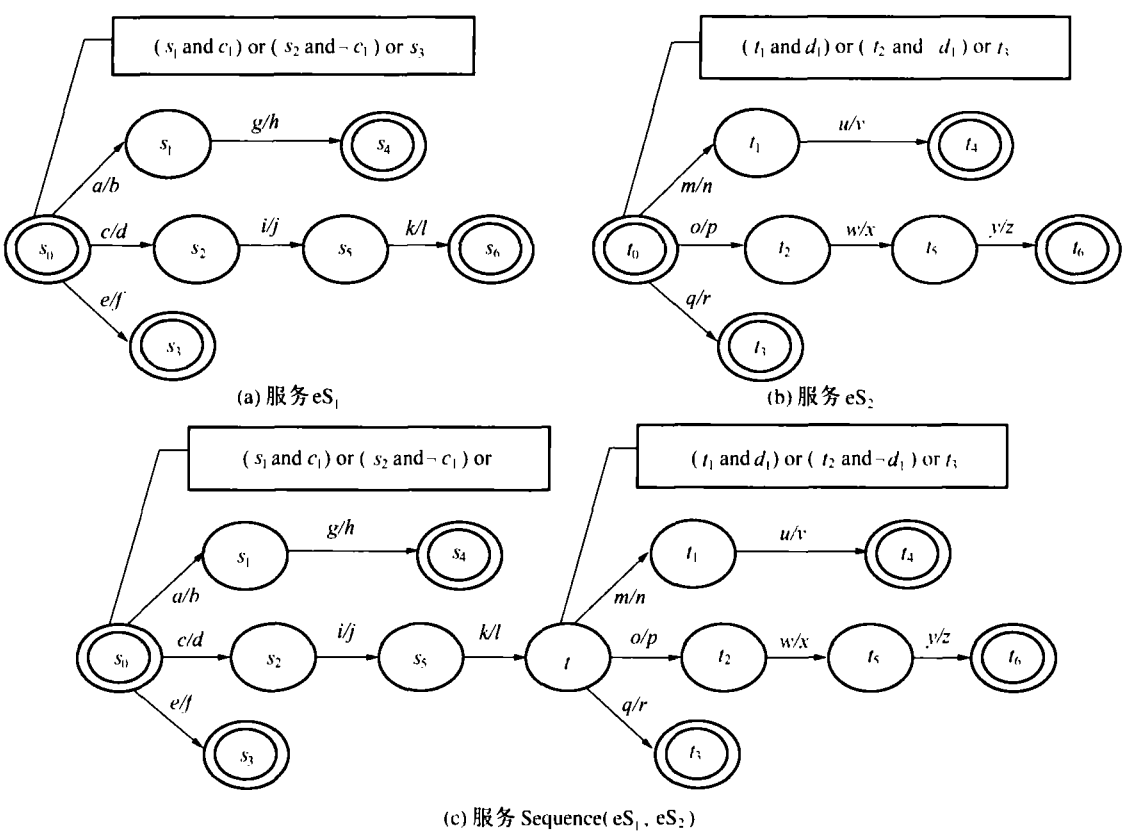


图 3　通过算子 Sequence 的服务组合

Fig. 3　Service composition through sequence

2) 组合算子 Alternative 表示 2 个服务只能执行其中的 1 个,因而任何 2 个服务 eS_1 和 eS_2 都能够通过算子 Alternative 组合成一个新的服务,该服务需要在服务 eS_1 和 eS_2 前增加一个带条件的新的起始状态,如图 4 所示.

服务 Alternative (eS_1, eS_2) 的形式化模型 $cFSA_{A(eS_1,eS_2)}$ 定义如下:

$cFSA_{A(eS_1,eS_2)} = (Q_{A(eS_1,eS_2)}, I_{A(eS_1,eS_2)}, O_{A(eS_1,eS_2)}, \delta_{A(eS_1,eS_2)}, s0_{A(eS_1,eS_2)}, F_{A(eS_1,eS_2)}, Q_{CA(eS_1,eS_2)}$, 其中状态集 $Q_{A(eS_1,eS_2)} = Q_{eS_1} \cup Q_{eS_2} \cup st0_{A(eS_1,eS_2)} = \{ st0_{A(eS_1,eS_2)}, s0_{eS_1}, s1_{eS_1}, \cdots, sn_{eS_1}, s0_{eS_2}, s1_{eS_2}, \cdots, sn_{eS_2}, \varepsilon \}$, 输入命令集 $I_{A(eS_1,eS_2)} = I_{eS_1} \cup I_{eS_2} \cup \varepsilon = \{ i0_{eS_1}, i1_{eS_1}, \cdots, im_{eS_1}, i0_{eS_2}, i1_{eS_2}, \cdots, im_{eS_2}, \varepsilon \}$, 输出消息集 $O_{A(eS_1,eS_2)} =$

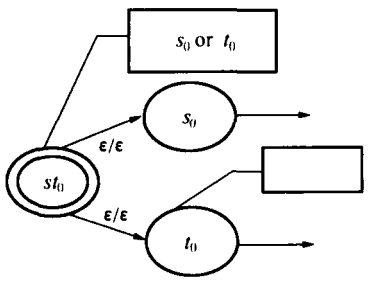


图4 通过算子 Alternative 的服务组合

Fig. 4 Service composition through alternative

$O_{eS_1} \cup O_{eS_2} \cup \varepsilon = \{o_{0_{eS_1}}, o_{1_{eS_1}}, \cdots, o_{k_{eS_1}}, o_{0_{eS_2}}, o_{1_{eS_2}}, \cdots, o_{k_{eS_2}}, \varepsilon\}$，字母表 $I_{S(eS_1,eS_2)} \times Q_{S(eS_1,eS_2)} \cup \{\varepsilon/\varepsilon\} = I_{eS_1} \times O_{eS_1} \cup I_{eS_2} \times O_{eS_2} \cup \{\varepsilon/\varepsilon\} = \{io_{0_{eS_1}}, io_{1_{eS_1}}, \cdots, io_{t_{eS_1}}, io_{0_{eS_2}}, io_{1_{eS_2}}, \cdots, io_{t_{eS_2}}, \varepsilon/\varepsilon\}$，转移函数 $\delta_{A(eS_1,eS_2)} = \delta_{eS_1} \cup \delta_{eS_2} \cup \{st_{0_{A(eS_1,eS_2)}} \times \varepsilon/\varepsilon \times \text{true} \to s_{0_{eS_1}}, st_{0_{A(eS_1,eS_2)}} \times \varepsilon/\varepsilon \times \text{true} \to s_{0_{eS_2}}\} = \{\delta_{0_{eS_1}}, \delta_{1_{eS_1}}, \cdots, \delta_{n_{eS_2}}, \delta_{0_{eS_2}}, \delta_{1_{eS_2}}, \cdots, \delta_{n_{eS_2}}, st_{0_{A(eS_1,eS_2)}} \times \varepsilon/\varepsilon \times \text{true} \to s_{0_{eS_1}}, st_{0_{A(eS_1,eS_2)}} \times \varepsilon/\varepsilon \times \text{true} \to s_{0_{eS_2}}\}$，起始状态为 $st_{0_{A(eS_1,eS_2)}}$，接收状态集 $F_{A(eS_1,eS_2)} = F_{eS_1} \cup F_{eS_2} = \{s_{u_{eS_1}}, \cdots, s_{v_{eS_1}}, s_{u_{eS_2}}, \cdots, s_{v_{eS_2}}\}$，状态条件集 $QC_{A(eS_1,eS_2)} = QC_{eS_1} \cup QC_{eS_2} \cup (s_{0_{eS_1}} \text{ or } s_{0_{eS_2}})$，即 $\forall s_{i_{A(eS_1,eS_2)}} \in Q_{A(eS_1,eS_2)} \wedge s_{i_{A(eS_1,eS_2)}} \notin F_{A(eS_1,eS_2)}$，如果 $s_{i_{A(eS_1,eS_2)}} = s_{i_{eS_1}}$ 则 $s_{i_{A(eS_1,eS_2)}}$ 的条件是 $qc_{i_{eS_1}}$；如果 $s_{i_{A(eS_1,eS_2)}} = s_{i_{eS_2}}$ 则 $s_{i_{A(eS_1,eS_2)}}$ 的条件是 $qc_{i_{eS_2}}$；如果 $s_{i_{A(eS_1,eS_2)}} = st_{0_{A(eS_1,eS_2)}}$ 则 $s_{i_{A(eS_1,eS_2)}}$ 的条件是 $(s_{0_{eS_1}} \text{ or } s_{0_{eS_2}})$.

3) 组合算子 Choice 是组合算子 Alternative 的推广，算子 Choice 表示从 n 个服务中选择 m 个服务执行 $(m \le n)$. 因而服务 Choice $(m, n, eS_1, \cdots, eS_n)$ (简写为 Choice $(eS_1, \cdots, eS_n)$) 需要增加一个带条件的新的起始状态，如图5所示.

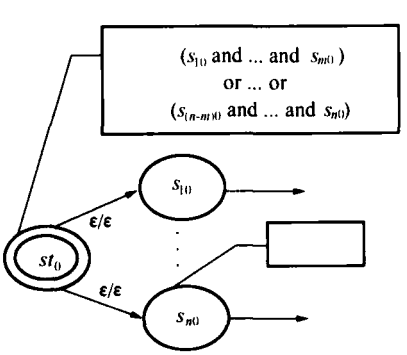


图5 通过算子 Choice 的服务组合

Fig. 5 Service composition through Choice

服务 Choice $(eS_1, \cdots, eS_n)$ 的形式化模型 $cFSA_{Ch(eS_1,\cdots,eS_n)}$ 定义如下：

$cFSA_{Ch(eS_1,\cdots,eS_n)} = (Q_{Ch(eS_1,\cdots,eS_n)}, I_{Ch(eS_1,\cdots,eS_n)}, O_{Ch(eS_1,\cdots,eS_n)}, \delta_{Ch(eS_1,\cdots,eS_n)}, F_{Ch(eS_1,\cdots,eS_n)}, S0_{Ch(eS_1,\cdots,eS_n)}, AC_{Ch(eS_1,\cdots,eS_n)})$ 其中状态集 $Q_{Ch(eS_1,\cdots,eS_n)} = Q_{eS_1} \cup \cdots \cup Q_{eS_n} \cup st0_{Ch(eS_1,\cdots,eS_n)} = \{st0_{Ch(eS_1,\cdots,eS_n)}, s0_{eS_1}, s1_{eS_1}, \cdots, sn_{eS_1}, \cdots, s0_{eS_n}, s1_{eS_n}, \cdots, sn_{eS_n}\}$，输入命令集 $I_{Ch(eS_1,\cdots,eS_n)} = I_{eS_1} \cup \cdots \cup I_{eS_n} \cup \varepsilon = \{i0_{eS_1}, i1_{eS_1} \cdots, im_{eS_1}, \cdots, i0_{eS_1}, i1_{eS_1}, \cdots, im_{eS_1 n}, \varepsilon\}$，输出消息集 $O_{Ch(eS_1,\cdots,eS_n)} = O_{eS_1} \cup \cdots \cup O_{eS_n} \cup \varepsilon = \{o0_{eS_1}, o1_{eS_1}, \cdots, om_{eS_1}, \cdots, o0_{eS_n}, o1_{eS_n}, \cdots, om_{eS_n}, \varepsilon\}$，字母表 $I_{Ch(eS_1,\cdots,eS_n)} \times O_{Ch(eS_1,\cdots,eS_n)} \cup \{\varepsilon/\varepsilon\} = I_{eS_1} \times O_{eS_1} \cup \cdots \cup I_{eS_n} \times O_{eS_n} \cup \{\varepsilon/\varepsilon\} = \{io0_{eS_1}, io1_{eS_1}, \cdots, io_{t_{eS_1}}, \cdots, io0_{eS_n}, io1_{eS_n}, \cdots, io_{t_{eS_n}}, \varepsilon/\varepsilon\}$，转移函数 $\delta_{Ch(eS_1,\cdots,eS_n)} = \delta_{eS_1} \cup \cdots \cup \delta_{eS_n} \cup \{st0_{Ch(eS_1,\cdots,eS_n)} \times \varepsilon/\varepsilon \times \text{true} \to s0_{eS_1}, \cdots, st0_{Ch(eS_1,\cdots,eS_n)} \times \varepsilon/\varepsilon \times \text{true} \to s0_{eS_n}\} = \{\delta_{0_{eS_1}}, \delta_{1_{eS_1}}, \cdots, \delta_{n_{eS_2}}, \delta_{0_{eS_2}}, \delta_{1_{eS_2}}, \cdots, \delta_{n_{eS_2}}, st0_{A(eS_1,eS_2)} \times \varepsilon/\varepsilon \times \text{true} \to s0_{eS_1}, \cdots, st0_{Ch(eS_1,\cdots,eS_n)} \times \varepsilon/\varepsilon \times \text{true} \to s0_{eS_n}\}$，起始状态为 $st0_{Ch(eS_1,\cdots,eS_n)}$，接收状态集 $F_{Ch(eS_1,eS_2)} = F_{eS_1} \cup \cdots \cup F_{eS_n} = \{s_{u_{eS_1}}, \cdots, s_{v_{eS_1}}, \cdots, s_{u_{eS_n}}, \cdots, s_{v_{eS_n}}\}$，状态条件集 $QC_{Ch(eS_1,\cdots,eS_n)} = QC_{eS_1} \cup \cdots \cup QC_{eS_n} \cup ((s0_{eS_1}, \text{and} \cdots \text{and } s0_{eS_m}) \text{ or} \cdots \text{or} (s0_{eS_{(n-m)}} \text{ and} \cdots \text{and } s0_{eS_n}))$.

4) 组合算子 Condition 表示只有满足一定的条件才执行某个服务，因而服务 Condition (ϕ, eS) 需要增加1个带条件的新的起始状态，如图6所示.

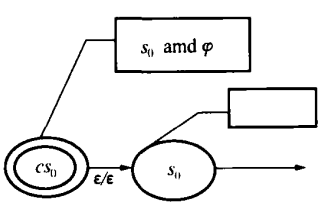


图6 通过算子 Condition 的服务组合

Fig. 6 Service composition through condition

服务 Condition (ϕ, eS) 的形式化模型 $cFSA_{Co(\phi,eS)}$ 定义如下：

$cFSA_{Co(\phi,eS)} = (Q_{Co(\phi,eS)}, I_{Co(\phi,eS)}, O_{Co(\phi,eS)}, \delta_{Co(\phi,eS)}, F_{Co(\phi,eS)}, s0_{Co(\phi,eS)}, QC_{Co(\phi,eS)})$，其中状态集 $Q_{Co(\phi,eS)} = Q_{eS} \cup s0_{Co(\phi,eS)} = \{s0_{Co(\phi,eS)}, s0_{eS}, s1_{eS}, \cdots, sn_{eS}\}$，输入命令集 $I_{Co(\phi,eS)} = I_{eS} \cup \varepsilon = \{i0_{eS}, i1_{eS}, \cdots, im_{eS}, \varepsilon\}$，输出消息集 $O_{Co(\phi,eS)} = O_{eS} \cup \varepsilon = \{o0_{eS}, o1_{eS}, \cdots, om_{eS}, \varepsilon\}$，字母表 $I_{Co(\phi,eS)} \times O_{Co(\phi,eS)} \cup \{\varepsilon/\varepsilon\} = I_{eS} \times O_{eS} \cup \{\varepsilon/\varepsilon\} = \{io0_{eS}, io1_{eS}, \cdots, io_{t_{eS}}, \varepsilon/\varepsilon\}$，转移函数 $\delta_{Co(\phi,eS)} =$

$\delta_{eS} \cup \{ s0_{Co(\phi,eS)} \times \varepsilon/\varepsilon \times true \rightarrow s0_{eS} \} = \{ \delta_{0_{eS}}, \delta_{1_{eS}}, \cdots, \delta_{n_{eS}}, s0_{Co(\phi,eS)} \times \varepsilon/\varepsilon \times true \rightarrow s0_{eS} \}$,起始状态为 $s0_{Co(\phi,eS)}$,接收状态集 $= F_{Co(\phi,eS)} = F_{eS} = \{ s_{u_{eS}}, \cdots, s_{v_{eS}} \}$,状态条件集 $QC_{Co(\phi,eS)} = QC_{eS} \cup (s0_{eS}$ and $\phi)$.

5)组合算子 If-Then-Else 根据条件判断从 2 个服务中选择 1 个执行,因而服务 If-Then-Else (ϕ, eS_1, eS_2) 需要增加 1 个带条件的新的起始状态,如图 7 所示.

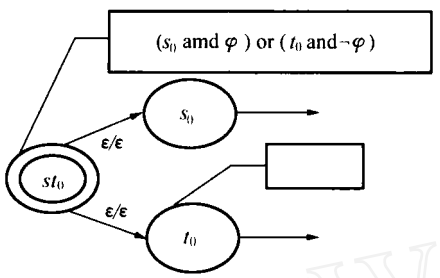


图 7　通过算子 If-Then-Else 的服务组合

Fig. 7　Service composition through If-Then-Else

服务 If-Then-Else (ϕ, eS_1, eS_2) 的形式化模型 $cFSA_{If(\phi,eS_1,eS_2)}$ 定义如下:

$cFSA_{If(\phi,eS_1,eS_2)} = (Q_{If(\phi,eS_1,eS_2)}, I_{If(\phi,eS_1,eS_2)}, O_{If(\phi,eS_1,eS_2)}, \delta_{If(\phi,eS_1,eS_2)}, s0_{If(\phi,eS_1,eS_2)}, F_{If(\phi,eS_1,eS_2)}, QC_{If(\phi,eS_1,eS_2)})$,其中状态集 $Q_{If(\phi,eS_1,eS_2)} = Q_{eS_1} \cup Q_{eS_2} \cup s0_{If(\phi,eS_1,eS_2)} = \{ s0_{If(\phi,eS_1,eS_2)}, s0_{eS_1}, s1_{eS_1}, \cdots, sn_{eS_1}, s0_{eS_2}, s1_{eS_2}, \cdots, sn_{eS_2} \}$,输入命令集 $I_{If(\phi,eS_1,eS_2)} = I_{eS_1} \cup I_{eS_2} \cup \varepsilon = \{ i0_{eS_1}, i1_{eS_1}, \cdots, im_{eS_1}, i0_{eS_2}, i1_{eS_2}, \cdots, im_{eS_2}, \varepsilon \}$,输出消息集 $O_{If(\phi,eS_1,eS_2)} = O_{eS_1} \cup O_{eS_2} \cup \varepsilon = \{ o0_{eS_1}, o1_{eS_1}, \cdots, ok_{eS_1}, o0_{eS_2}, o1_{eS_2}, \cdots, ok_{eS_2}, \varepsilon \}$,字母表 $I_{If(\phi,eS_1,eS_2)} \times O_{If(\phi,eS_1,eS_2)} \cup \{\varepsilon/\varepsilon\} = I_{eS_1} \times O_{eS_1} \cup I_{eS_2} \times O_{eS_2} \cup \{\varepsilon/\varepsilon\} = \{ io0_{eS_1}, io1_{eS_1}, \cdots, io0_{eS_1}, io0_{eS_2}, io1_{eS_2}, \cdots, io0_{eS_2}, \varepsilon/\varepsilon \}$,转移函数 $\delta_{If(\phi,eS_1,eS_2)} = \delta_{eS_1} \cup \delta_{eS_2} \cup \{ s0_{If(\phi,eS_1,eS_2)} \times \varepsilon/\varepsilon \times \phi \rightarrow s0_{eS_1}, s0_{If(\phi,eS_1,eS_2)} \times \varepsilon/\varepsilon \times \neg\phi \rightarrow s0_{eS_2} \} = \{ \delta_{0_{eS_1}}, \delta_{1_{eS_1}}, \cdots, \delta_{n_{eS_2}}, \delta_{0_{eS_2}}, \delta_{1_{eS_2}}, \cdots, \delta_{n_{eS_2}}, s0_{If(\phi,eS_1,eS_2)} \times \varepsilon/\varepsilon \times \phi \rightarrow s0_{eS_1}, s0_{If(\phi,eS_1,eS_2)} \times \varepsilon/\varepsilon \times \neg\phi \rightarrow s0_{eS_2} \}$,起始状态为 $s0_{If(\phi,eS_1,eS_2)}$,接收状态集 $F_{If(\phi,eS_1,eS_2)} = F_{eS_1} \cup F_{eS_2} = \{ s_{u_{eS_1}}, \cdots, s_{v_{eS_1}}, s_{u_{eS_2}}, \cdots, s_{v_{eS_2}} \}$,状态条件集 $QC_{If(\phi,eS_1,eS_2)} = QC_{eS_1} \cup QC_{eS_2} \cup ((s0_{eS_1}$ and $\phi)$ or $(s0_{eS_2}$ and $\neg\phi))$.

6)组合算子 Iteration 表示 1 个服务重复执行多次(n 次),因而服务 Iteration (n, eS) 需要在服务 eS 的每个接收状态前增加 1 个带条件的判断状态,即判断服务 eS 是否执行完 n 次,如图 8 所示.

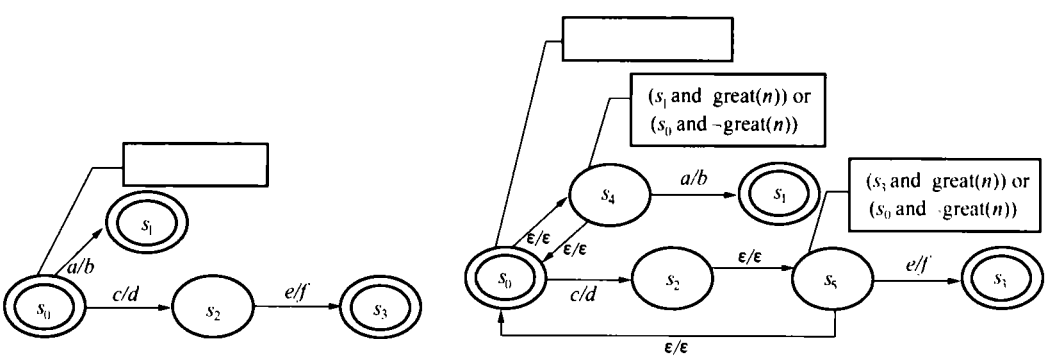


图 8　通过算子 Iteration 的服务组合

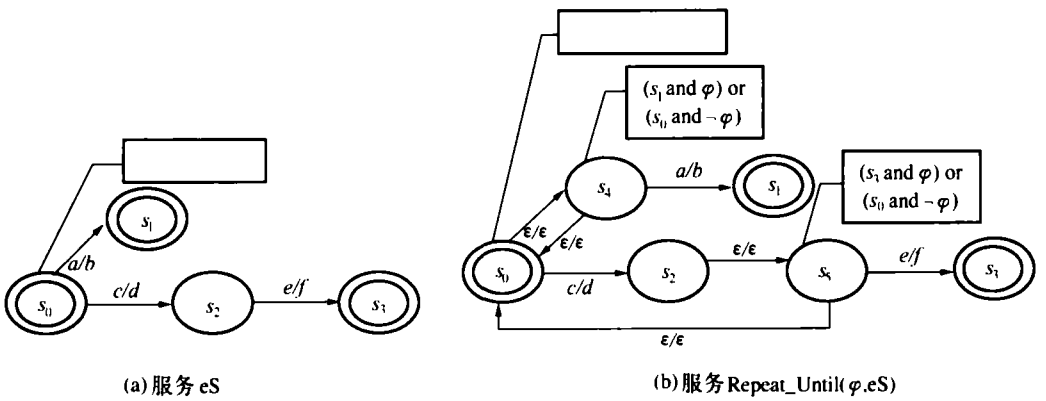
Fig. 8　Service composition Through Iteration

服务 Iteration (n, eS) 的形式化模型 $cFSA_{It(n,eS)}$ 定义如下:

$cFSA_{It(n,eS)} = Q_{It(n,eS)}, I_{It(n,eS)}, O_{It(n,eS)}, \delta_{It(n,eS)}, F_{It(n,eS)}, s0_{It(n,eS)}, QC_{It(n,eS)}$,其中状态集 $Q_{It(n,eS)} = Q_{eS} \cup \{ s'_{u_{eS}}, \cdots, s'_{v_{eS}} \} = \{ S0_{es}, S1_{es}, \cdots, Sn_{es}, s'_{u_{eS}}, \cdots, s'_{u_{eS}} \}$,输入命令集 $I_{It(n,eS)} = I_{eS} \cup \varepsilon = \{ i0_{eS}, i1_{eS}, \cdots, im_{eS}, \varepsilon \}$,输出消息集 $O_{It(n,eS)} = O_{eS} \cup \varepsilon = \{ o0_{eS}, o1_{eS}, \cdots, om_{eS}, \varepsilon \}$,字母表 $I_{It(n,eS)} \times O_{It(n,eS)} \cup \{\varepsilon/\varepsilon\} = I_{eS} \times O_{eS} \cup \{\varepsilon/\varepsilon\} = \{ io0_{eS}, io1_{eS}, \cdots, io_{t_{eS}}, \varepsilon/\varepsilon \}$,转移函数 $\delta_{It(n,eS)} = \delta_{eS} \cup \{ s'_{u_{eS}} \times \varepsilon \times \varepsilon \neg great(n) \rightarrow s0_{It(n,eS)} \} \cup \{ s'_{u_{eS}} \times io_{u_{eS}} \times great(n) \rightarrow s_{u_{eS}} \} \cup \cdots \cup \{ s'_{v_{eS}} \times \varepsilon/\varepsilon \times \neg great(n) \rightarrow s0_{It(n,eS)} \} \cup \{ s'_{v_{eS}} \times io_{v_{eS}} \times great(n) \rightarrow s_{v_{eS}} \} = \{ \delta_{0_{eS}}, \delta_{1_{eS}}, \cdots, \delta_{n_{eS}}, s'_{u_{eS}} \times \varepsilon/\varepsilon \times \neg great(n) \rightarrow s0_{It(n,eS)}, s'_{u_{eS}} \times io_{u_{eS}} \times great(n) \rightarrow s_{u_{eS}}, \cdots, s'_{v_{eS}} \times \varepsilon/\varepsilon \times \neg great(n) \rightarrow s0_{It(n,eS)}, s'_{v_{eS}} \times io_{v_{eS}} \times great(n) \rightarrow s_{v_{eS}} \}$,起始状态为 $s0_{It(n,eS)}$,接收状态集 $F_{It(n,eS)} = F_{eS} = \{ s_{u_{eS}}, \cdots, s_{v_{eS}} \}$,状态条件集 $QC_{It(n,eS)} = QC_{eS} \cup ((s0_{It(n,eS)}$ and $\neg great(n))$ or $(s_{u_{eS}}$ and $great(n))) \cup \cdots \cup ((s0_{It(n,eS)}$ and

¬great(*n*)) or ($s_{v_{eS}}$ and great(*n*))).

7)组合算子 Repeat_Until 表示执行某个服务,直到满足一定的条件,因而服务 Repeat_Until (ϕ, eS) 需要在服务 eS 的每个接收状态前增加1个带条件的判断状态,即判断服务 eS 是否满足条件 ϕ,如图9所示.



(a)服务 eS (b)服务 Repeat_Until(φ,eS)

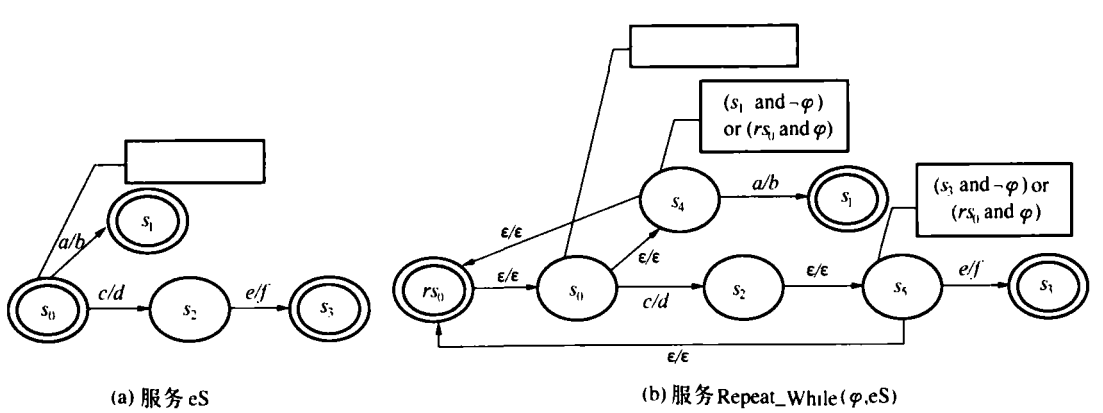
图9 通过算子 Repeat_Until 的服务组合

Fig. 9 Service composition through Repeat_Until

服务 Repeat_Until (ϕ, eS) 的形式化模型 $cFSA_{RU(\phi,eS)}$ 定义如下:

$cFSA_{RU(\phi,eS)} = (Q_{RU(\phi,eS)}, I_{RU(\phi,eS)}, O_{RU(\phi,eS)}, \delta_{RU(\phi,eS)}, F_{RU(\phi,eS)}, s0_{RU(\phi,eS)}, QC_{RU(\phi,eS)})$,其中状态集 $Q_{RU(\phi,eS)} = Q_{eS} \cup \{s'_{u_{eS}}, \cdots, s'_{v_{eS}}\} = \{s0_{eS}, s1_{eS}, \cdots, sn_{eS}, s'_{u_{eS}}, \cdots, s'_{v_{eS}}\}$,输入命令集 $I_{RU(\phi,eS)} = I_{eS} \cup \varepsilon = \{i0_{eS}, i1_{eS}, \cdots, im_{eS}, \varepsilon\}$,输出消息集 $O_{RU(\phi,eS)} = O_{eS} \cup \varepsilon = \{o0_{eS}, o1_{eS}, \cdots, om_{eS}, \varepsilon\}$,字母表 $I_{RU(\phi,eS)} \times O_{RU(\phi,eS)} \cup \{\varepsilon/\varepsilon\} = I_{eS} \times O_{eS} \cup \{\varepsilon/\varepsilon\} = \{io0_{eS}, io1_{eS}, \cdots, iot_{eS}, \varepsilon/\varepsilon\}$,转移函数 $\delta_{RU(\phi,eS)} = \delta_{eS} \cup \{s'_{u_{eS}} \times \varepsilon/\varepsilon \times \neg\phi \to s0_{RU(n,eS)}\} \cup \{s'_{u_{eS}} \times io_{u_{eS}} \times \phi \to s_{u_{eS}}\} \cup \cdots \cup \{s'_{v_{eS}} \times \varepsilon/\varepsilon \times \phi \to s0_{RU(n,eS)}\} \cup \{s'_{v_{eS}} \times io_{v_{eS}} \times \phi \to s_{v_{eS}}\} = \{\delta_{0_{eS}}, \delta_{1_{eS}}, \cdots, \delta_{n_{eS}}, s'_{u_{eS}} \times \varepsilon/\varepsilon\phi \to s0_{RU(n,eS)}, s'_{u_{eS}} \times io_{u_{eS}} \times \phi \to s_{u_{eS}}, \cdots, s'_{v_{eS}} \times \varepsilon/\varepsilon \times \neg\phi \to s0_{RU(n,eS)}, s'_{v_{eS}} \times io_{v_{eS}} \times \phi \to s_{v_{eS}}\}$,起始状态为 $s0_{RU(\phi,eS)}$,接收状态集 $F_{RU(\phi,eS)} = F_{eS} = \{s_{u_{eS}}, \cdots, s_{v_{eS}}\}$,状态条件集 $QC_{RU(\phi,eS)} = QC_{eS} \cup ((s0_{RU(\phi,eS)}$ and $\neg\phi$) or ($s_{u_{eS}}$ and ϕ)) $\cup \cdots \cup ((s0_{RU(\phi,eS)}$ and $\neg\phi$) or ($s_{v_{eS}}$ and ϕ)).

8)组合算子 Repeat_While 表示当满足某个条件就执行服务,直到该条件不满足为止,可以看出,组合算子 Repeat_While 是组合算子 Condition 和组合算子 Repeat_Until 的结合,因而服务 Repeat_While (ϕ, eS) 需要增加1个带条件的新的起始状态,并在服务 eS 的每个接收状态前增加一个带条件的判断状态,如图10所示.



(a)服务 eS (b)服务 Repeat_While(φ,eS)

图10 通过算子 Repeat_While 的服务组合

Fig. 10 Service composition through Repeat_While

服务 Repeat_While (ϕ, eS) 的形式化模型 $cFSA_{RW(\phi,eS)}$ 定义如下:

$cFSA_{RW(\phi,eS)} = (Q_{RW(\phi,eS)}, I_{RW(\phi,eS)}, O_{RW(\phi,eS)}, \delta_{RW(\phi,eS)}, F_{RW(\phi,eS)}, s0_{RW(\phi,eS)}, QC_{RW(\phi,eS)})$,其中状态集 $Q_{RW(\phi,eS)} = Q_{eS} \cup \{s'_{u_{eS}}, \cdots, s'_{v_{eS}}\} \cup s0_{RW(\phi,eS)} = \{s0_{eS}, s1_{eS}, \cdots, sn_{eS}, s'_{u_{eS}}, \cdots, s'_{v_{eS}}, s0_{RW(\phi,eS)}\}$,输入命令集

$I_{RW(\phi,\mathrm{eS})} = I_{\mathrm{eS}} \cup \varepsilon = \{ i0_{\mathrm{eS}}, i1_{\mathrm{eS}}, \cdots, im_{\mathrm{eS}}, \varepsilon \}$,输出消息集 $O_{RW(\phi,\mathrm{eS})} = O_{\mathrm{eS}} \cup \varepsilon = \{ o0_{\mathrm{eS}}, o1_{\mathrm{eS}}, \cdots, om_{\mathrm{eS}}, \varepsilon \}$,字母表 $I_{RW(\phi,\mathrm{eS})} \times O_{RW(\phi,\mathrm{eS})} \cup \{\varepsilon/\varepsilon\} = I_{\mathrm{eS}} \times O_{\mathrm{eS}} \cup \{\varepsilon/\varepsilon\} = \{ io0_{\mathrm{eS}}, io1_{\mathrm{eS}}, \cdots, iot_{\mathrm{eS}}, \varepsilon/\varepsilon \}$,转移函数 $\delta_{RW(\phi,\mathrm{eS})} = \delta_{\mathrm{eS}} \cup \{ s'_{u_{\mathrm{eS}}} \times \varepsilon/\varepsilon \times \phi \to s0_{RW(n,\mathrm{eS})} \} \cup \{ s'_{u_{\mathrm{eS}}} \times io_{u_{\mathrm{eS}}} \times \phi \to s_{u_{\mathrm{eS}}} \} \cup \cdots \cup \{ s'_{v_{\mathrm{eS}}} \times \varepsilon/\varepsilon \times \phi \to s0_{RW(n,\mathrm{eS})} \} \cup \{ s'_{v_{\mathrm{eS}}} \times io_{v_{\mathrm{eS}}} \times \phi \to s_{v_{\mathrm{eS}}} \} \cup \{ s0_{RW(n,\mathrm{eS})} \times \varepsilon/\varepsilon \times \mathrm{true} \to s0_{\mathrm{eS}} \} = \{ \delta_{0_{\mathrm{eS}}}, \delta_{1_{\mathrm{eS}}}, \cdots, \delta_{h_{\mathrm{eS}}}, s'_{u_{\mathrm{eS}}} \times \varepsilon/\varepsilon \times \phi \to s0_{RU(n,\mathrm{eS})}, s'_{u_{\mathrm{eS}}} \times io_{u_{\mathrm{eS}}} \times \neg\phi \to s_{u_{\mathrm{eS}}}, \cdots, s'_{v_{\mathrm{eS}}} \times \varepsilon/\varepsilon \times \phi \to s0_{RW(n,\mathrm{eS})}, s'_{v_{\mathrm{eS}}} \times io_{v_{\mathrm{eS}}} \times \neg\phi \to s_{v_{\mathrm{eS}}}, s0_{RW(n,\mathrm{eS})} \times \varepsilon/\varepsilon \times \mathrm{true} \to s0_{\mathrm{eS}} \}$,起始状态为 $s0_{RW(n,\mathrm{eS})}$,接收状态集 $F_{RW(\phi,\mathrm{eS})} = F_{\mathrm{eS}} = \{ s_{u_{\mathrm{eS}}}, \cdots, s_{v_{\mathrm{eS}}} \}$,状态条件集 $QC_{RW(\phi,\mathrm{eS})} = QC_{\mathrm{eS}} \cup ((s0_{RW(\phi,\mathrm{eS})} \text{ and } \neg\phi) \text{ or } (s_{u_{\mathrm{eS}}} \text{ and } \phi)) \cup \cdots \cup ((s0_{RW(\phi,\mathrm{eS})} \text{ and } \neg\phi) \text{ or } (s_{v_{\mathrm{eS}}} \text{ and } \phi))$.

2.2　复杂结构的服务组合模型

上面介绍了直接通过算子 Sequence、Alternative、Choice、Condition、If-Then-Else、Iteration、Repeat_Until 或 Repeat_While 进行服务组合的简单结构的服务组合模型.几个服务组合算子可以联合起来进行复杂结构的服务组合,例如,Repeat_While (ϕ, Sequence(eS_1, eS_2)) 就是通过算子 Sequence 和 Repeat_While 来进行服务组合的.

由 2.1 节介绍的简单结构服务组合可知,通过任意组合算子所得到的组合服务都是 1 个有限状态自动机,因而复杂结构的服务组合可以按顺序依次进行.例如,对于服务组合 Repeat_While (ϕ, Sequence(eS_1, eS_2)),首先将服务 eS_1 和服务 eS_2 进行 Sequence 组合,得到服务 Sequence(eS_1, eS_2),然后再对服务 Sequence(eS_1, eS_2) 进行 Repeat_While 组合.服务 Repeat_While (ϕ, Sequence(eS_1, eS_2)) 的形式化模型 $cFSA_{RWS}$ 定义如下:

$cFSA_{RWS} = (Q_{RWS}, I_{RWS}, O_{RWS}, \delta_{RWS}, F_{RWS}, s0_{RWS}, QC_{RWS})$,其中状态集 $Q_{RWS} = (Q_{eS_1} \cup Q_{eS_2} - s0_{eS_2}) \cup \{ s'_{u_{\mathrm{eS}}}, \cdots, s'_{v_{\mathrm{eS}}} \} \cup s0_{RWS}$,输入命令集 $I_{RWS} = I_{eS_1} \cup I_{eS_2} \cup \varepsilon$,输出消息集 $O_{RWS} = O_{eS_1} \cup O_{eS_2} \cup \varepsilon$,字母表为 $I_{RWS} \times O_{RWS} \cup \{\varepsilon/\varepsilon\} = I_{eS_1} \times O_{eS_1} \cup I_{eS_2} \times O_{eS_2} \cup \{\varepsilon/\varepsilon\}$,转移函数 $\delta_{RWS} = \delta_{eS_1} \cup \delta_{eS_2} \cup \{ s'_{u_{eS12}} \times \varepsilon/\varepsilon \times \phi \to s0_{RWS} \} \cup \{ s'_{u_{eS12}} \times io_{u_{eS12}} \times \neg\phi \to s_{u_{eS12}} \} \cup \cdots \cup \{ s'_{v_{eS12}} \times \varepsilon/\varepsilon \times \phi \to s0_{RWS} \} \cup \{ s'_{v_{eS12}} \times io_{v_{eS12}} \times \neg\phi \to s_{v_{eS12}} \} \cup \{ s0_{RWS} \times \varepsilon/\varepsilon \times \mathrm{true} \to s0_{eS12} \}$,起始状态为 $s0_{RWS}$,接收状态集 $F_{RWS} = F_{eS_1} \cup F_{eS_2} - s_{w_{eS_1}}$,状态条件集 $QC_{RWS} = QC_{eS_1} \cup QC_{eS_2} \cup ((s0_{RWS} \text{ and } \neg\phi) \text{ or } (s_{u_{eS12}} \text{ and } \phi)) \cup \cdots \cup ((s0_{RWS} \text{ and } \neg\phi) \text{ or } (s_{v_{eS12}} \text{ and } \phi))$,其中 $s_{u_{eS12}}, \cdots, s_{v_{eS12}}$ 表示服务 Sequence(eS_1, eS_2) 的接收状态,$s'_{u_{eS12}}, \cdots, s'_{v_{eS12}}$ 分别表示 $s_{u_{eS12}}, \cdots, s_{v_{eS12}}$ 前面所增加的条件判断状态,$io_{v_{eS12}}$ 表示服务 Sequence(eS_1, eS_2) 的字母表,$s0_{eS12}$ 表示服务 Sequence(eS_1, eS_2) 的起始状态.

2.3　代数性质

1) 服务组合算子 Sequence 满足结合律.

2) 服务组合算子 Alternative 满足结合律、交换律和幂等律.

3) 服务组合算子 Sequence 对服务组合算子 Alternative 满足分配律.

4) Choice(1, 2, eS_1, eS_2) = Alternative(eS_1, eS_2).

5) Sequence (Iteration (n, eS), Iteration (m, eS)) = Iteration ($n+m$, eS).

6) Iteration(n, eS) = Sequence($\underbrace{eS, \cdots, eS}_{n\text{次}}$).

7) Condition(ϕ, eS) = If-Then-Else(ϕ, eS, ε).

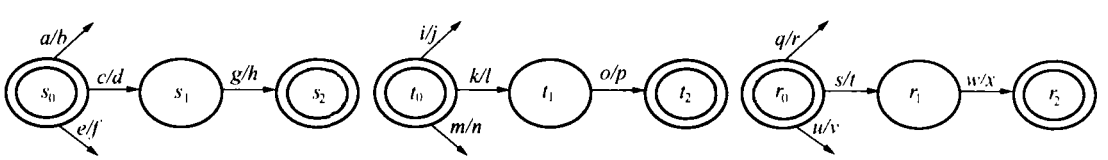
8) If-Then-Else (ϕ, eS_1, eS_2) = If-Then-Else ($\neg\phi$, eS_2, eS_1).

9) Repeat_Until(ϕ, eS) = Sequence(eS, Repeat_While($\neg\phi$, eS)).

10) 服务组合算子操作是封闭的,即通过任意组合算子所得到的组合服务仍然是 1 个带条件的有限状态自动机.

所有这些性质都可以类似的证明,下面只证明性质 1).

证明:假设服务 eS_1、eS_2 和 eS_3 分别如图 11 所示:



(a) 服务 eS_1　　(b) 服务 eS_2　　(c) 服务 eS_3

图 11　服务 eS_1,服务 eS_2,服务 eS_3

Fig. 11　Service eS_1, service eS_2, and service eS_3

则利用2.1和2.2节介绍的服务组合方法可得服务 Sequence(Sequence(eS_1, eS_2), eS_3)和服务 Sequence(eS_1, Sequence(eS_2, eS_3))是相同的,如图12所示:

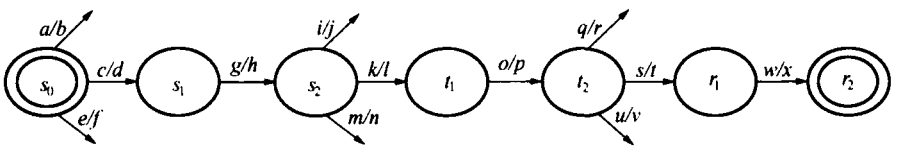


图12 服务 Sequence(Sequence(eS_1,eS_2),eS_3)和 Sequence(eS_1,Sequence(eS_2,eS_3))

Fig. 12 Service Sequence(Sequence(eS_1,eS_2),eS_3) and Sequence(eS_1,Sequence(eS_2,eS_3))

所以 Sequence(Sequence(eS_1, eS_2), eS_3) = Sequence(eS_1, Sequence(eS_2, eS_3)).证毕.

3 实现与分析

因为通过任意组合算子所得到的组合服务仍然是1个带条件的有限状态自动机(见2.3节的性质10),所以任何组合服务都可以转化为1个有限状态自动机来实现.关键问题是如何将几个服务转化为1个基于有限状态自动机的组合服务.由于服务组合算子操作的封闭性,可以利用递归的方法来实现这种转化.例如,服务 Repeat_While(ϕ, Sequence(eS_1, eS_2))可以转化为图13所示的有限状态自动机,即首先将服务 eS_1 和服务 eS_2 进行 Sequence 组合,得到服务 Sequence(eS_1, eS_2),然后再对服务 Sequence(eS_1, eS_2)进行 Repeat_While 组合.

用Java语言实现了将几个服务转化为1个基于有限状态自动机的组合服务算法 ServicesTocFSA,算法 ServicesTocFSA 描述如下:

算法:服务组合算法 ServicesTocFSA

输入:需要组合的原子服务、组合服务描述说明 CSD

输出:一个带条件的有限状态自动机

1)从组合服务描述说明 CSD 中读取第一个组合算子 CS,并取出组合算子 CS 的参数 PS.

2)如果 PS 中不含组合算子,则按2.1节介绍的对组合算子 CS 的组合方法进行服务组合,得到一个带条件的有限状态自动机 cFSA,并 result = cFSA,Goto 6).

3)从 PS 中取出组合服务描述说明 CSD 的子描述 SCSD.

4)递归调用 ServicesTocFSA(SCSD),得到结果 mresult.

5)按2.1节介绍的对组合算子 CS 的组合方法对结果 mresult 进行服务组合,得到一个带条件的有限状态自动机,假设结果是 result.

6)返回 result.

7)算法结束.

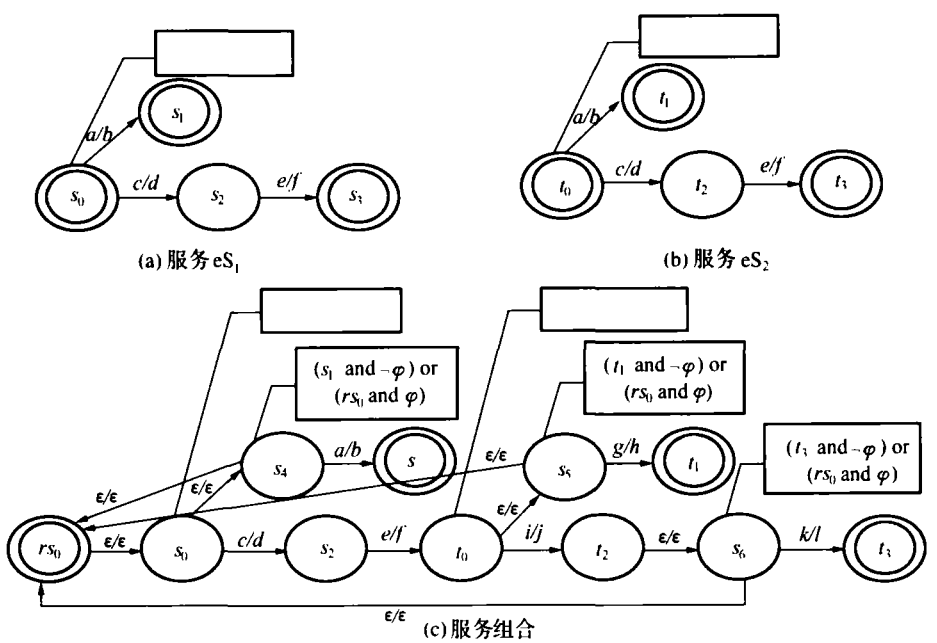


图13 服务 Repeat_While(ϕ,Sequence(eS_1,eS_2)

Fig. 13 Service Repeat_While(ϕ,Sequence(eS_1,eS_2)

由于算法 ServicesTocFSA 能够将几个服务转化成一个有限状态自动机,从而可以根据转化后得

到的有限状态自动机来执行新的组合服务.

基于有限状态自动机的服务组合具有高效性,因为算法 ServicesTocFSA 是多项式时间,也就是说,将几个服务转化为一个基于有限状态自动机的组合服务能在多项式时间内完成;而组合服务执行的时间依赖于具体的服务.

4　结束语

针对目前服务计算的理论基础研究中存在的问题,在 D Berardi 和 A Wombacher 的基础上研究了基于有限状态自动机的服务组合问题.针对服务组合的需求和特点,首先提出了一种带条件的有限状态自动机模型 cFSA,并给出了基于 cFSA 的服务理论模型.在基于 cFSA 的服务理论模型的基础上提出了一种基于有限状态自动机的服务组合的形式化理论模型,并给出了该模型的代数性质和实现方法.进一步工作主要有:研究基于有限状态自动机的服务组合的优化问题和验证问题等.

参考文献:

[1]PAPAZOGLOU M P, GEORGAKOPOULOS D. Serive-oriented computing[J]. Communcications of the ACM, 2003, 46(10): 25 - 65.

[2] DIETER F, CHRISTOPH B. The Web service modeling framework WSMF[J]. Electronic Commerce Research and Applications, 2002, 1(2): 113 - 137.

[3] MCLLRAITH S, SON T C, ZENG H. Semantic Web services[J]. IEEE Intelligent Systems, 2001, 16(2): 46 - 53.

[4] 蒋运承, 张海俊, 董明楷, 史忠植. 多主体系统中的动态服务匹配[J]. 电子学报, 2004, 32(3): 457 - 461.
JIANG Yuncheng, ZHANG Haijun, DONG Mingkai, SHI Zhongzhi. Dynamic service matchmaking in multi-agent system[J]. Acta Electronica Sinica, 2004, 32(3): 457 - 461.

[5] CHINNICI R, MOREAU J J, RYMAN A, *et al*. Web services description language(WSDL) Version 2.0 Part 1: core language[EB/OL]. http://www.w3.org/TR/wsdl20/, 2005 - 08 - 21.

[6] MARTIN D, ANKOLEKAR A, BURSTEIN M, *et al*. OWL - S 1.1 release[EB/OL]. http://www.daml.org/services/owl - s/1.1/, November 26, 2004.

[7] 史忠植, 蒋运承, 张海俊, 董明楷. 基于描述逻辑的主体服务匹配[J]. 计算机学报, 2004, 27(5): 625 - 635.
SHI Zhongzhi, JIANG Yuncheng, ZHANG Haijun, DONG Mingkai. Agent service matchmaking based on description logic[J]. Chinese Journal of Computers, 2004, 27(5): 625 - 635.

[8] LEYMANN F. Web services flow language (WSFL) Version 1.0[EB/OL]. http://www - 3.ibm.com/software/solutions/webservices/pdf/WSFL.pdf, May, 2001.

[9] THATTE S. XLANG: Web services for business process design[EB/OL]. http://www.gotdotnet.com/team/xml_wsspecs/xlang - c/default.htm, November 12, 2005.

[10] ANDREWS T, CURBERA F, DHOLAKIA H, *et al*. Business process execution language for Web services Version 1.1[EB/OL]. http://www - 106.ibm.com/developer - works/webservices/library/ws - bpel, November 10, 2005.

[11] 蒋运承, 史忠植. QoS 驱动的主体服务匹配[J]. 小型微型计算机系统, 2005, 26(4): 687 - 692.
JIANG Yuncheng, SHI Zhongzhi. Quality of service driven agent service matchmaking[J]. Mini-Micro Systems, 2005, 26(4): 687 - 692.

[12] BERARDI D, ROSA F D, SANTIS L D, *et al*. Finite state automata as conceptual model for e-Services[J]. Journal of Design & Process Science: Transactions of the SDPS, Society for Process & Design Sciences, 2003, 7: 21 - 30.

[13] WOMBACHER A, FANKHAUSER P, MAHLEKO B, *et al*. Matchmaking for business processes based on choreographies[J]. International Journal of Web Services, 2004, 1(4): 1545 - 7362.

作者简介:

蒋运承, 男, 1974 年生, 博士, 副教授, 2004 年毕业于中国科学院计算技术研究所, 现为中山大学计算机科学系博士后. 主要研究方向为描述逻辑、语义 Web 和多 Agent 系统. 主持和参加国家和省市自然科学基金项目、863 计划项目 7 项. 在中国科学、软件学报、计算机学报等刊物上发表论文 20 余篇.

汤　庸, 男, 1964 年生, 博士, 教授, 博士生导师. 主要研究方向为时态数据库、知识工程、软件工程、CSCW. 主持和参加国家和省市自然科学基金、重点攻关、重大科技专项及地方合作项目 20 余项. 出版著作 8 部, 科研成果获全国和省厅级科技进步奖、优秀论文奖和优秀 CAI 软件等奖励 12 项. 发表论文 50 余篇.

邓培民, 男, 1950 年生, 教授, 硕士生导师, 毕业于广西师范大学. 主要研究方向为自动机理论及应用、计算机代数. 主持和参加国家和省市自然科学基金项目 3 项. 在计算机学报、数学进展等刊物上发表论文 20 余篇.