Web Service Composition: An Approach Using Effect-Based Reasoning

Puwei Wang[1,3] and Zhi Jin[1,2]

[1] Institute of Computing Technology, Chinese Academy of Sciences
[2] Academy of Mathematics and System Sciences, Chinese Academy of Sciences
[3] Graduate University of Chinese Academy of Sciences
Beijing 100080, China
wangpw@ict.ac.cn, zhijin@amss.ac.cn

Abstract. This paper proposes an ontology-based approach to compose Web services using the effect-based reasoning. The environment ontology is to provide formal and sharable specifications of environment entities of Web services in a particular domain. For each environment entity, there is a corresponding hierarchical state machine for specifying its dynamic characteristics. Then, this approach proposes to use the effects of a Web service on its environment entities for specifying the Web service's capabilities and designates the effect as the traces of the state transitions the Web service can impose on its environment entities. So, the service composition can be conducted by the effect-based reasoning.

1 Introduction

Web services are self-contained and modular components which are autonomous and loosely-coupled. Web service composition happens when a request can not be satisfied by any elementary Web services. It combines some elementary Web services to create a new and composite Web service so that the composite Web service has the capabilities to satisfy the request. Capability descriptions of Web services are important for enhancing the quality of the composition. Therefore, we need an expressive external manifestation of Web services to describe service capabilities for the composition.

This paper proposes an approach for Web service composition by using effect-based reasoning on the environment entities. The distinct feature is to introduce environments as the objective reference of Web service capability. In fact, environment is a key concern in the requirements modelling and has been recognized as the semantic basis of the meaning of the requirements. In our approach, we view environment of a Web service to be composed of those controllable entities (or called "environment entities") that the Web service can impose effects on. Our motivations are listed as follows: 1) *Environment can exhibit the capabilities of Web services while Web services' behaviors are installed in the environment.* In requirement engineering domain, it has been argued that the meaning of requirements can be depicted by the optative interaction of software system with its environment as well as the causal relationships among these interactions [1].

D. Georgakopoulos et al. (Eds.): ICSOC 2006 Ws, LNCS 4652, pp. 62–73, 2007.

2) *Environment entities are inherent sharable for loosely coupled Web services.* Web services are often developed by different teams, and are described in different conceptual framework without agreement. Based on the optative interaction of Web services with its environment entities, understanding between the Web services are assured by instinct. 3) *Dynamic composition of Web services are depicted based on their stable environment.* Though Web services are autonomous and evolving in their life-cycles for continual users' demands, the environments which exhibit their capabilities are usually stable.

Example 1. An example is embedded in this paper to illustrate our ideas. Assume that there are four services, *ticket-selling service*(TSA), *ticket-delivery service*(TDS), *hotel service*(HOS) and *bank service*(BAS) respectively. There are three sharable environment entities for the four services, *ticket*, *hotelroom* and *creditcard*. Given a goal $\mathcal{G}_{travel}$ that needs *budget traveling service* for travelers. This goal can be represented as a set of desired effects on *ticket*, *hotelroom* and *creditcard*. Obviously, any of the four services can not satisfy $\mathcal{G}_{travel}$ by its own. What we expect to do is to obtain a composition of the four services satisfying $\mathcal{G}_{travel}$ conducted by reasoning on *ticket*, *hotelroom* and *creditcard* in terms of satisfiability of the desired effects on these environment entities.

The rest of this paper is structured as follows: Section 2 gives the definition of the environment ontology. Section 3 defines the effect on an environment entity and the effect-based Web service capability profile. Section 4 describes a Web service composition conducted by the effect-based reasoning. Finally, section 5 analyzes related works and draws a conclusion.

2 Environment Ontology

Environment entities are domain-relevant, and then the conceptualization of environment entities, i.e., the environment ontology, can constitute as sharable domain knowledge base for Web services. We extend the general ontology structure by attaching each environment entity a tree-like hierarchical state machine for specifying its domain life-cycle. The environment ontology is defined as follows.

Definition 1. $\mathcal{E}nv\mathcal{O} \stackrel{\text{def}}{=} \{Ent, \mathcal{X}^c, \mathcal{H}^c, \mathcal{HSM}, inter, res\}$, *in which:*

- Ent *is a finite set of environment entities,*
- $\mathcal{X}^c \subseteq Ent \times Ent$ *is an ingredient relation between the environment entities,* $\forall e_1, e_2 \in Ent$, $\langle e_1, e_2\rangle \in \mathcal{X}^c$ *means that* e_1 *is an ingredient of* e_2*,*
- $\mathcal{H}^c \subseteq Ent \times Ent$ *is a taxonomic relation between the environment entities.* $\forall e_1, e_2 \in Ent$, $\langle e_1, e_2\rangle \in \mathcal{H}^c$ *means that* e_1 *is a subconcept of* e_2*,*
- $\mathcal{HSM}$ *is a finite set of tree-like hierarchical state machines (called "THSM"),*
- $inter \subseteq \mathcal{HSM} \times \mathcal{HSM}$ *is a message exchange relation between THSMs.* $hsm_1, hsm_2 \in \mathcal{HSM}$, $\langle hsm_1, hsm_2\rangle \in inter$ *means that* hsm_1 *and* hsm_2 *interact with each other by message exchange,*
- $res : Ent \leftrightarrow \mathcal{HSM}$ *is a bijective relation.* $\forall e \in Ent$*, there is one and only* $hsm \in \mathcal{HSM}$*, such that* $hsm{=}res(e)$*.It is say that* hsm *is the THSM of* e*.*

For a THSM, its states may be ordinary states or super-states (s_{super}) which can be further subdivided into another basic state machines ($\mathcal{N}_{sub}$). In our approach, each state s' in $\mathcal{N}_{sub}$ is called a *child* of super-state s_{super} (s' *child* s_{super}), and start state λ_0 in $\mathcal{N}_{sub}$ is called *default child* of s_{super}. Hierarchical skeleton is expected to assure that hierarchical state machine has flexibility and different granularity. Tree-like skeleton assures that each basic state machine has one and only one super-state. A THSM (called "domain THSM") in the environment ontology is designed for life-cycle description of an environment entity in a domain. The effects of Web services then can be modeled based on the domain THSMs.

For the four elementary services in above example, their environment is specified as a simplified Budget Traveling Environment Ontology (called "*BTO*").

Table 1. Budget Traveling Environment Ontology

Entity	**Meaning**
ticket	Taking travelers to their destinations. Its domain THSM is $hsm(ticket)$
hotelroom	Housing travelers. Its domain THSM is $hsm(hotelroom)$
creditcard	Providing a method of payment. Its domain THSM is $hsm(creditcard)$
merchandise	Goods in business. Its domain THSM is $hsm(merchandise)$
itinerary	Route of ticket. Its domain THSM is $hsm(itierary)$

Relation	**Meaning**
$\langle ticket, merchandise\rangle \in \mathcal{H}^c_{bto}$	Ticket is a merchandise, i.e., domain THSM of ticket inherits domain THSM of merchandise
$\langle itinerary, ticket\rangle \in \mathcal{X}^c_{bto}$	Itinerary is a part of ticket, i.e., domain THSM of itinerary is a part of domain THSM of ticket

$\langle hsm(ticket), hsm(creditcard)\rangle \in inter_{bto}$	Ticket can be paid by credit card
$\langle hsm(hotelroom), hsm(creditcard)\rangle \in inter_{bto}$	Hotel room can be paid by credit card

We develop an application demo for visualizing the representation of *BTO* which is encoded in XML-style. Fig.1 is a screenshot for showing the domain THSMs, $hsm(ticket)$, $hsm(hotelroom)$ and $hsm(creditcard)$, and the message exchange relation between them. They are used to represent life-cycle of environment entities *ticket*, *hotelroom* and *creditcard*. In the *ticket*'s domain THSM, *instantiated*, *available* and *sold* are three super-states, which are subdivided into three basic state machines respectively. Similarly, domain THSMs of environment entities *creditcard* and *hotelroom* are described.

Moreover, there is a message exchange relation between $hsm(ticket)$ and $hsm(creditcard)$, i.e., $\langle hsm(ticket), hsm(creditcard)\rangle \in inter$, because that output from *creditcard*'s state *valid* can trigger the state transition of *ticket* from *ordered* to *sold*, and output from *ticket*'s state *sold* can trigger state transition of *creditcard* from *non-charged* to *charged*.

Obviously, it describes that ticket can be paid by credit card. Similarly, there also is a message exchange relation between $hsm(hotelroom)$ and $hsm(creditcard)$ because that hotel room also can be paid by credit card. For simplicity, message exchanges are only denoted as light-gray lines with double arrowheads in Fig.1.

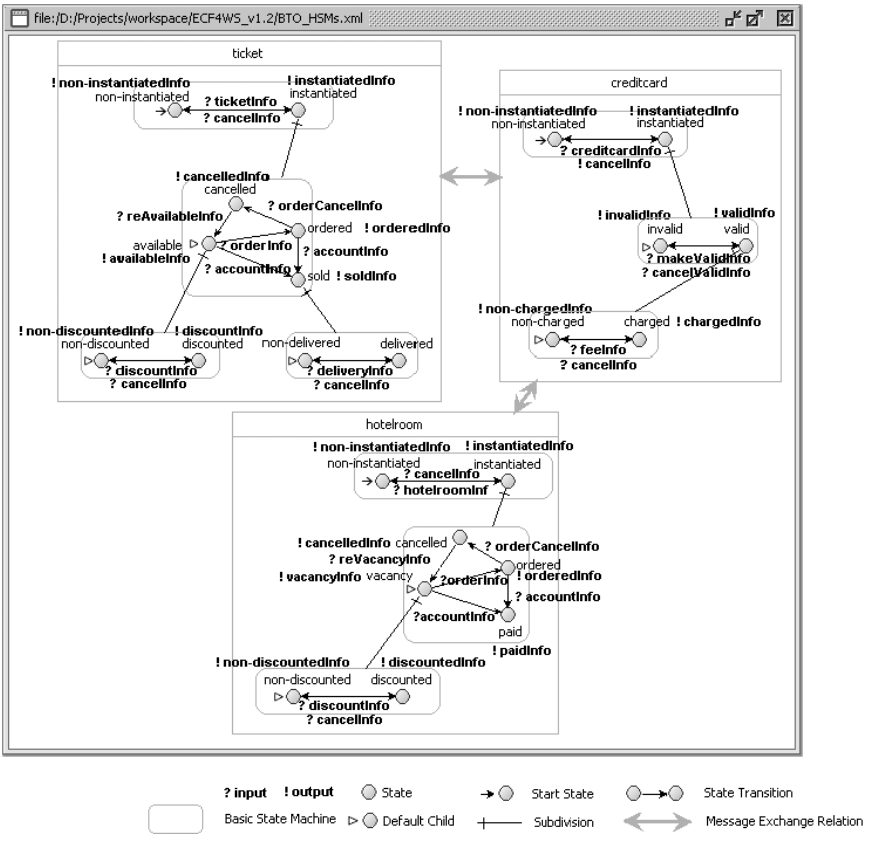


Fig. 1. Scereenshot of domain THSMs of *ticket*, *hotelroom* and *creditcard*

3 Web Service Capability Profile

With the explicit representation of the environment ontology, the capability profile of a Web service can be given. We first define the effect of Web services on their environment entities. An effect on an environment entity is described as a triplet which contains an initial state, a target state and a set of middle states (these middle states will be included in the trace from the initial state to the target state) of this environment entity. Let e be an environment entity. The effect on e can be formulated as: $effect(e) = e : \langle s_i, \mathcal{S}_m, s_t \rangle, s_i, s_t \in e.State, \mathcal{S}_m \subseteq e.State$, in which s_i is an initial state, s_t is a target state and $\mathcal{S}_m$ is a set of middle states ($e.State$ is the set of states in e's domain THSM). The traces from s_i to s_t via $\mathcal{S}_m$ consist of: (1) state transition in a basic state machine, or (2) transition from a state to its default child, or (3) transition from a state to its super-state. For example, an effect that a simple ticket-selling service imposes on environment entity *ticket* can be described as $ticket : \langle available, \{ordered\}, sold \rangle$. Based on

the effect, we can acquire the trace from $hsm(ticket)$ (shown in Fig.1) in BTO: $available \rightarrow ordered \rightarrow sold$. This trace actually is the specific life-cycle description while the ticket-selling service imposes the effect on $ticket$.

A capability profile of Web service is given based on effects of the Web service. It is defined as $\{Ent_{sub}, \mathcal{M}s, effs\}$, in which:

- $Ent_{sub} = \{e_1, ..., e_n\}$ is a set of environment entities that Web service can impose effects on;
- $\mathcal{M}s = \{\mathcal{M}(e_1), ..., \mathcal{M}(e_n)\}$. Each $\mathcal{M}(e_i)$ is partitioned into two subsets: $\mathcal{M}^{in}(e_i)$ and $\mathcal{M}^{out}(e_i)$ for denoting inputs and outputs that Web service needs and produces about the environment entity $e_i (i \in [1, n])$;
- $effs = \{effect(e_1), ..., effect(e_n)\}$ is a set of effects (called "effect set") that Web service imposes on $e_1, ..., e_n$.

Capability specification can be generated by adding rich semantics (i.e., state transitions) automatically to the capability profile from environment ontology.

The effect-based capability profile of *ticket-selling service*(TSA) then can be given. The effect set of TSA is described as $\{ticket{:}\langle available, \{ordered, cancelled\}, sold\rangle\}$. The following is the XML-style capability profile of TSA.

```
<capability Id="ticket-selling service,TSA">
  <entity>BTO:ticket</entity>
  <input ent="ticket">orderInfo,orderCancelInfo,accountInfo,reAvailableInfo</input>
  <output ent="ticket">soldInfo</output>
  <effect ent="ticket">
    <initialState>available</initialState>
    <middleSet>ordered,cancelled<middleSet>
    <targetState>sold</targetState>
  </effect>
</capability>
```

This capability profile describes that *a service TSA which provides ticket selling service, where users can put tickets on hold without being charged, and they also have opportunity to cancel the pending tickets.* Similarly, the capability profiles of TDS, HOS and BAS also can be given using the same structure.

The environment ontology is a sharable knowledge base for Web services. By traversing domain THSM of an environment entity, traces from the initial state to the target state via a set of middle states (i.e., go through an effect $e : \langle s_i, \mathcal{S}_m, s_t\rangle$ on the environment entity e) triggered by a series of inputs can be generated. These traces constitute a THSM (called "specific THSM"). Therefore, each specific THSM is corresponding to an effect on an environment entity.

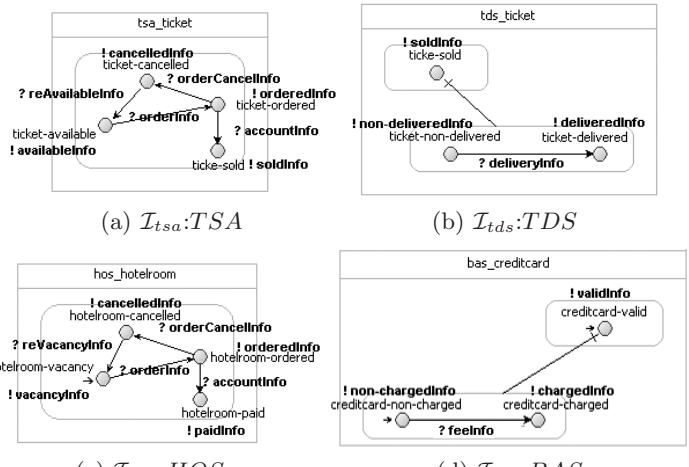
A model $\mathcal{I} = \{\mathcal{K}, inter_k\}$ then is presented, in which $\mathcal{K} = \{k_{e1}, ..., k_{en}\}$ is a set of specific THSMs corresponding to effect set $effs = \{effect(e_1), ..., effect(e_n)\}$, and $inter_k$ contains the set of message exchange relations on $\mathcal{K}$. The model $\mathcal{I}$ is called the *semantic schema* of the effect set $effs$. It is viewed to be a capability specification of Web service. This capability specification is based on a process model and will not be entangled by implementation of Web services. The algorithm for generating the semantic schema of the effect set based on environment ontology is presented in our previous works [2].

Let us look at the effect sets and their semantic schemas of the four elementary services: TSA, TDS, HOS and BAS. They are listed in Table.2.

Table 2. Elementary Effect Sets and Their Semantic Schemas

Ticket-Selling Service (TSA)
$eff s_{tsa} = \{ticket{:}\langle available,\ \{ordered, cancelled\},\ sold\rangle\}$
$\mathcal{I}_{tsa} = \{\{k^{tsa}_{ticket}\}, \phi\}$ Fig.2(a)
Ticket-Delivery Service (TDS)
$eff s_{tds}$=$\{ticket{:}\langle sold, \phi,\ delivered\rangle\}$
$\mathcal{I}_{tds} = \{\{k^{tds}_{ticket}\}, \phi\}$ Fig.2(b)
Hotel Service (HOS)
$eff s_{hos}$=$\{hotelroom{:}\ \langle vacancy,\ \{ordered, cancelled\}, paid\rangle\}$
$\mathcal{I}_{hos} = \{\{k^{hos}_{hotelroom}\}, \phi\}$ Fig.2(c)
Bank Service (BAS)
$eff s_{bas} = \{creditcard{:}\langle valid, \phi, chareged\rangle\}$
$\mathcal{I}_{bas} = \{\{k^{bas}_{creditcard}\}, \phi\}$ Fig.2(d)

Fig.2 is the screenshot showing the semantic schemas $\mathcal{I}_{tsa}, \mathcal{I}_{tds}, \mathcal{I}_{hos}, \mathcal{I}_{bas}$ (k^{tsa}_{ticket} is denoted by tsa_ticket, k^{tds}_{ticket} is denoted by tds_ticket, $k^{hos}_{hotelroom}$ is denoted by hos_hotelroom, and $k^{bas}_{creditcard}$ is denoted by bas_creditcard), which are viewed as the capability specifications of the four elementary services.



(a) $\mathcal{I}_{tsa}$:TSA (b) $\mathcal{I}_{tds}$:TDS

(c) $\mathcal{I}_{hos}$:HOS (d) $\mathcal{I}_{bas}$:BAS

Fig. 2. Scereenshot of semantic schemas of the four elementary services

4 Service Composition by Effect-Based Reasoning

Web service composition is to generate a composite service that consists of a set of elementary services to satisfy a goal. In our approach, Web service composition is conducted by effect-based reasoning.

A goal $\mathcal{G}_{travel}$, which describes a desired service for providing *budget traveling agency* (BTA), is given as a capability profile in XML style.

```
<goal Id="Budget Traveling Agency,BTA">
  <entity>BTO:ticket,BTO:hotelroom,BTO:creditcard</entity>
  <input ent="ticket">orderInfo,orderCancelInfo,deliveryInfo,reAvailableInfo</input>
  <input ent="hotelroom">orderInfo,orderCancelInfo,reVacancyInfo</input>
  <output ent="ticket">deliveredInfo</output>
  <output ent="hotelroom">paidInfo</output>
  <ourput ent="creditcard">chargedInfo</output>
  <effect ent="ticket">
    <initialState>available</initialState>
    <middleSet>ordered,cancelled<middleSet>
    <targetState>delivered</targetState>
  </effect>
  <effect ent="hotelroom">
    <initialState>vacancy</initialState>
    <middleSet>ordered,cancelled<middleSet>
    <targetState>paid</targetState>
  </effect>
  <effect ent="creditcard">
    <initialState>valid</initialState>
    <targetState>charged</targetState>
  </effect>
</goal>
```

This goal profile describes that *"A service BTA which provides ticket selling and hotel service, where user can order ticket and hotel room, and if there is an emergency,user can cancel the pending ticket or hotel room."* In this goal profile, the effect set is described in Table.3.

Table 3. Desired Effect Set and Its Semantic Schema

Budget Traveling Agency (BTA)
$eff s_{bta}$=$\{ticket$:$\langle available, \{ordered, cancelled\}, delivered\rangle, creditcard$: $\langle valid, \phi, chareged\rangle$, $hotelroom$:$\langle vacancy, \{cancelled\}, paid\rangle\}$
$\mathcal{I}_{bta} = \{\{k^{bta}_{ticket}, k^{bta}_{creditcard}, k^{bta}_{hotelroom}\}, inter_{bta}\}$

Given the four elementary services $\mathcal{W} = \{TSA, TDS, HOS, BAS\}$ which are described by the four effect sets $eff s_{tsa}, eff s_{tds}, eff s_{hos}, eff s_{bas}$, a *composition* of $\mathcal{W}$ is a semantic schema $\mathcal{I}$ such that $\mathcal{I}$ delegates all transitions in semantic schemas $\mathcal{I}_{tsa}, \mathcal{I}_{tds}, \mathcal{I}_{hos}, \mathcal{I}_{bas}$ of $eff s_{tsa}, eff s_{tds}, eff s_{hos}, eff s_{bas}$. Given the goal $\mathcal{G}_{travel}$ which can not be satisfied by any elementary services in $\mathcal{W}$, service composition to satisfy the goal $\mathcal{G}_{travel}$ is to check whether there exists composition $\mathcal{I}_{bta}$ of $\mathcal{W}$ which is the semantic schema of $eff s_{bta}$.

Formally, given two semantic schemas $\mathcal{I}_1$=$\{\{k^1_{e1}, k^1_{e2}\}, \{\langle k^1_{e1}, k^1_{e2}\rangle\}\}$ and $\mathcal{I}_2$= $\{\{k^2_{e1}, k^2_{e2}\}, \{\langle k^2_{e1}, k^2_{e2}\rangle\}\}$, semantic schema $\mathcal{I}$ which delegates all transitions in $\mathcal{I}_1$ and $\mathcal{I}_1$ is constructed as $\mathcal{I} = \{\{k_{e1}, k_{e2}\}, \{\langle k_{e1}, k_{e2}\rangle\}\}$, where k_{e1}=$k^1_{e1} \diamond k^2_{e1}$ and $k_{e2} = k^1_{e2} \diamond k^2_{e2}$ (k^i_e denotes a specific THSM of an environment entity e and $\diamond$ is a composition operator). Concretely, the task to obtain desired composition $\mathcal{I}$ of $\mathcal{I}_1$, $\mathcal{I}_2$ can be decomposed to two sub-tasks.

First, we compose specific THSMs in $\mathcal{I}_i (i = 1, 2)$ which are of same environment entity. For example, in the two semantic schemas $\mathcal{I}_{tsa} = \{\{k^{tsa}_{ticket}\}, \phi\}$ (Fig.2(a)), $\mathcal{I}_{tds} = \{\{k^{tds}_{ticket}\}, \phi\}$(Fig.2(b)), k^{tsa}_{ticket} and k^{tds}_{ticket} are of same environment entity *ticket*, we need to check whether there exists target THSM k^{bta}_{ticket}(seeing Table.3) by composing k^{tsa}_{ticket} and k^{tds}_{ticket}.

Second, message exchange relations between these composed specific THSMs are constructed according to sharable domain knowledge, i.e., environment ontology. For example, we suppose that $k^{bta}_{creditcard}$(seeing Table.3) is the composition of $k^{bas}_{creditcard}$ and ϕ (There is not another specific THSM of *creditcard* except for $k^{bas}_{creditcard}$ in elementary semantic schemas). It is described in BTO that a state *valid* in $k^{bta}_{creditcard}$ can trigger a state transition from *ordered* to *sold* in k^{bta}_{ticket}. Therefore, there is a message exchange relation between k^{bta}_{ticket} and $k^{bta}_{creditcard}$.

Back to the example, we present here how to compose the semantic schemas, $\mathcal{I}_{tsa}, \mathcal{I}_{tds}, \mathcal{I}_{hos}$, and $\mathcal{I}_{bas}$, of four elementary services (Ticket-Selling Service(TSA), Ticket-Delivering Service(TDS), Hotel Service (HOS) and Bank Service(BAS), seeing Fig.2) to obtain the desired semantic schema $\mathcal{I}_{bta}$.

First, we present how to check whether there exists target THSM k^{bta}_{ticket} by composing elementary THSMs k^{tsa}_{ticket} and k^{tds}_{ticket}. In [3], D.Berardi has developed a technique for composition of finite state machines in terms of satisfiability of a formula of Deterministic Propositional Dynamic Logic (DPDL). We formulate the problem on *composition existency* of the target THSM k^{bta}_{ticket} based on the Berardi's formula of DPDL: *The DPDL formula Φ_{travel}, built as a conjunction of the following formulas, is satisfiable if and only if there exists the target THSM k^{bta}_{ticket} by composing elementary THSMs k^{tsa}_{ticket} and k^{tds}_{ticket}.*

Concretely, there is a set of atomic propositions $\mathcal{P}_{travel}$ in Φ_{travel}. We have (i) one proposition s_j for each state s_j of k^j_{ticket}, $j \in \{bta, tsa, tds\}$, which is true if k^j_{ticket} is in state s_j; (ii) propositions $moved_j$, $j \in \{bta, tsa, tds\}$, denoting whether k^j_{ticket} performed a state transition. And the transition from a state to its default child state or a state to its super-state are viewed to be a state transition. Formally, s'_d *child* $s \Leftrightarrow s' = \delta(s, \tau)$, where τ denotes a special input that triggers the transition from state s to its default child s'_d. Moreover, s' *child* $s \Leftrightarrow s = \delta(s', \varsigma)$, where ς denotes another special input that triggers the transition from child state s' to its super-state s. And, α denotes an input $\alpha \in \{\tau, \varsigma\} \cup \Sigma^{in} \cup \Sigma^{in}_1 \cup ... \cup \Sigma^{in}_n$, where Σ^{in}_i denotes inputs of each elementary THSM. The master modality $[u]$, which states universal assertions, denotes the closure of inputs. The modality $\langle u \rangle$ states existence assertions. Then, we have the following formulas.

(i) *Formulas capturing the target THSM k^{bta}_{ticket}.*

$$[u](\textit{ticket-available}_{bta} \rightarrow \neg \textit{ticket-ordered}_{bta})$$

This kind of formulas states that k^{bta}_{ticket} can never be simultaneously in the two states $\textit{ticket-available}_{bta}$ and $\textit{ticket-ordered}_{bta}$.

$$[u](\textit{ticket-available}_{bta} \rightarrow \langle orderInfo \rangle true \wedge [orderInfo]\textit{ticket-ordered}_{bta})$$
$$[u](\textit{ticket-sold}_{bta} \rightarrow \langle \tau \rangle true \wedge [\tau]\textit{ticket-non-delivered}_{bta})$$
$$[u](\textit{ticket-non}_{bta}\textit{-delivered} \rightarrow \langle \varsigma \rangle true \wedge [\varsigma]\textit{ticket-sold}_{bta})$$

This kind of formulas encodes the state transitions that k^{bta}_{ticket} can perform. The first formula asserts that if k^{bta}_{ticket} is in state $\textit{ticket-available}_{bta}$ and receives the input $orderInfo$, it necessarily moves to state $\textit{ticket-ordered}_{bta}$. The second formula asserts that if k^{bta}_{ticket} is in state $\textit{ticket-sold}_{bta}$ and receives the special input

τ, it necessarily moves to its default child $\textit{ticket-non-delivered}_{bta}$. It encodes a transition from state $\textit{ticket-sold}_{bta}$ to its default child $\textit{ticket-non-delivered}_{bta}$. Similarly, the third one encodes a transition from state $\textit{ticket-non-delivered}_{bta}$ to its super-state $\textit{ticket-sold}_{bta}$.

$$[u](\textit{ticket-available}_{bta} \rightarrow [\textit{cancelInfo}]\textit{false})$$
$$[u](\textit{ticket-ordered}_{bta} \rightarrow [\tau]\textit{false})$$
$$[u](\textit{ticket-ordered}_{bta} \rightarrow [\varsigma]\textit{false})$$

This kind of formulas encodes the state transitions that are not defined on k^{bta}_{ticket}. The first formula asserts that k^{bta}_{ticket} never move when it is in state $\textit{ticket-available}_{bta}$ and receives the input $\textit{cancelInfo}$. The second formula asserts that $\textit{ticket-ordered}_{bta}$ has no children. The third formula asserts that $\textit{ticket-ordered}_{bta}$ has no super-state.

(*ii*) *Formulas capturing elementary THSMs (e.g.,* k^{tsa}_{ticket}*).*

$$[u](\textit{ticket-available}_{tsa} \rightarrow \neg\textit{ticket-ordered}_{tsa})$$

This kind of formulas asserts that k^{tsa}_{ticket} can never be simultaneously in the two states $\textit{ticket-available}_{tsa}$ and $\textit{ticket-ordered}_{tsa}$, which has an analogous meaning as that relative to k^{bta}_{ticket}.

$$[u](\textit{ticket-available}_{tsa} \rightarrow [\textit{orderInfo}](\textit{moved}_{tsa} \wedge \textit{ticket-ordered}_{tsa} \vee \neg\textit{moved}_{tsa} \wedge \textit{ticket-available}_{tsa}))$$

This kind of formulas asserts that in the composition, either it is k^{tsa}_{ticket} that performs the state transition and therefore it moves to state $\textit{ticket-ordered}_{tsa}$, or the state transition is performed by another elementary THSM, hence, k^{tsa}_{ticket} did not move and remained in the current state $\textit{ticket-available}_{tsa}$.

$$[u](\textit{ticket-available}_{tsa} \rightarrow [\textit{deliveryInfo}]\neg\textit{moved}_{tsa})$$

This kind of formulas encodes the situation when a state transition is not defined. The formula asserts that if the THSM k^{tsa}_{ticket} is in state $\textit{ticket-availabel}_{tsa}$ and it receives an input $\textit{deliveryInfo}$, it dose not move.

(*iii*) *The following formulas must hold for the overall composition.*

$$\textit{ticket-available}_{bta} \wedge \textit{ticket-available}_{tsa} \wedge \textit{ticket-sold}_{tds}$$

It asserts that THSMs $k^{bta}_{ticket}, k^{tsa}_{ticket}$ and k^{tds}_{ticket} start from their start states.

$$[u](\langle \textit{orderInfo} \rangle \textit{true} \rightarrow [\textit{orderInfo}](\textit{moved}_{tsa} \vee \textit{moved}_{tds}))$$

This kind of formulas express that at each step at an elementary THSM moves. The formula asserts that if input $\textit{orderInfo}$ is received, then necessarily a state transition ($\textit{moved}_{tsa}$ or $\textit{moved}_{tds}$) is executed by at least an elementary THSM.

Based on standard Tableau Algorithm [3], we construct the target THSM k^{bta}_{ticket} by composing k^{tsa}_{ticket} and k^{tds}_{ticket} when validating the satisfiability of formula Φ_{travel}. Because that there is only and only one $k^{bas}_{ceditcard}$ which is a specific THSM of *creditcard* in the four elementary semantic schemas, $k^{bta}_{creditcard}$ just is $k^{bas}_{creditcard}$. Similarly, $k^{bta}_{hotelroom}$ also just is $k^{hos}_{hotelroom}$.

$$creditcard\text{-}valid_{bta} \uparrow \langle ticket\text{-}ordered_{bta}, \text{accountInfo}, ticket\text{-}sold_{bta} \rangle$$
$$ticket\text{-}sold_{bta} \uparrow \langle creditcard\text{-}non\text{-}charged_{bta}, \text{feeInfo}, creditcard\text{-}chareged_{bta} \rangle$$

Second, above two message exchanges are described in BTO about environment entities *ticket* and *creditcard*. Therefore, $\langle k^{bta}_{ticket}, k^{bta}_{creditcard} \rangle \in inter_{bta}$. Similarly, we also can get $\langle k^{bta}_{hotelroom}, k^{bta}_{creditcard} \rangle \in inter_{bta}$

Finally, target Web service (BTA, its capability specification is $\mathcal{I}_{bta}$) that satisfies goal $\mathcal{G}_{travel}$ is constructed by composing the semantic schemas of effect sets of elementary services, i.e., $TSA(\mathcal{I}_{tsa})$, $TDS(\mathcal{I}_{tds})$, $HOS(\mathcal{I}_{hos})$ and $BAS(\mathcal{I}_{bas})$. The result $\mathcal{I}_{bta}$ by composing the four elementary services is shown in Fig.3.

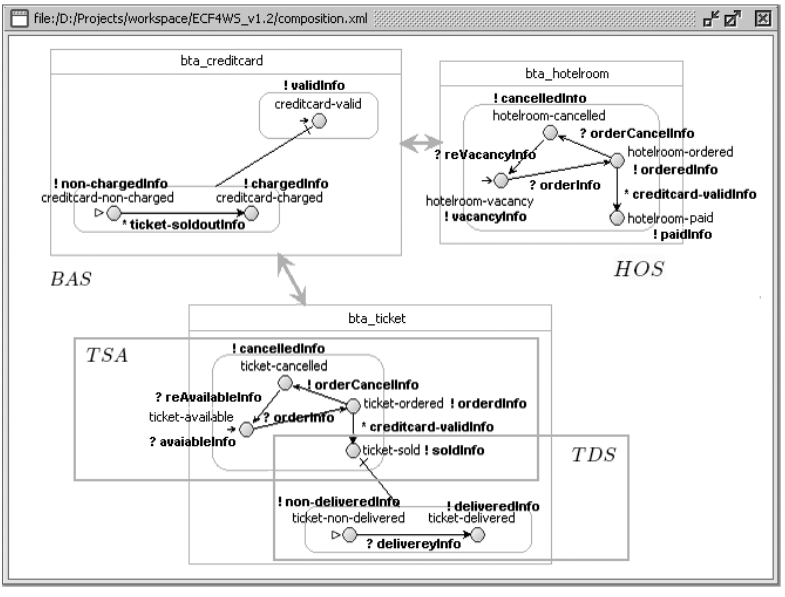


Fig. 3. Composition that satisfies goal $\mathcal{G}_{travel}$ of TSA, TDS, HOS, BAS

Concretely, TSA changes environment entity *ticket* from state *available* to state *sold* via a two middle states $\{ordered, cancelled\}$. At the same time, BAS changes environment entity *creditcard* from state *valid* to state *charged*. The two processes are synchronous by using two message exchanges. output of *creditcard*'s state *valid* triggers state transition of *ticket* from *ordered* to *sold*, and output of *ticket*'s state *sold* triggers state transition of *creditcard* from *non-charged* to *charged*, i.e., ticket is paid by credit card. Sequentially, TDS changes environment entity *ticket* from state *sold* to state *delivered*. In the same way, HOS changes environment entity *hotelroom* from state *vacancy* to state *paid* by using message exchanges with BAS. The goal $\mathcal{G}_{travel}$ can be satisfied.

5 Related Work and Conclusion

There are some available approaches to Web service composition. For the service composition, service description is an important issue. [4] propose an approach to the automated composition of Web services based on a translation of OWL-S service description [5] to situation calculus. Moreover, [6] has proposed a framework which uses a Hierarchical Task Network planner called SHOP2 to compose Web services. SHOP2 provides algorithms for translating an OWL-S service description to a SHOP2 domain. This kind of efforts assumes that Web service is described as interface-based model. However, it is not enough to describe service capabilities.

The OWL-S process model [7] is also proposed for Web service composition. Other representative composition efforts [8,9,10] use Petri net. Moreover, state machine is popularly proposed for representing Web service. In [11], the *Roman* model focuses on *activities* of services, and the services are represented as finite state automata with transitions labeled by these activities. The Conversation model [12] focuses on messages passed between Web services, and again uses finite state automata to model the internal processing of a service. WSMO [13] specifies internal and external choreography of Web services using abstract state machines. This kind of efforts assumes that Web service is described as its local behavior. It is more expressive than interface-based model to describe service capabilities. However, it may be too tied with implementation of services.

[14,15] propose context-oriented approaches that support coordinating and collaborating Web services. Generally, this kind of efforts emphasize the context information in the actual execution of Web services. They also do not involve service capability description at higher-level of abstraction.

Different from the above mentioned approaches, our approach follows the environment-based requirements engineering to specify requirement and elementary services for composition. Instead of focusing on the interface-based model or the process model of Web services, we give regard to effects imposed by the services on their environment entities. The capabilities of Web services are expressed by the traces of optative interactions of the services with the environment entities. Thus, environment entities play the role of collaborating Web services. The main contributions of this paper include that:

- The structured effects of the sharable environment entities, which are modeled by state machines, make the Web service specifications expressive and understandable with each other;
- The effect based capability profile makes the Web service capability specification more meticulous without exposing the Web service's realization details;
- We present that the problem of Web service composition is characterized to be a combination of effects of elementary services on their environment.

In our future work, a logic formalism will be given to express the constraints on Web services. We also will improve the Description Logic-based system for implementing more effective service composition. Moreover, the environment ontology and capability, goal profiles will be related to current popular semantic

description language of Web services, such as OWL-S. We also will focus on how to create and verify the capability and goal profiles.

Acknowledgment

Supported by the National Natural Science Fund for Distinguished Young Scholars of China under Grant No.60625204, the Key Project of National Natural Science Foundation of China under Grant No.60496324, the National 863 High-tech Project of China under Grant No.2006AA01Z155, the National 973 Fundamental Research and Development Program of China under Grant No.2002CB312004, the Knowledge Innovation Program of Chinese Academy of Sciences and MADIS.

References

1. Jin, Z.: Revisiting the Meaning of Requirements. Journal of Computer Science and Technology 22(1), 32–40 (2006)
2. Wang, P., Jin, Z., Liu, L.: An Approach for Specifying Capability of Web Services based on Environment Ontology. In: ICWS 2006, pp. 365–372 (2006)
3. Berardi, D., Calvanese, D., Giacomo, G.D., et al.: Automatic Composition of E-services That Export Their Behavior. In: Orlowska, M.E., Weerawarana, S., Papazoglou, M.M.P., Yang, J. (eds.) ICSOC 2003. LNCS, vol. 2910, Springer, Heidelberg (2003)
4. McIlraith, S., Son, T.C.: Adapting Golog for Composition of Semantic Web Services. In: KR 2002, pp. 482–496 (2002)
5. The OWL Services Coalition: OWL-S: Semantic Markup for Web Services (2004), http://www.daml.org/services/owl-s/1.1/overview/
6. Sirin, E., Parsia, B., Wu, D., Hendler, J., Nau, D.: HTN planning for Web Service composition using SHOP2. Journal of Web Semantics (2004)
7. Pistore, M., et al.: Process-Level Composition of Executable Web Services: On-the-fly Versus Once-for-all Composition. In: Gómez-Pérez, A., Euzenat, J. (eds.) ESWC 2005. LNCS, vol. 3532, pp. 62–77. Springer, Heidelberg (2005)
8. Narayanan, S., McIlraith, S.: Simulation, Verification and Automated Composition of Web Services. In: WWW 2002
9. Hamadi, R., Benatallah, B.: A Petri Net-based Model for Web Service Composition. In: ADC 2003, pp. 191–200 (2003)
10. Yu Tang, et al.: SRN: An Extended Petri-Net-Based Workflow Model for Web Service Composition. In: ICWS 2004
11. Berardi, D., Calvanese, D., Giacomo, G.D., et al.: Automatic Composition of Transition based Semantic Web Services with Messaging. In: VLDB 2005
12. Bultan, T., Fu, X., Hull, R., Su, J.: Conversation Specification: A New Approach to Design and Analysis of E-Service Composition. In: WWW 2003, pp. 403–410 (2003)
13. Fensel, D., et al.: Ontology-based Choreography of WSMO Services. In: WSMO Final Draft, http://www.wsmo.org/TR/d14/v0.4/
14. Maamar, Z., Mostefaoui, S.K., Yahyaoui, H.: Toward an Agent-Based and Context-Oriented Approach for Web Services Composition. In: IEEE Transactions on Knowledge and Data Engineering, May 2005, vol. 17(5), pp. 686–697 (2005)
15. Little, M., et al.: Web Services Context Specification (WS-Context) (April 2007), http://docs.oasis-open.org/ws-caf/ws-context/v1.0/wsctx.html