Automatic Composition of Web Services

N. Guermouche, O. Perrin, C. Ringeissen

LORIA

Réunion COPS 3
Outline

1. Composition Problem
2. Conversational Timed Automata
3. RBAC in web services
Outline

1. Composition Problem

2. Conversational Timed Automata

3. RBAC in web services
Web services:

- Finite state machine $Q = (S, s_0, F, M, A, T)$:
  - $S$: the set of states.
  - $s_0 \in S$: the initial state.
  - $F \subseteq S$: the set of final states.
  - $M$: the set of messages ($!m$ : output message, $?m$ : input message).
  - $A$: the set of actions.
  - $T \subseteq S \times (M \cup A) \times S$

- Have a local store.
- A message queue.
The model:

- A Web services community.
- A goal service.
- The information system.
The problem

The composition problem

How to compose the Web services from the community to satisfy the goal service?
The Web services composition

1. Web services coordination:
   - Try to perform the composition without the involvement of a third party (a mediator).
     1. Community pre-filtering.
     2. Cartesian product.
     3. Post-filtering of the previous cartesian product.

Community pre-filtering:

1. Gather the set of Web services into a set of clusters:
   - Construct pairs of linked web services $Q_1 \Leftrightarrow Q_2$.
   - Merge into one cluster the pairs having a common member $Q_i \leftrightarrow Q_j$.

2. Perform the cartesian product for each cluster.

3. Cartesian product post-filtering (virtual filtering of the unreached states):
   - Remove transitions corresponding to an input message not preceded by the output counterpart.
Web services coordination

Community pre-filtering:

Gather the set of Web services into a set of clusters:

- Constructing pairs of linked web services $Q_1 \leftrightarrow Q_2$.
- Merging into one cluster the pairs having a common member $Q_i \leftrightarrow Q_j$.

Virtual filtering of unreached states

We perform a virtual removing of unreached states in order to be able to check later if the unreached states can be reached thanks to a mediator.
Web services coordination

A filtered cartesian product $Q$ satisfies a goal service $Q_g$ if:

- for each operations and messages trace of the **goal service** $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$ such that $\alpha_i \subseteq M \cup A$ (a message or an action), there exists a trace $\vartheta_0, \alpha_0, \vartheta_1, \alpha_1, \ldots, \alpha_{n-1}, \vartheta_n$ of the **cartesian product** where $\vartheta_j$ is a sequence of messages.
Mediator construction

If no composition can satisfy the goal service:
  ▶ Verify if an eventual construction of a mediator can yield reached states from unreached ones
Outline

1. Composition Problem
2. Conversational Timed Automata
3. RBAC in web services
Timed Modelling in Web Service Composition

- Boualem Benatallah, Fabio Casati, Julien Ponge and Farouk Toumani. *On Temporal Abstractions of Web Services Protocols*
  - Compatibility and replaceability analysis

  - Verification of properties expressed in duration calculus, for a given composition.

Our (ultimate) goal

Consider the composition problem for web services expressed as timed automata.
Web Services as Timed Automata

Start → Search

Login

addToCart

Search → Selection

S<1h, Order, O:=0

Selection → Ordered

Cancelled

Cancelled → Completed

Completed

Ordered

Return

Return → Shipped

Shipped

Shipped → Search

Search

S:=0

S:=0

S:=0

O:=0

O<3h

Cancel

Return

S<7j

Ship

Ship

removeToCart

N. Guermouche, O. Perrin, C. Ringeissen (LORIA)
WS Timed Transition Systems (WSTTS)
Timed automata equipped with a set of clock variables and transitions guarded by constraints over clock variables.
Transitions: annotated by timed constraints and resets of clock variables.
Constraints: true $\mid x \sim c \mid \phi_1 \land \phi_2$ where $\sim \in \{\leq, <, =, \neq, >, \geq\}$, $x \in X$, and $c$ is in a set of time values.

Definition
A WSTTS is a tuple $P = (Q, q_0, F, \Sigma, \delta, Inv)$ such that
- $Q$ is a set of states, $q_0$ is the initial state, and $F \subseteq Q$ is a set of final states
- $\Sigma$ is an alphabet (input message, output message, atomic process, $\epsilon$-transition)
- $\delta \subseteq Q \times \Sigma \times \Phi(x) \times 2^X \times Q$, with an element of the alphabet, a guard and the clocks to be reset
- $Inv : Q \rightarrow \Phi(X)$, denotes invariants associated to states
Example

(Deadline). Automatic transition when a deadline $D$ is reached:

\[ x := 0 \Rightarrow \epsilon, (x > D) \Rightarrow \]

Use of an $\epsilon$-transition.

Remark: Unlike classical automata, $\epsilon$-transitions add to the expressive power of timed automata.

Example

(Interval of validity). An action $a$ is processed within a given amount of time $t$:

\[ x := 0 \Rightarrow a, (x \leq t) \Rightarrow \]
Semantics of WSTTS

Definition

(Semantics of WSTTS). Let $S = (Q, q_0, F, \Sigma, \delta, Inv)$ be a WSTTS. The semantics is defined as a labelled transition $(\Gamma, \gamma_0, \rightarrow)$, where $\Gamma \subseteq Q \times T_C$ is the set of configurations, $\gamma_0 = (q_0, u_0)$ is the initial configuration, and $\rightarrow$ is defined as follows:

1. (Elapse of time) $(q, u) \xrightarrow{\text{tick}} (q, u + \Delta)$ if $u + \Delta \in Inv(q)$, and
2. (Location switch) $(q, u) \xrightarrow{a} (q', u[Y \mapsto 0])$ if there exists $(q, a, \phi, Y, q') \in \delta$ such that $u \in Sol(\phi)$. 

N. Guermouche, O. Perrin, C. Ringeissen (LORIA)
Product of WSTTS

Definition

(Product of WSTTS). Let $S_i = (Q_i, q_{i0}, F_i, \Sigma, \delta_i, Inv_i)$ be a WSTTS built over a set of clocks $X$ and an alphabet $\Sigma$ for $i = 1, \ldots, n$. The product $S_1 \times \cdots \times S_n$ is a WSTTS $(Q, q_0, F, \Sigma, \delta, Inv)$ defined as follows:

1. $Q = Q_1 \times \cdots \times Q_n$
2. $q_0 = q_{10} \times \cdots \times q_{n0}$
3. $F = F_1 \times \cdots \times F_n$
4. The set of switches is defined as follows:
   - atomic process: $(a$ is an atomic process) $(q[q_i], a, \phi, Y, q[q_i \leftarrow q_i']) \in \delta$ if $(q_i, a, \phi, Y, q_i') \in \delta_i$,
   - message: $(q[q_i, q_j], \epsilon, \phi_i \land \phi_j, Y_i \cup Y_j, q[q_i \leftarrow q_i', q_j \leftarrow q_j']) \in \delta$ if $(q_i, ?m, \phi_i, Y_i, q_i') \in \delta_i$ and $(q_j, !m, \phi_j, Y_j, q_j') \in \delta_j$
5. $Inv : Q \rightarrow \Phi(X)$ is defined such that for any $(q_1, \ldots, q_n) \in Q$, $Inv((q_1, \ldots, q_n)) = Inv_1(q_1) \land \cdots \land Inv_n(q_n)$
Composition Problem of WSTTS’s

Definition

(Composition problem of WSTTS’s). Let $S$ be a set of WSTTS’s, $G$ be a goal WSTTS, and $C$ be a client WSTTS. Is there a mediator WSTTS $M$ and $n$ WSTTS’s $S_1, \ldots, S_n$ from $S$ such that

$$C \times M \times S_1 \times \cdots \times S_n \equiv C \times G$$

and the alphabet of $M$ is only made of input and output messages?

Remark

The “equivalence” $\equiv$ should be understood in a broad sense: similar, bisimilar, equivalence of traces, \ldots
Questions

- Do we need invariants? There are no invariants in classical timed automata.
- What about final states? There are no final states in Trento’s framework.
- What about the choice of using $\epsilon$-transitions to express the conversation in the product of WSTTS?
Outline

1. Composition Problem
2. Conversational Timed Automata
3. RBAC in web services
RBAC, OrBAC, sessions

Models for managing autorizations policies

- RBAC: Role-Based Access Control model
- OrBAC: Organization-Based Access control
- Sessions and security policies
The RBAC model

- Centered on roles
  - role is an organizational concept
  - users are assigned to roles
  - permissions are assigned to roles
  - users acquire permissions by being members of roles

- Only positives autorizations (permissions)
  - a permission is a pair action-object

- Concept of session
  - to achieve an action on an object, the user must create a session
  - a session allows for activation/deactivation of roles
  - a session is associated to a single user
Core RBAC

- Different sets: USERS (Users), ROLES (Roles), OPS (Operations), OBS (Objects), SESSIONS (Sessions).
- $UA \subseteq USERS \times ROLES$ (a many-to-many mapping)
- $assigned_users : ROLES \rightarrow 2^{USERS}$
- $PRMS = 2^{OPS \times OBS}$ (Permissions)
- $PA \subseteq PRMS \times ROLES$ (a many-to-many mapping)
- $assigned_permissions : ROLES \rightarrow 2^{PRMS}$
- $Op : PRMS \rightarrow OPS$ (permission-to-operation mapping)
- $Ob : PRMS \rightarrow OBS$ (permission-to-object mapping)
- $user_session : USERS \rightarrow 2^{SESSIONS}$
- $session_role : SESSIONS \rightarrow 2^{ROLES}$
- $avail_session_perms : SESSIONS \rightarrow 2^{PRMS}$ (permissions available to a user in a session)
Other features

- RBAC is hierarchical: $r_1 \succeq r_2$ implies:
  - $\forall u \in \text{USERS}, (u, r_1) \in UA \Rightarrow (u, r_2) \in UA$
  - $\forall p \in \text{PRMS}, (p, r_2) \in PA \Rightarrow (p, r_1) \in PA$
  - $r_1$ is the senior role and $r_2$ is the junior role

- Constraints in RBAC: restriction
  - static constraint: two roles can not be simultaneously assigned to a user
  - dynamic constraint: two roles can not be simultaneously active in a session

- Used in Unix Solaris version 8 or in API Authorization Manager
  - RBAC of Windows Server 2003

N. Guermouche, O. Perrin, C. Ringeissen (LORIA) Automatic Composition of Web Services Réunion COPS 3 25 / 30
GrantPermission is defined as:

\[
GrantPermission(object, operation, role : NAME) \leftarrow \]
\[
(operation, object) \in \text{PERMS}; \ role \in \text{ROLES}
\]
\[
PA' = PA \cup (operation, object) \rightarrow role
\]
\[
\text{assigned\_permissions}' = \text{assigned\_permissions} \setminus \{role \rightarrow \text{assigned\_permissions}(roles)\} \cup \{role \rightarrow (\text{assigned\_permissions}(role) \cup \{(operation, object)\})\}
\]
The OrBAC model

- Centered on organizations
  - roles, views, activities are organizational concepts

Additions
- contextual permissions
- prohibitions, obligations, recommendations
- different security policies specific to an organization (hierarchies)
- security policies of system that includes several organizations
Modeling temporal constraints

Study of the TRBAC/GTRBAC models

- interesting concepts:
  - temporal constraints on activation/deactivation of roles,
  - temporal dependencies between roles activation/deactivation,
  - priorities (conflict resolution),
  - ...

- very complex, ... maybe too complex?
  - temporal constraints everywhere
  - appears to be hard to implement

Concept of session

- a mean to include temporal constraints and access controls
- can be used to limit (specify) the use of a given service
- issues
Sessions

Sessions capabilities

- access rights: roles, actions/messages, permissions, context, objects (views)
- temporal constraints: duration of a session, expiration, activation, ...
- execution flow: number of invocations of a session, sequence of sessions, transfer of the sessions ownership, sub-sections, ...

Open issues

- a formalism to express the capabilities associated to a session
- a formalism to define rules for the session lifecycle (start, termination, etc.)
- composition/mediator generation according to sessions constraints (see the previous algorithm)
- mechanisms to efficiently verify such constraints at run time
A small example

Amazon service
- two sessions, that inherit from a parent session
- permissions and constraints are refined
- orchestrated using dependencies (here temporal dependencies)

Bank service
- conversation between the Amazon and the bank services
- multi-organizations session
- temporal constraints