Verification and Validation of Web Service Compositions

Raman Kazhamiakin
raman@dit.unitn.it
DIT, University of Trento, Italy

Joint work with Marco Pistore, Paritosh Pandya, and Luca Santuari
Context

ASTRO project

• Goal
  • Development of formal techniques and (semi) automated tools that support the design, evolution, and execution of Web service compositions

• Tasks
  • Service composition verification and analysis
  • Service monitoring
  • Service synthesis

• Means
  • BPEL as a Web service description language: business processes and business protocols
  • Formal techniques substantially extended in order to address WS-specific problems (model checking, AI planning and synthesis, etc.)
  • Integrated environment

Any method that prevents the programmer writing code, is a good method

T. Reenskaug
In the beginning...

- Problem domain: **Web service compositions**
  - distributed business processes
  - stateful, long-running component services (e.g., **WS-BPEL** services)

- Goal: analysis of **correctness of the composition behavior**
  - deadlock, livelock freeness
  - behavioral requirements (Message Sequence Charts, LTL properties)

*If the customer makes request, eventually he will receive an offer*

*If the customer cancels the transaction, all the other participants should also cancel*

*Successful termination of the process is possible*
In the beginning...

- Problem domain: **Web service compositions**
  - distributed business processes
  - stateful, long-running component services (e.g., **WS-BPEL** services)

- Goal: analysis of **correctness of the composition behavior**
  - deadlock, livelock freeness
  - behavioral requirements (Message Sequence Charts, LTL properties)

- Initial approach: **model checking**
  - components as State Transition Systems
  - composition as **synchronous** product
  - variables of **finite** ranges, no functions
  - behavior is **timeless**
Outline

- Analysis of communication models
- Data-flow analysis
- Analysis of time-related properties
- Ongoing work and future directions
Outline

- **Analysis of communication models**
  
- Data-flow analysis

- Analysis of time-related properties

- Ongoing work and future directions
Problem #1: Interactions

*Synchronous model of communications is not adequate!*
Virtual Travel Agency

- Provide combined flight and hotel booking service
- Integrate separate **Hotel** and **Flight** booking services
- Participants are represented with their **BPEL** specifications
VTA processes

- **User**
  - invoke (Request)
  - on message (NA)
  - case (NAck)
  - invoke (NAck)
  - on message (Offer)
  - case (Ack)
  - invoke (Ack)
  - receive (Ticket)

- **Flight**
  - receive (F.Request)
  - invoke (F.Request)
  - on message (F.Offer)
  - invoke (F.Offer)
  - on message (F.NAck)
  - invoke (F.NAck)
  - on message (F.Ack)
  - invoke (F.Ack)
  - receive (F.Ticket)

- **Hotel**
  - receive (H.Request)
  - invoke (H.Request)
  - on message (H.NA)
  - invoke (H.NA)
  - on message (H.Offer)
  - invoke (H.Offer)
  - on message (H.NAck)
  - invoke (H.NAck)
  - on message (H.Ack)
  - invoke (H.Ack)
  - receive (H.Ticket)

- **VTA**
  - receive (Request)
  - invoke (Request)
  - on message (NA)
  - invoke (NA)
  - on message (Offer)
  - invoke (Offer)
  - on message (H.NA)
  - invoke (H.NA)
  - on message (H.Offer)
  - invoke (H.Offer)
  - on message (H.NAck)
  - invoke (H.NAck)
  - on message (H.Ack)
  - invoke (H.Ack)
  - receive (Ticket)

**Case**
- (Not Available)
- (Available)
Composition properties

- The composition is **synchronizable**
  - At any moment of time only one component emits a message
  - The receiver is immediately ready to consume the message

- **Synchronous communication model**
  - Components synchronize on shared actions
  - Efficient reasoning techniques
  - Universally used in verification tools for web service compositions

- The synchronous communication model is **adequate** for synchronizable compositions
  - The presence of queues in the implementation does not add new behaviors
VTA Processes – cancellation

User
- invoke (Ack)
- case (No Cancel)
- receive (Ticket)
- on message (YES)
- on message (Ticket)

- invoke (Cancel)

Flight
- receive (F.Ack)

VTA
- on message (Ack)
- invoke (Ack)
- on message (Cancel)
- invoke (F.Ack)
- on message (YES)
- on message (F.Ticket)
- invoke (F.Cancel)
- on message (F.YES)
- invoke (YES)
- invoke (Ticket)
- on message (Ticket)
- on alarm
- receive (F.Cancel)
The synchronous communication model is violated
VTA Processes – cancellation

The real execution is correct
Problem #1: Interactions

Synchronous model of communications is not adequate!

- Wide range of advanced scenarios
  - concurrent emissions, message losses, message reordering
- Complex queue and message management mechanisms

We cannot apply one communication model for all compositions!

- Diversity of middleware and protocol implementations
- Tradeoff between expressiveness and analysis complexity
Our approach [Kazhamiakin, Pistore, Santuari, WWW’06]

- Define a **set of communication models**
  - Different levels of complexity
  - Different interaction mechanisms
  - Common framework

- Given a certain composition scenario determine an adequate communication model
  - Represents all real executions of the composition
  - Preserves behavioral properties

- **Incremental** analysis process
  - From simple to complex communication models
  - Check if the communication model is adequate w.r.t. the scenario
  - If yes, perform the formal verification against this model
Our approach: formal definitions

- Three main ingredients:
  
  - Component services are formally modeled as **State Transition Systems**
  
  - The modalities of the communications are formalized as a **Communication Model**
  
  - The composite behavior of the component services according to a specific communication model is formally described as a **Global State Transition System**
From BPEL to STS

- **State Transition System** $\Sigma = \langle S, S_0, I, O, R \rangle$ where
  - $S$ – finite set of states
  - $S_0$ – set of initial states
  - $I$ – set of input actions
  - $O$ – set of output actions
  - $R \subseteq S \times (I \cup O \cup \emptyset) \times S$ – transition relation

```plaintext
PROCESS Flight
STATES \{Start, switch_IsAvailable, OUT_FNA, SUCCESSS, ...\}
INPUT FRequest, FNack, FAck
OUTPUT FNA, FOffer, FTicket
INIT
    state = Start
TRANS
    Start – [IN FRequest] -> switch_IsAvailable
    switch_IsAvailable – [TAU] -> OUT_FNA
    switch_IsAvailable – [TAU] -> OUT_FOffer
    ...
```
A communication model $\Delta$ is defined by a set of queues $\langle Q_1, Q_2, \ldots, Q_n \rangle$ where each queue $Q_i$ has associated:

- A set of messages $M_i$
- A (finite or infinite) bound $B_i$ on the messages it can contain
- An ordering constraint: ordered or unordered

Allows for the definition of various interaction mechanisms

- Synchronous (1 ordered queue with bound 1)
- Ordered asynchronous (1 ordered unbounded queue for each actor)
- Unordered asynchronous (1 unordered unbounded queue)
- Mixed synchronous/asynchronous, mixed bounded unbounded, ...
Global State Transition System

- A Global State Transition System (GSTS):
  - defines the composite behavior of the system.
  - is parametric w.r.t. a communication model $\Delta$

- A GSTS is a tuple $G = \langle GS, GS_0, A, T \rangle$, where:
  - $GS$ are the global states; each state has the form $gs = (s_1, s_2, ..., s_n, q_1, q_2, ..., q_m)$, where:
    - $s_i$ is the state of the $i$-th component STS
    - $q_j$ describes the content of the $j$-th queue
  - $GS_0$ are the initial global states
  - $A$ are the input-output actions
  - $T \subseteq GS \times A \times GS$ is the transition relation
GSTS: transitions

• The transition relation $T \subseteq GS \times A \times GS$ is defined as follows:
  • If the i-th STS performs an output:
    • update the status of the STS
    • add the message to the associated queue (if the bound allows)
  • If the i-th STS performs an input:
    • consume a message from the associated queue (the queue has to be non-empty!)
    • update the status of the STS
  • If the i-th STS performs a TAU action:
    • update the status of the STS
Hierarchy of communication models

• Relation between models
  • $\Delta_1 < \Delta_2$
    Model 2 simulates model 1 if for any composition scenario $G_{\Delta_1} < G_{\Delta_2}$
  • Defined by the structure of the communication model (bounds, ordering and alphabets)
  • There exists the most general model that simulates any other model

• Hierarchy of communication models
  • Partially ordered set of models with the MG model as top element and the synchronous model as the bottom element

  $\Delta_{\text{sync}} < \ldots < \Delta_{\text{MG}}$
Finding an appropriate model

Given a set of STS, and a communication model $\Delta$, build a reachability graph of the GSTS (DFS algorithm)

- On every state of the search compare the set of enabled transitions with the one under the MG model
- If the sets are different, the model is not adequate

- Efficient analysis algorithm
- The resulting graph is used for further analysis
- On-the-fly boundedness analysis
- Allows for partial order reduction techniques
- Implemented as a part of the Astro verification toolkit
Outline

- Analysis of communication models
- **Data-flow analysis**
- Analysis of time-related properties
- Ongoing work and future directions
Problem #2: Data flow

Data flow should be properly modeled and analyzed!
Loan Approval

Ignoring data affects control flow:

- [amount >= 10000]
- [amount < 10000]
- [risk != low]
- [risk == low]

result := yes

<INVOKE> assess
<INVOKE> approve
Loan Approval

Ignoring data affects control flow:

```plaintext
<INVOKE>
assess
result := yes
<INVOKE>
approve
```

Constraint:

- [amount <= 10000]
- [amount >= 10000]
- [risk != low]
- [risk == low]

Diagram:

[Loan Approval diagram showing control flow and constraints]

Loan Approval

Ignoring data affects control flow:

\[
\text{amount} < 10000 \\
\text{amount} \geq 10000 \\
\text{risk} \neq \text{low} \\
\text{risk} = \text{low}
\]

\[
\text{<INVOKE> assess} \quad \text{<INVOKE> approve} \\
\text{result} := \text{yes}
\]
Loan Approval

Ignoring data affects control flow:

Bad scenario: amount > 10000, approve.result = 'no', assess.risk = 'low', result = 'yes'
Virtual Travel Agency

Incomplete information on service implementations and functions

request.time = offer.time?

Additional assumptions on the internal of the service implementations are needed
Problem #2: Data flow

Data flow should be properly modeled and analyzed!

- **Complex data model** of WS compositions
  - Infinite ranges, custom types, custom functions

- Necessity to manage **information incompleteness**
  - Put additional assumptions on unknown functions/operations

- Often ignored in analysis frameworks
  - Only control-flow analysis
  - Finite data ranges
Our approach \[Kazhamiakin, Pistore, ICWS’06\]

- Extend a composition model with **data flow**
  - Variables, functions, expressions
  - Extended behavioral semantics

- Provide a set of **analysis techniques**
  - Abstraction-based approach
  - Support for **universal** (hold for all system executions) and **existential** (hold for some system executions) properties

- **Iterative** analysis process
  - Allow to put additional constraints on unknown functions
  - Combine the verification and the elicitation of requirements
Our approach: formal definitions

- **Data context:** \(<V, T, F>\) - variables, types, functions
  - **Expressions:** \(E := (t_1 = t_2) \mid \neg E \mid E_1 \text{ or } E_2\), where \(t := x \mid f(t_1, \ldots, t_n)\)
  - **Ground state:** \(g = \{<x, v>\}\) - set of valuations of the variables

- **Extended Transition System** \(\Sigma = <V, S, S_0, I, O, R>\) where
  - \(V\) - finite set of variables
  - \(S\) - set of pairs \(<s, g>\)

- transition relation

\[
S, g \xrightarrow{\varphi} S', g' \quad \text{where } \varphi \text{ is true and action } a \text{ is fired with set of assignments } \{x := t\}\]
The model is infinite...

- How can we define abstractions?
  - Define a **set of propositions** representing certain facts
  - **Valuation of propositions** instead of valuation of variables – finite model

- **Conservative** (branching) **abstraction**
  - Concrete system $C$ is **over-approximated**: when the fact cannot be determined, both states are allowed
  - Allow for more behaviors than the real system
    \[ L(A) \geq L(C) \]
  - Applicable for universal properties but **not for existential** properties
The model is infinite...

- How can we define abstractions?
  - Define a **set of propositions** representing certain facts
  - **Valuation of propositions** instead of valuation of variables – finite model

- **Conservative (branching) abstraction**

```
a:= 10 b:=1
[a>b] a:=a - 5 [!a>b] result:=error
```

\[ P = \{(a>b), !(a>b)\} \]
The model is infinite...

- How can we define abstractions?
  - Define a set of propositions representing certain facts
  - Valuation of propositions instead of valuation of variables – finite model

- **Conservative (branching) abstraction**
  - Interpretation of the valuation: *set of states, where the true propositions evaluate to true, false* - to false
  - Applicability of the transition: *the transition is applicable, if its condition evaluates to true in some state of the interpretation*
  - The effect of the transition: *the resulting valuation is a valuation, obtained by modifying some state of the initial valuation*
The model is infinite...

- How can we define abstractions?
  - Define a **set of propositions** representing certain facts
  - **Valuation of propositions** instead of valuation of variables – finite model

- **Knowledge level abstraction**
  - Concrete system $C$ is **under-approximated**
  - The fact may be either known to be **true**, or **unknown**
  - The safest information on the propositions is used
  - Allow for less behaviors than the real system
    \[ L(C) \geq L(A) \]
  - Applicable for **existential properties**
The model is infinite...

- How can we define abstractions?
  - Define a **set of propositions** representing certain facts
  - **Valuation of propositions** instead of valuation of variables – finite model

- **Knowledge level abstraction**

\[
P = \{(a>b), !(a>b)\}
\]
The model is infinite...

- How can we define abstractions?
  - Define a *set of propositions* representing certain facts
  - *Valuation of propositions* instead of valuation of variables – finite model

- **Knowledge level abstraction**
  - Interpretation of the valuation: *set of states, where the “known” facts are true, “unknown” facts may be true or false*

  - Applicability of the transition: *the transition is applicable, if its condition evaluates to true in all the states of the interpretation*

  - The effect of the transition: *the resulting valuation is the most conservative valuation with respect to the set of facts that can be deduced*
Abstract model

- **Abstract Transition System** \( \Sigma = \langle B, S, S_0, I, O, R \rangle \) where
  - \( B \) – finite set of propositions
  - \( S \) – set of pairs \( \langle s, Val \rangle \), where \( Val \) is a valuation of propositions

- transition relation
Analysis approach

- **Hybrid approach** both models are used
  - B-model: over-approximation, for universal properties
  - K-model: under-approximation, for existential properties

- For the **universal** property (assertion):
  - Prove that B-model satisfies assertion (return **TRUE**)
  - If not, prove that K-model violates assertion (return **FALSE**)
  - If not, refine (return **UNKNOWN**)

- For the **existential** property (possibility):
  - Prove that K-model satisfies possibility (return **TRUE**)
  - If not, prove that B-model violates possibility (return **FALSE**)
  - If not, refine (return **UNKNOWN**)

Implementation

- Preliminary implementation
  - Support for both models
  - Automated abstraction generation
  - Allows for the definition of assumptions on uninterpreted functions
  - NuSMV model checker verification

- Experiments

---

Finite ranges
Scalability
Outline

• Analysis of communication models

• Data-flow analysis

• **Analysis of time-related properties**

• Ongoing work and future directions
Problem #3: Time-related properties

Timed behavior should be properly modeled and analyzed!
authorization for the establishment and operation of a waste disposal or recycling plant

- Settings
  - complex distributed process involving various actors

- Scenario
  - long-term process (~3 months) with time-consuming activities

- Requirements
  - Local (internal constraints) + global (state regulations, normative acts)
  - Functional + timed
This diagram outlines a process involving several participants and stages, with key activities marked by specific durations.

**At most 3 days**
- Initial Request
- Register Protocol
- Evaluate Documents
- Prepare Integration
- Receive Notification
- Provide Integration
- Modify Date

**Within 5 days**
- Preliminary Notification
- Public Notification
- Inverse TC
- Start Procedure
- Verify Reviews
- Conference Call
- Final Call
- Conference

**from 5 to 10 days**
- Collect Integration
- Technical Analysis
- Provide Evaluation
- Send Acts

**Long-term activities:** manual tasks, decisions,...

**Timeouts**
- Receive Acts
- Provide Decision

The diagram highlights the flow of activities between different service centers, with clear indications of time frames for each step.
Complex timed requirements: Constraints, regulations, commitments

- Within 30 days after registration
- Within 30 days after registration
- Within 30 days after registration
- Within 30 days after registration, and at least 10 days before conference
- At least 5 days before conference
- Within 90 days after the 1st call
Problem #3: Time-related properties

Timed properties should be properly modeled and analyzed!

- Timed constructs of WS-* languages
  - BPEL `onAlarm`, `wait` activities
- Time-specific requirements, constraints commitments
  - simple properties: activity `durations`
  - complex properties: constraints on intervals between events, activities, states
Our approach [Kazhamiakin, Pandya, Pistore ARES’06, ICWS’06]

- Extend the **formal model** of the composition with **time**
  - TTS model (close to timed automata network with urgency)
  - formalize BPEL timed constructs and activity durations annotations
  - formalize complex requirements: (subset of) **Duration Calculus**

- Provide a set of **analysis techniques**
  - Verification of timeless properties on timed model
  - Verification of timed properties
  - Computation of timed properties

- Provide a formal **analysis framework**
  - Discrete model of time (finite timers, QDDC [Pandya, RTTOOLS'01])
  - NuSMV symbolic model checking
Our approach: formal definitions

- **Timed Transition System** $\Sigma = \langle X, S, S_0, I, O, R, Inv \rangle$ where
  - $X$ – finite set of **timers**
  - $Inv$ – function that associates **invariants** to states

- Transition relation
Our approach: formal definitions

- **Timed Transition System** $\Sigma = \langle X, S, S_0, I, O, R, \text{Inv} \rangle$ where
  - $X$ – finite set of **timers**
  - $\text{Inv}$ – function that associates **invariants** to states

- **Semantics: composition behavior**
  
  **Time elapsing transition:**
  - global state is not changed
  - all timers synchronously increment

  **Action of some TTS:**
  - transition of some TTS is executed
  - timers are not changes
From BPEL to TTS

• Instant activities

```xml
<invoke operation="op"/>
```

• Duration annotations

```xml
<activity duration="lessEqual(3D)"/>
```

• BPEL timeout

```xml
<pick>
  <onMessage operation="op">…</onMessage>
  <onAlarm for="PT5D">…</onAlarm>
</pick>
```
## Interval Specifications

- **Duration Calculus**
  - Properties over intervals
  - Allows to express complex timed requirements of behavioral specifications

<table>
<thead>
<tr>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>([P]^0)</td>
<td>Single state satisfying propositional formula (P)</td>
</tr>
<tr>
<td>[[P]]</td>
<td>(P) is satisfied in all states of the behavior</td>
</tr>
<tr>
<td>(D_1 \land D_2)</td>
<td>(D_1) is satisfied in the 1(^{st}) subinterval of behavior and (D_2) in the 2(^{nd})</td>
</tr>
<tr>
<td>(D_1 \land D_2)</td>
<td>Interval satisfies both formulae</td>
</tr>
<tr>
<td>(\neg D)</td>
<td>Interval does not satisfy the formula</td>
</tr>
<tr>
<td>(\text{len} \sim c)</td>
<td>The duration of interval is (\sim c)</td>
</tr>
</tbody>
</table>

\[\begin{align*}
[\text{P}]^0 & \quad \text{Single state satisfying propositional formula P} \\
[[\text{P}]] & \quad \text{P is satisfied in all states of the behavior} \\
D_1 \land D_2 & \quad D_1 \text{ is satisfied in the } 1^{st} \text{ subinterval of behavior and } D_2 \text{ in the } 2^{nd} \\
D_1 \land D_2 & \quad \text{Interval satisfies both formulae} \\
\neg D & \quad \text{Interval does not satisfy the formula} \\
\text{len} \sim c & \quad \text{The duration of interval is } \sim c
\end{align*}\]
The conference call should happen within 30 days after the registration and at least 10 days before the conference.

```
[] ( ([registration]0 ^ true ^ [conference]0) -> ( (len ≤ 30) ^ [call]0 ^ (len ≥ 10) )
```
Quantitative analysis

- **Extremal bounds** algorithm
  - Compute minimal/maximal bounds of the intervals, where the property holds
  - Asynchronous versions of the algorithms presented in [Pandya05,Campos et al.96]
  - Symbolic prototype implementation

- Often more effective than verification-based search

<table>
<thead>
<tr>
<th>Property</th>
<th>Time</th>
<th>States</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assertion</td>
<td>5.56sec</td>
<td>2919</td>
</tr>
<tr>
<td>Max</td>
<td>0.28sec</td>
<td>1005</td>
</tr>
<tr>
<td>Possibility</td>
<td>2.28sec</td>
<td>2919</td>
</tr>
<tr>
<td>Min</td>
<td>0.32sec</td>
<td>1005</td>
</tr>
</tbody>
</table>

- **Assertion**: Procedure always terminates within given period
- **Possibility**: It is possible to receive a conference call within given period
Implementation

- [Annotated] BPEL specifications
- WSTTS composition model
- NuSMV Specification
- NuSMV model checker
- Counterexample

Properties to compute (QDDC)
Properties to verify (LTL, QDDC)
Outline

- Analysis of communication models
- Data-flow analysis
- Analysis of time-related properties
- Ongoing work and future directions
Analysis of communication models

- Better **integration** with the **data flow** analysis
  - Currently: conservative analysis results on skeletons, additional verification on complete model

- The role of communication models in the **conformance testing**
  - Validation of **BPEL** compositions against **WS-CDL** specifications [WS-FM’06]
  - Realizability of choreography specifications [FORTE’06]
Ongoing and future work on...

Data-flow analysis

• Better **abstraction-based reasoning** techniques
  • Performance of the generation of K-model
  • Alternative encodings and verifiers

• **Counterexample analysis**
  • How can we extract the missing assumptions and constraints?

• Application to **run-time monitoring**
Ongoing and future work on...

Timed analysis

- Translation **optimizations** and better analysis techniques
  - State space clustering
  - Alternative encodings (mat-sat,...)

- **Alternative encoding**
  - E.g., UPPAAL model checking

- Again, application to **run-time monitoring**
Any questions