Model-based Methods for Linking Web Service Choreography and Orchestration

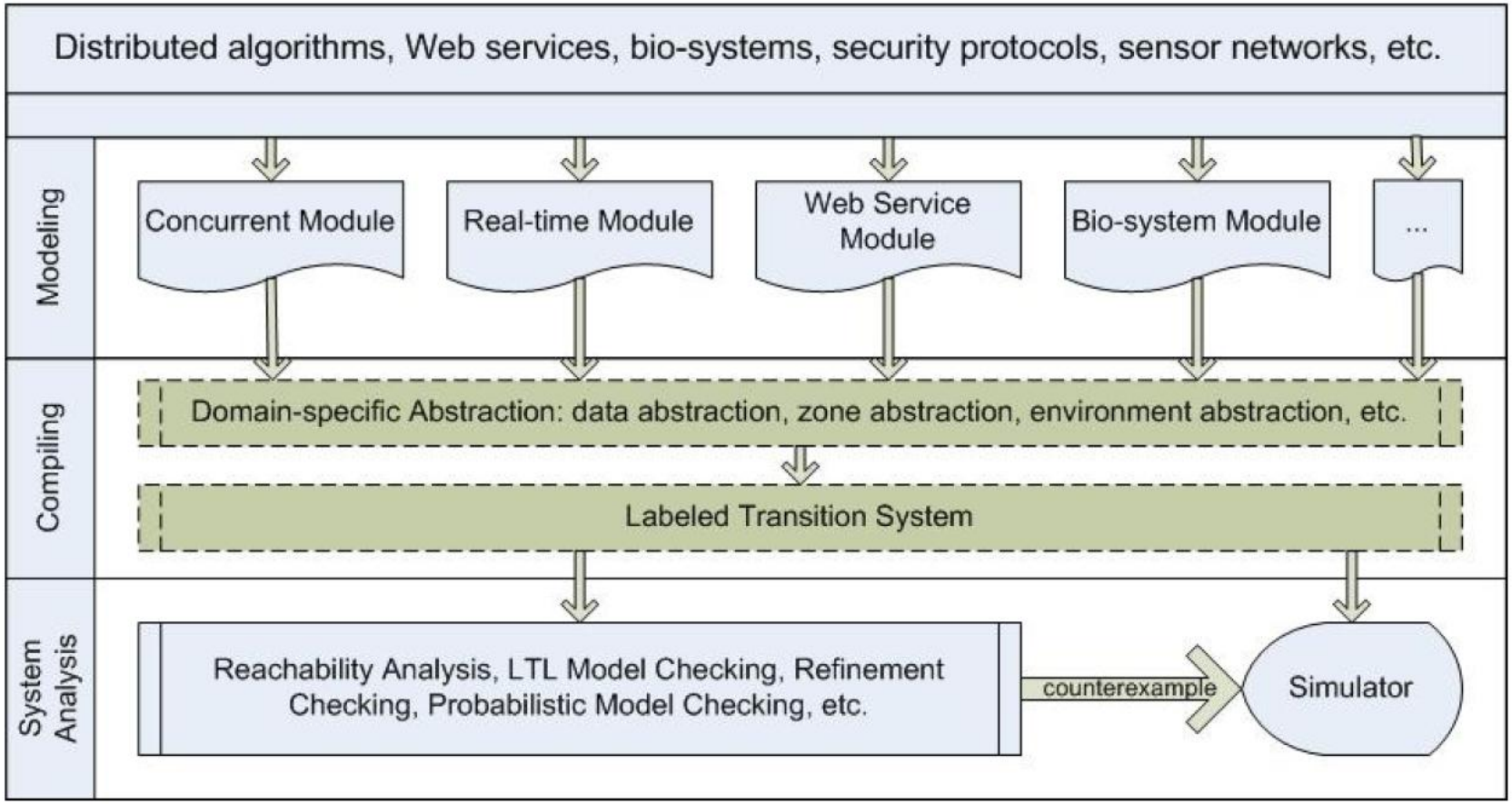
Jun Sun, Yang Liu, Jin Song Dongy, Geguang Pu and **Tian Huat Tan**

Outline

- PAT Introduction and Demo
- Overview of Web Services (WS)
  - Two views of WS
  - Problems addressed
- WS Modeling Languages
- WS Verifications
- Experiments
- Conclusion and Future Works

PAT: Process Analysis Toolkit

PAT is a SPIN-like self-contained environment for system specification, visualized simulation and automated verification.



Contribution

- **Formal Language Proposal** - We propose formal languages for modeling choreography and orchestration respectively with formal operational semantics.
- **Verification** - we provide mechanism to check both choreography and orchestration for
  - Deadlock-freeness, reachability and LTL
  - Whether an orchestration conform to a specific choreography
- **Synthesis** - We synthesize an orchestraction based on choreography if it is implementable. Otherwise, we use a repair process to generate an implementable choreography by inserting communications between providers.

Two Views of Web Services

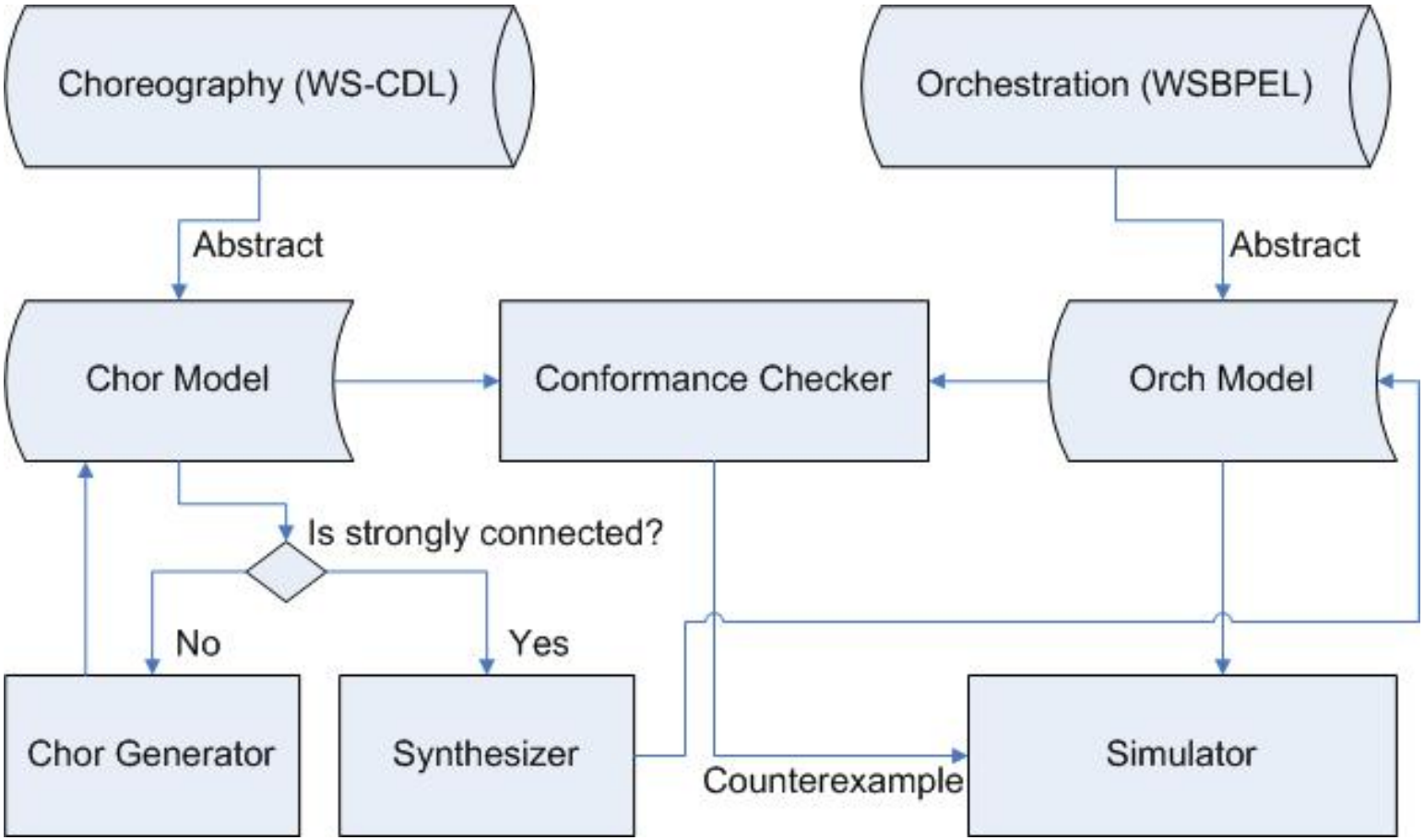
- Web service *choreography* describes collaboration protocols of cooperating Web service participants.
  - A global point of view
  - A contract among multiple corporations, i.e., a specification of requirements
  - May not be executable
  - WS-CDL (Web Service Choreography Description Language)

- Web service *orchestration* is the automated arrangement, coordination, and management of (external) Web Services at the message/execution level
  - A local point of view
  - An orchestration is the composition of concrete services provided by each corporation who realizes the contract.
  - Executable
  - WS-BPEL (Web Service Business Process Execution Language)

Problems Addressed

- Verification
  - Whether a choreography or an orchestration is correct with respect to critical system properties
    - Deadlock-freeness
    - Reachability testing
    - Temporal logic formulae (LTL)
  - Whether they are consistent with each other
    - the orchestration faithfully implements all and only what the contract states.
- Synthesis
  - to decide whether a choreography can be realized faithfully by any orchestration (refereed as implementable) and
  - synthesize a prototype orchestration if possible.

WS Module Workflow



WS Modeling Languages

- Intermediate modeling languages for Web services
  - Languages like WS-CDL or WS-BPEL are designed for machine consumption and therefore are lengthy and complicated in structure
  - Mismatches between WS-CDL and WS-BPEL
  - Intermediate languages focus on the interactive behavioral aspect
  - Our verification and synthesis approaches is not bound to one particular Web service

Choreography Language

$$
\begin{array}{lll}
\mathcal{I} ::= Stop & & \text{– inaction} \\
\quad | \quad Skip & & \text{– termination} \\
\quad | \quad svr(A, B, \tilde{ch}) \rightarrow \mathcal{I} & & \text{– service invocation} \\
\quad | \quad ch(A, B, exp) \rightarrow \mathcal{I} & & \text{– channel commmunication} \\
\quad | \quad x := exp;\ \mathcal{I} & & \text{– assignment} \\
\quad | \quad if\ b\ \mathcal{I}\ else\ \mathcal{J} & & \text{– conditional} \\
\quad | \quad \mathcal{I} \,\square\, \mathcal{J} & & \text{– choice} \\
\quad | \quad \mathcal{I} \,|||\, \mathcal{J} & & \text{– service interleaving} \\
\quad | \quad \mathcal{I};\ \mathcal{J} & & \text{– sequential}
\end{array}
$$

Online Shopping Example

1. $BuySell() = B2S(Buyer, Seller, \{Bch\}) \rightarrow Session();$
2. $Session() = Bch(Buyer, Seller, QuoteRequest) \rightarrow Bch(Seller, Buyer, QuoteResponse.x) \rightarrow$
3. $\quad if\ (x <= 1000)\{$
4. $\quad\quad Bch(Buyer, Seller, QuoteAccept) \rightarrow Bch(Seller, Buyer, OrderConfirmation) \rightarrow$
5. $\quad\quad S2H(Seller, Shipper, \{Bch, Sch\}) \rightarrow$
6. $\quad\quad (Sch(Shipper, Seller, DeliveryDetails.y) \rightarrow Stop$
7. $\quad\quad\ ||| Bch(Shipper, Buyer, DeliveryDetails.y) \rightarrow Stop)$
8. $\quad \}else\{$
9. $\quad\quad Bch(Buyer, Seller, QuoteReject) \rightarrow Session()$
10. $\quad\quad\ \square Bch(Buyer, Seller, Terminate) \rightarrow Stop$
11. $\quad \};$

Semantic Model for Choreography

- A system configuration is a 2-tuple (*I*,*V*)
  - *I* is a choreography and *V* is a mapping from the variables to their values
- Labeled Transition System (LTS) is (*S*,*init*,*T*)
  - *S* is the set of reachable configurations,
  - *init* is the initial state (i.e., the initial choreography and the initial valuation of the variables) and
  - *T* is a labeled transition relation defined by the semantics rule $(\mathcal{I}, V) \xrightarrow{e} (\mathcal{I}', V')$
  - Transition $\langle s_0, e_0, s_1, e_1, \cdots, e_{n-1}, s_n \rangle$
  - Execution $\langle e_0, e_1, \cdots, e_k \rangle$
  - Single Trace $traces(\mathcal{I})$
  - Traces

Orchestration Language

$$
\begin{array}{lll}
P ::= & Stop \mid Skip & \text{– primitives} \\
\mid & inv!\tilde{ch} \rightarrow P & \text{– service invoking} \\
\mid & inv?\tilde{x} \rightarrow P & \text{– service being invoked} \\
\mid & ch!exp \rightarrow P & \text{– channel output} \\
\mid & ch?x \rightarrow P & \text{– channel input} \\
\mid & x := exp;\ P & \text{– assignment} \\
\mid & if\ b\ P\ else\ Q & \text{– conditional branching} \\
\mid & P \,\Box\, Q & \text{– orchestration choice} \\
\mid & P \,\triangle\, Q & \text{– interrupt} \\
\mid & P \,|||\, Q & \text{– interleaving} \\
\mid & P;\ Q & \text{– sequential}
\end{array}
$$

Online Shopping Example

$$
\begin{aligned}
&\textit{Role Buyer } \{ \\
&\quad \textit{var counter} = 0; \\
&\quad \textit{Main}() \quad = \textit{B2S}!\{\textit{Bch}\} \rightarrow \textit{Session}(); \\
&\quad \textit{Session}() \quad = \textit{Bch}!\textit{QuoteRequest} \rightarrow \textit{counter}{+}{+};\ \textit{Bch}?\textit{QuoteResonse}.x \rightarrow \\
&\qquad\qquad \textit{if } (x <= 1000)\{ \\
&\qquad\qquad\quad \textit{Bch}!\textit{QuoteAccept} \rightarrow \textit{Bch}?\textit{OrderConfirmation} \\
&\qquad\qquad\qquad \rightarrow \textit{Bch}?\textit{DeliveryDetails}.y \rightarrow \textit{Stop} \\
&\qquad\qquad \} \\
&\qquad\qquad \textit{elseif } (\textit{counter} > 3)\{\textit{Bch}!\textit{QuoteReject} \rightarrow \textit{Session}()\} \textit{ else } \{\textit{Stop}\}; \\
&\} \\
&\textit{Role Seller } \{ \\
&\quad \textit{var } x \quad = 1200; \\
&\quad \textit{Main}() \quad = \textit{B2S}?\{\textit{ch}\} \rightarrow \textit{Session}(); \\
&\quad \textit{Session}() \quad = \textit{ch}?\textit{QuoteRequest} \rightarrow \textit{ch}!\textit{QuoteResonse}.x \rightarrow (\textit{ch}?\textit{QuoteAccept} \rightarrow \\
&\qquad\qquad \textit{ch}!\textit{OrderConfirmation} \rightarrow \textit{S2H}!\{\textit{ch}, \textit{Sch}\} \rightarrow \textit{Sch}?\textit{DeliveryDetails}.y \rightarrow \\
&\qquad\qquad \textit{Stop} \ \square\ \textit{ch}?\textit{QuoteReject} \rightarrow \textit{Session}()); \\
&\} \\
&\textit{Role Shipper } \{ \\
&\quad \textit{var detail} \quad = \text{“}20/10/2009\text{”}; \\
&\quad \textit{Main}() \quad = \textit{S2H}?\{\textit{ch}1, \textit{ch}2\} \rightarrow \\
&\qquad\qquad (\textit{ch}1!\textit{DelieryDetails}.\textit{detail} \rightarrow \textit{Stop} \ |||\ \textit{ch}2!\textit{DelieryDetails}.\textit{detail} \rightarrow \textit{Stop}); \\
&\}
\end{aligned}
$$

Verifications

- Deadlock-freeness
- Reachability testing
- Temporal logic formulae (LTL)
- Conformance Checking
  - An orchestration *O* is valid implementation of a choreography *I* if and only if *O* refines *I*,
    i.e., *traces*(*O*) ⊆ *traces*(*I*)

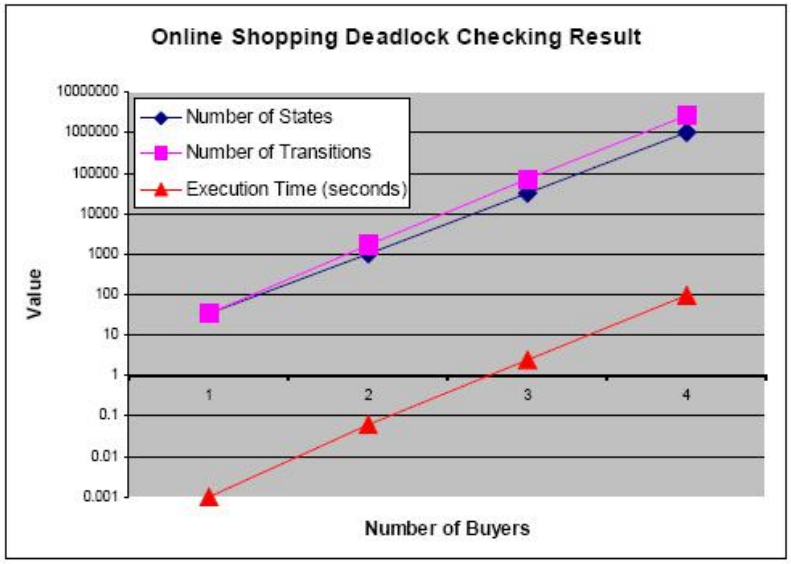
Normalized LTS Let $(S, init, T)$ be a LTS. The normalized LTS is $(NS, Ninit, NT)$ where NS is the set of subsets of S, $Ninit = \tau^*(init)$, and $NT = \{(P, e, Q) \mid P \in NS \wedge Q = \{s : S \mid \exists\, v_1 : P,\ \exists\, v_2 : S,\ (v_1, e, v_2) \in T \wedge s \in \tau^*(v_2)\}\}$.

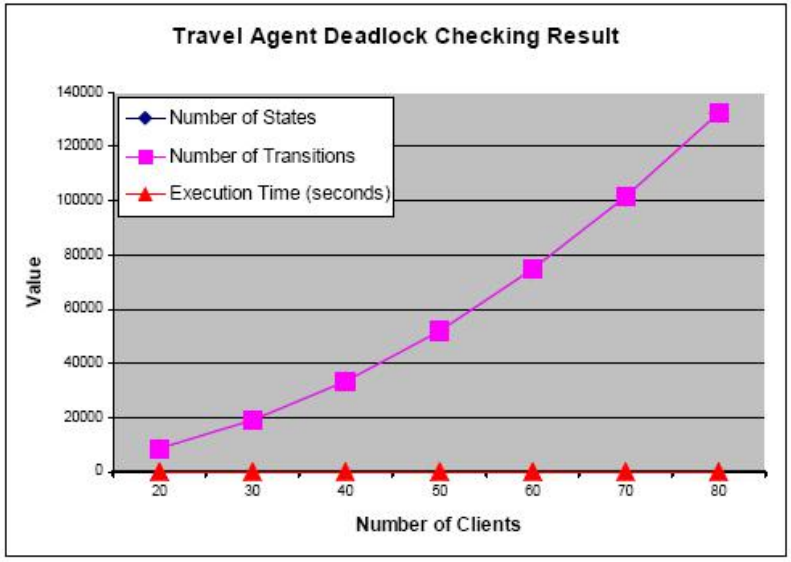
Conformance Checking Algorithm

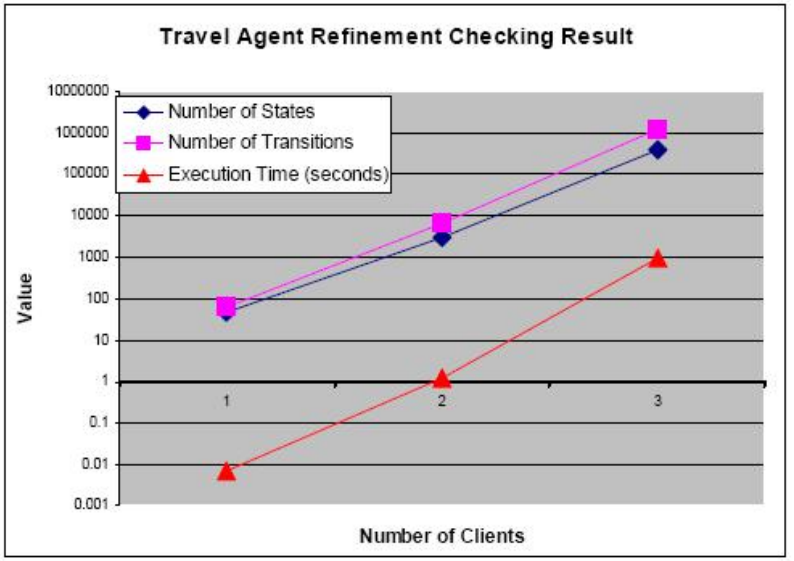
procedure $conformance(\mathcal{O}, \mathcal{I})$

1. $checked := \varnothing;\ pending.push((init_{\mathcal{O}}, \tau^*(init_{\mathcal{I}})));$
2. **while** $pending\ is\ not\ empty$ **do**
3. $\quad (Orc, NChor) := pending.pop();$
4. $\quad checked := checked \cup \{(Orc, NChor)\};$
5. $\quad$ **if** $enabled(Orc) \nsubseteq (enabled(NChor) \cup \{\tau\})$ **then**
6. $\quad\quad$ **return** $false;$
7. $\quad$ **endif**
8. $\quad$ **foreach** $(Orc', NChor') \in \underline{next(Orc, NChor)}$
9. $\quad\quad$ **if** $(Orc', NChor') \notin checked$ **then**
10. $\quad\quad\quad pending.push((Orc', NChor'));$
11. $\quad\quad$ **endif**
12. $\quad$ **endfor**
13. **endwhile**
14. **return** $true;$

Experiments







Conclusion

- A model-based methods for fully automatic analysis of Web service compositions
  - Intermediate languages
  - Verification
  - Synthesis (light-weight)
- Future Works
  - Language enrichment
    - Event handlers, fault handlers and compensation handlers.
  - To support more Web service composition language, e.g., Orc language.
  - Automatic conversions between WS-BPEL/WS-CDL and our language
  - Optimization techniques
    - Candidates include those which are designed for parameterized systems and infinite state systems