Verification of Web Services

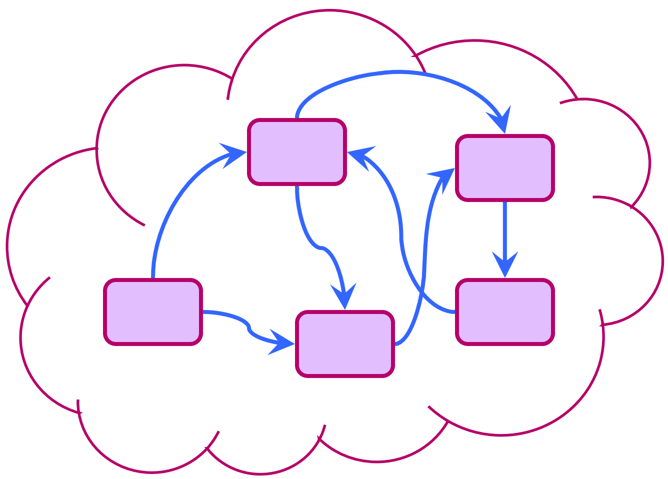
Jianwen Su
University of California, Santa Barbara

The Verification Problem

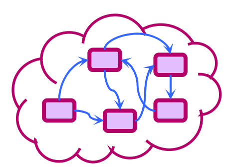
Given

- a web service/composition/choreography/workflow/...
- a goal φ

do all executions satisfy the goal?

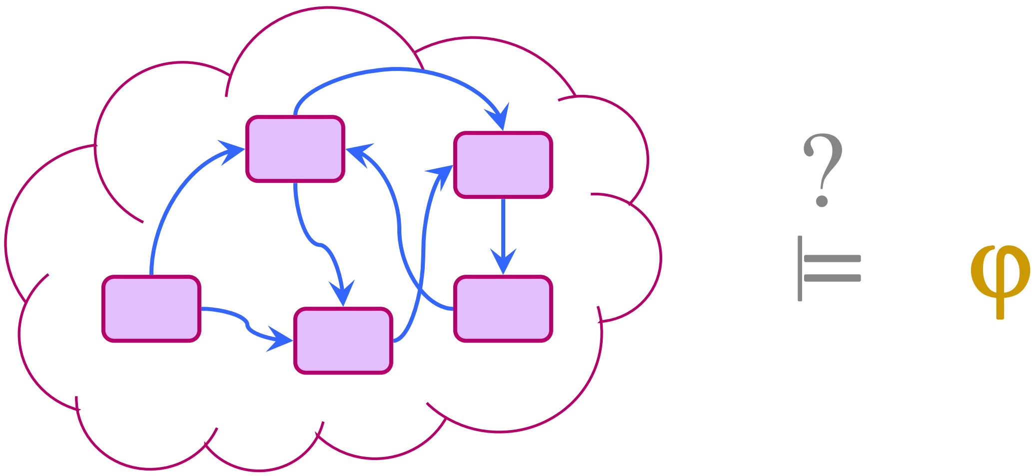


$$\overset{?}{\models} \quad \varphi$$

Choices for  and φ

Outline

- Motivations
- Transitions systems
- BPEL services and compositions
- Choreographies (of BPEL services)
- Artifact-centric workflow
- Concluding remarks



Software Systems in the Real World

- Wide range of applications:
  - Web stores, e-tailors, ...
  - Accounting, financial systems, ...
  - Automated flight control, ...
  - Patient profiles, cases, care records, ...
  - Governments: local, federal, courts, prisons, ...
  - ...
- Challenges:
  - Interoperation & integration
  - Design and analysis
  - Improvements (evolution)

Web Services: Standardization

- The Web: Flexible human-software interaction
- Web services: Flexible software-software interaction
  - SAAS: Software As A Service
- A working definition: software services accessible via standardized protocols
- SOA: a potential basis for software system design, interoperation, integration, …
  - Lots of interest in trade press, academic community, standards bodies, . . .
  - Applications in e-commerce, telecom, science, cloud, government, education, . . .

Fundamental Elements (WS Apps)

- Process: a collection of actions to be taken in a meaningful manner (sequential, parallel, conditional, ...)

- Communication or messages: different software systems need to cooperate, collaborate

- Data: guide the actions to be taken and processes to follow

- Actors (human, external environment): their reasoning for making decisions may not be captured in the logic specification/running systems

Research Challenges (Biz Workflows)

- Models: process, data, messages, actors
- Analysis and verification
- Integration/interoperation
- Improvements
  (biz intelligence, operation optimization, ...)
- Management of workflows and executions

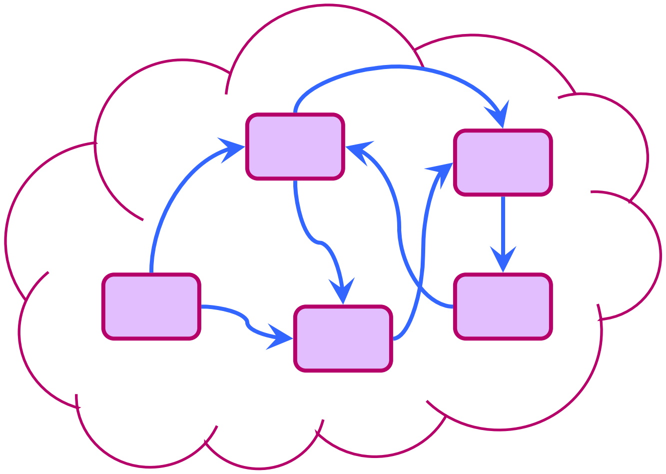
Goal of This Talk

- Focus on analysis & verification problem
  - Depending on models

- A sampler of verification problems, approaches and results

Outline

- Motivations
- Transitions systems
- BPEL services and compositions
- Choreographies (of BPEL services)
- Artifact-centric workflow
- Concluding remarks



$\overset{?}{\models} \quad \varphi$

Transition Systems

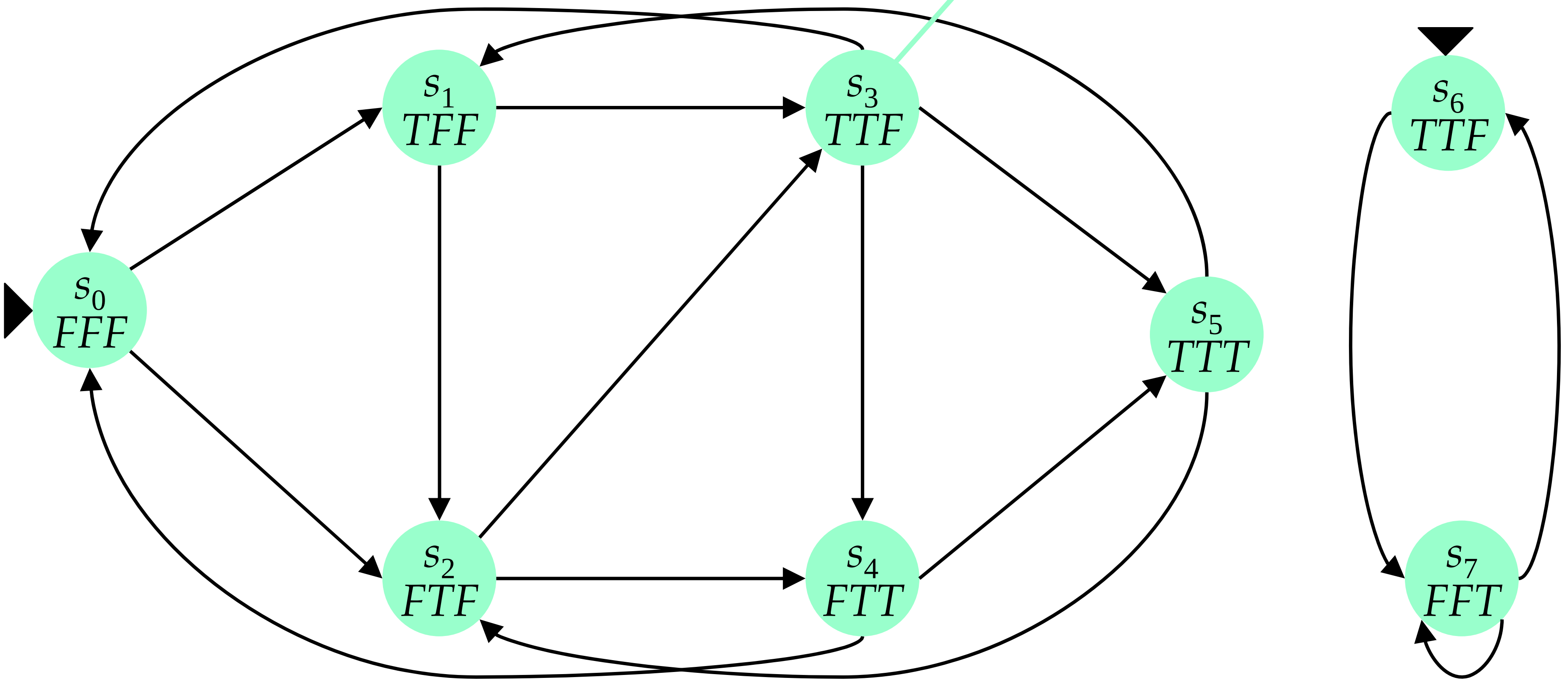
- A finite transition system (Kripke structure) is a tuple $T = (S, I, R, L)$ where
  - a finite set of states S
  - a set of initial states $I \subseteq S$
  - a transition relation $R \subseteq S \times S$
  - a labeling function $L : S \to 2^P$

- P : a set of atomic propositions

Example

- $P = \{q_1, q_2, q_3\}$

$L(s_3) = \{q_1, q_2\}$

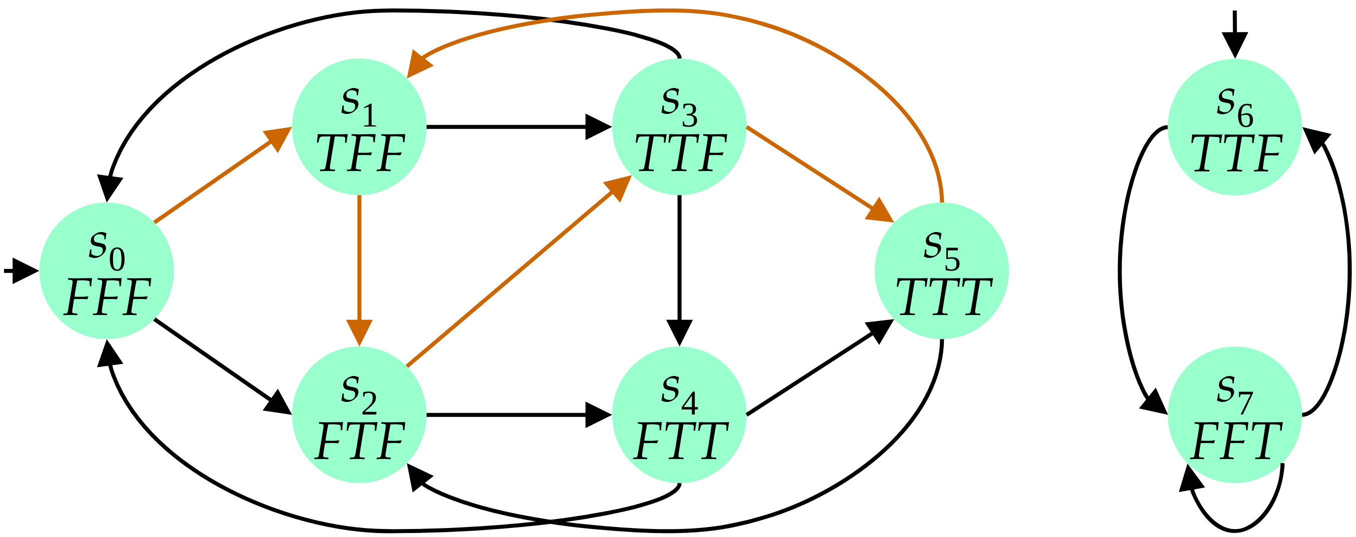


Runs (Execution Paths)

- Given a finite transition system $T = (S, I, R, L)$
- A run is an infinite sequence of states

  $$Z = s_0 s_1 s_2 \cdots$$

  where for each $i \geq 0$, $(s_i, s_{i+1}) \in R$



- $s_0 s_1 s_2 s_3 s_5 s_1 s_2$ …

Linear Temporal Logic (LTL)

- A set P of atomic propositions: $q_1, q_2, q_3, \ldots$
- Logical connectives: $\wedge, \vee, \neg$
- Temporal operators:
  - X φ : φ is true in the next state
  - G φ : φ is true in every state
  - ψ U φ : ψ is true in every state before the state φ is true
  - F φ : φ is true in some future state

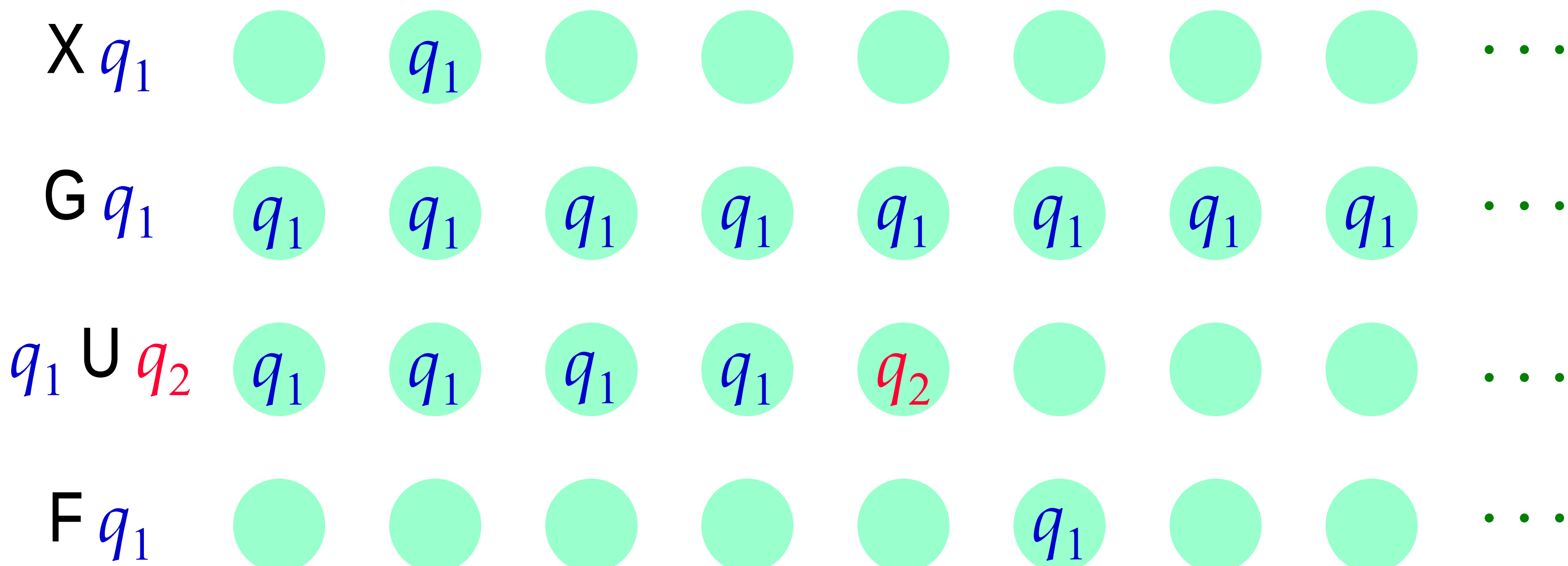
X: next G: always U: until F: eventually

- Example: G $(money \rightarrow$ F $food)$

Semantics of Temporal Operators

- Truth value of a formula is defined on runs
- Propositional connectives have the usual meaning
- Temporal operators:

  X: next G: always U: until F: eventually



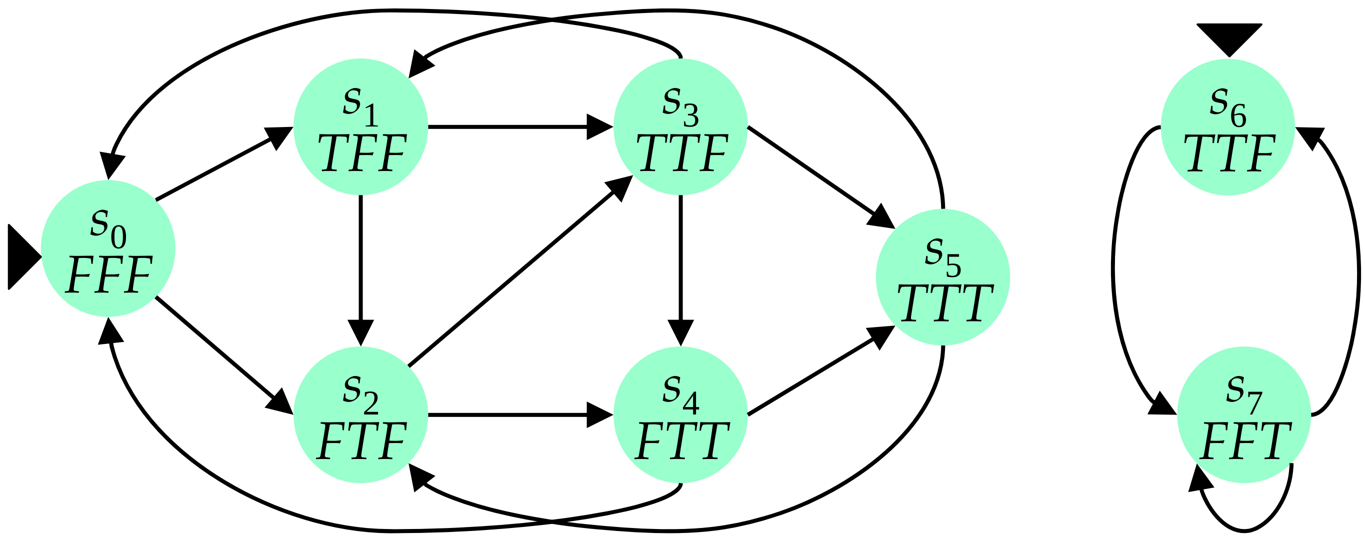
$$\mathsf{F}\, q_1 \equiv \mathit{true}\, \mathsf{U}\, q_1 \qquad \mathsf{G}\, q_1 \equiv \neg \mathsf{F} \neg q_1$$

LTL Semantics

- A state is a set of propositions
- A run $Z = s_0 s_1 s_2 \cdots$ satisfies an LTL formula:
  - $Z \models q$ if $s_0 \models q$ or $q \in L(s_0)$
  - $Z \models \neg\varphi$ if $Z \not\models \varphi$
    $Z \models \varphi \wedge \psi$ if $Z \models \varphi$ and $Z \models \psi$
    $Z \models \varphi \vee \psi$ if $Z \models \varphi$ or $Z \models \psi$
  - $Z \models \mathsf{X}\,\varphi$ if $s_1 s_2 \cdots \models \varphi$
  - $Z \models \mathsf{G}\,\varphi$ if for each i, $s_i s_{i+1} \cdots \models \varphi$
  - $Z \models \mathsf{F}\,\varphi$ if for some i, $s_i s_{i+1} \cdots \models \varphi$
  - $Z \models \psi\,\mathsf{U}\,\varphi$ if for some i, $s_i s_{i+1} \cdots \models \varphi$ and
    for each $j < i$, $s_j s_{j+1} \cdots \models \psi$

Transition Systems and LTL

- A transition system T satisfies an LTL formula φ if every run of T satisfies φ



- $\mathsf{F}\, q_3$

  $\mathsf{G}(\neg q_3 \rightarrow \mathsf{X}\, q_3)$

Verifying LTL Properties

- Problem: given a transition system T, an LTL formula φ, determine if φ is satisfied by T (i.e. every run of T)
- A decision algorithm:
  1. Construct a Büchi automaton $B_{\neg\varphi}$ equivalent to $\neg\varphi$
  2. Explore (depth-first search) simultaneously T and $B_{\neg\varphi}$,
     - if a repeat is found involving a final state of $B_{\neg\varphi}$, halt and output “no” (with the found path)

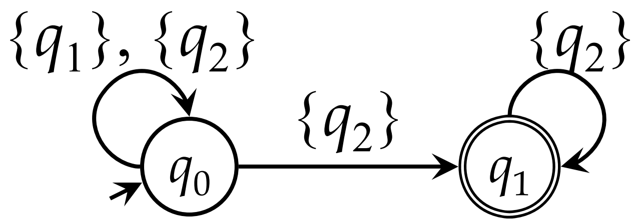
  Otherwise, output “Yes” (T satisfies φ)

Büchi Automata

- P is a (finite) set of propositions
- A Büchi automaton is a tuple $B = (Q, I, \delta, F)$ where
  - ❖ Q is a finite set of states
  - ❖ $I \subseteq Q$ is a (nonempty) set of initial states
  - ❖ $F \subseteq Q$ is a set of final states
  - ❖ $\delta \subseteq Q \times 2^P \times Q$ is a transition relation
- Essentially nondeterministic finite state automata but accepting infinite words:
  - ❖ A word in $(2^P)^\omega$ is accepted if final states are entered infinitely often

The language of B, $L(B)$, is the set of words accepted

An Example



LTL to Büchi Automata

- A Büchi automaton B is equivalent to an LTL formula φ: an infinite sequence Z satisfies φ iff $Z \in L(B)$

- For each LTL formula φ, one can construct a Büchi automaton B_φ equivalent to φ
  - Number of states in B_φ is $2^{O(|\varphi|)}$

- Emptiness of a Büchi automaton can be determined in $O(n)$ where n is the number of states

[Merz MOVEP 2001]

Model Checking

T : a transition system, φ : an LTL formula

1. Construct a Büchi automaton $B_{\neg\varphi}$ equivalent to $\neg\varphi$
2. Explore (depth-first search) simultaneously T and $B_{\neg\varphi}$,
   - if a repeat is found involving a final state of $B_{\neg\varphi}$, halt and output “no” (the trace is the counter example)

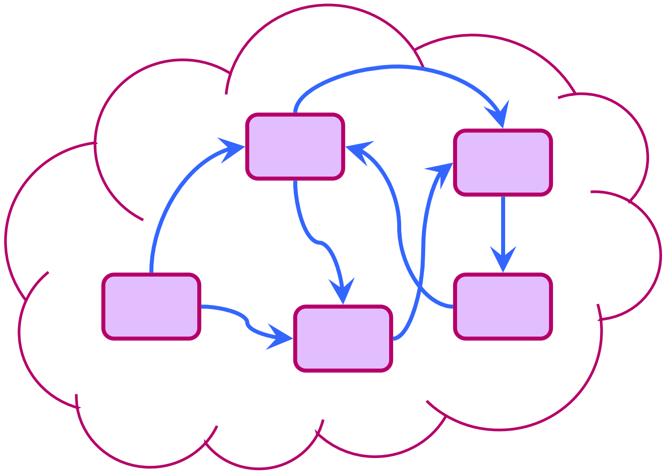
   Otherwise, output “Yes” (T satisfies φ)

- Complexity: $O(2^{O(|\varphi|)}|T|)$ time, PSPACE

[Merz MOVEP 2001]

Outline

- Motivations
- Transitions systems
- BPEL services and compositions
- Choreographies (of BPEL services)
- Artifact-centric workflow
- Concluding remarks



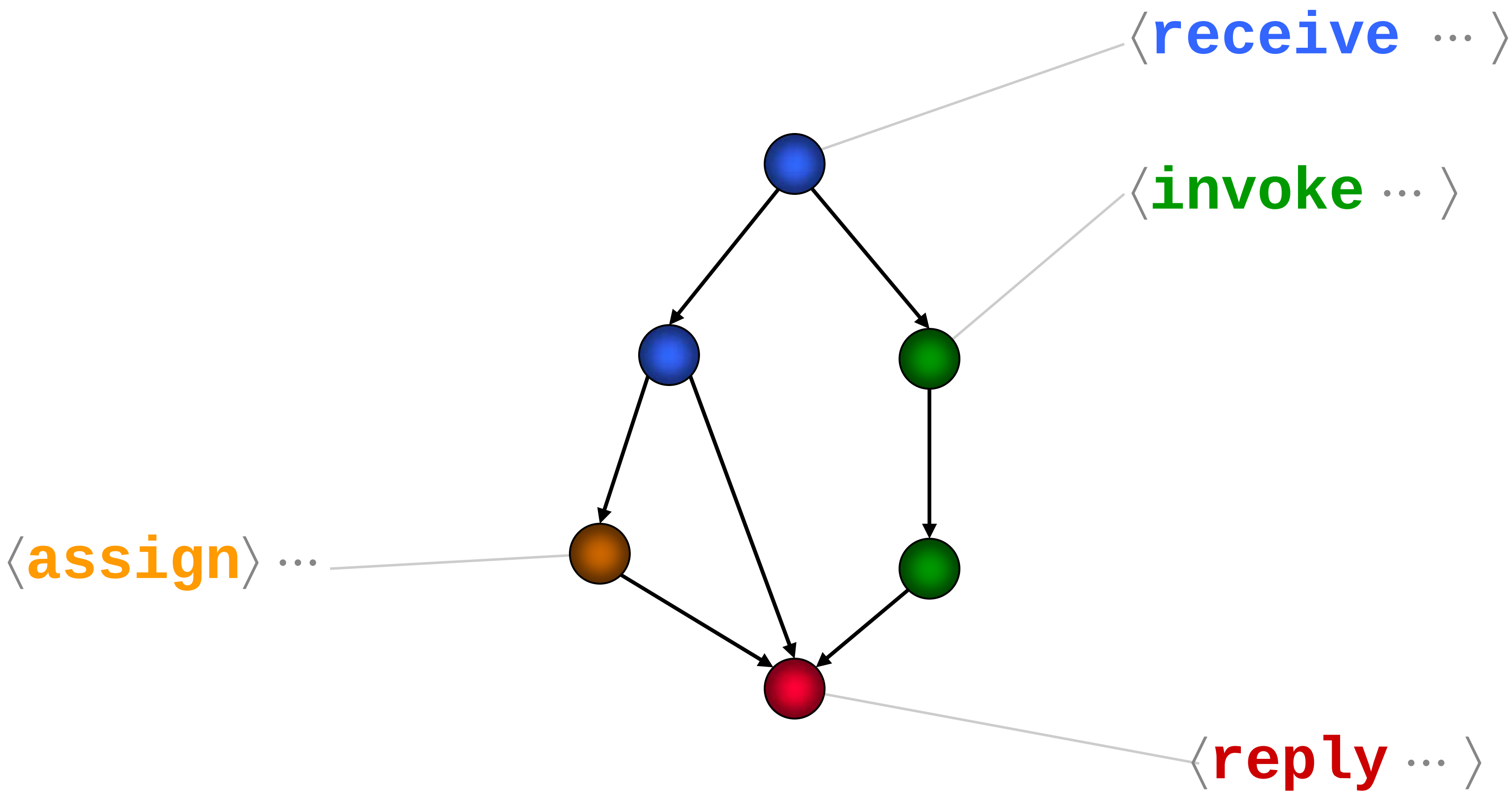
$\overset{?}{\models} \quad \varphi$

Business Process Execution Language

- Allow specification of compositions of Web services
  - business processes as coordinated interactions of Web services
- Allow abstract and executable processes
- Influences from
  - Traditional flow models
  - Structured programming
  - Successor of WSFL and XLANG
- Assumes WSDL ports

- OASIS standard

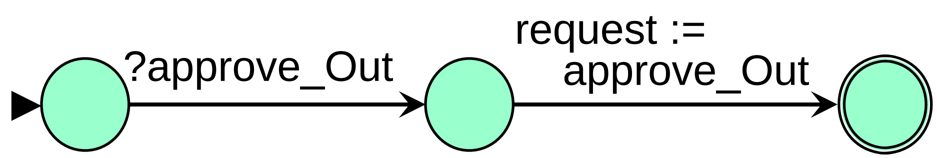
Illustrating a BPEL Service



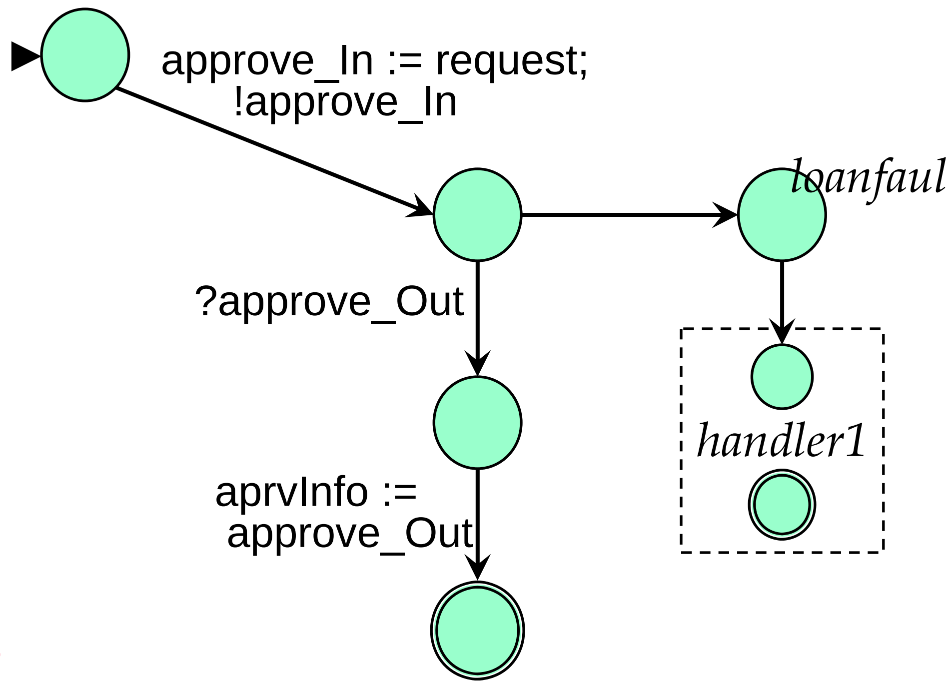
BPEL to Transition Systems [Fu-Bultan-S. WWW '04]

- Translate each atomic activity to a transition system with single entry, single exit

```
⟨receive …
   operation = "approve"
   variable = "request" /⟩
```




```
⟨invoke
      operation="approve",
      invar="request",
      outvar="aprvInfo" ⟩
   ⟨catch  faultname="loanfault"⟩
      ⟨ … handler1 … /⟩
   ⟨/catch⟩
⟨/invoke⟩
```



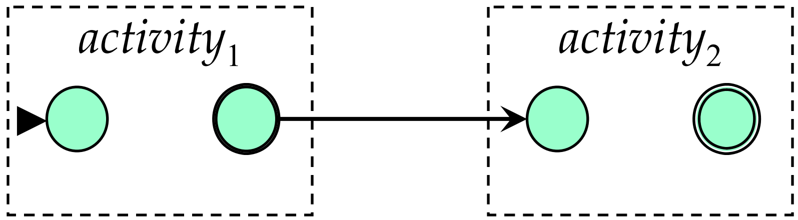
Treat actions as propositions

BPEL to Transition Systems

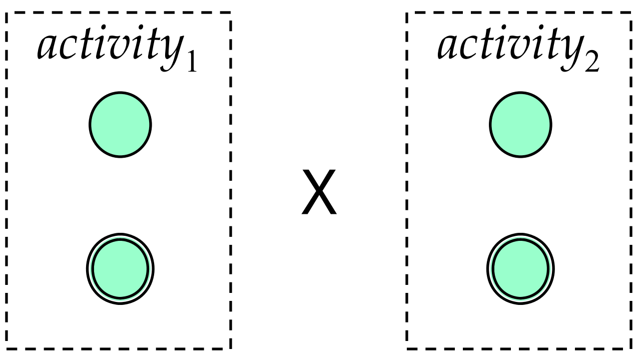
[Fu-Bultan-S. WWW ’04]

- Control flow constructs: assemble pieces of transition systems

⟨**sequence**⟩
  ⟨… $activity_1$ …/⟩
  ⟨… $activity_2$ …/⟩
⟨/**sequence**⟩



⟨**flow**⟩
  ⟨$activity_1$ …⟩
    ⟨source linkname = “link1”…/⟩
  ⟨/$activity_1$⟩
  ⟨$activity_2$ …⟩
    ⟨target linkname = “link1”/⟩
  ⟨/$activity_2$⟩
⟨/**flow**⟩



disallow the orders prohibited by the link

Verifying BPEL Services

- S: a BPEL service, P: a set of propositions,
  φ: an LTL formula
- Determine if every execution of S satisfies φ
- Algorithm:
  1. Construct a transition system $T_{S,P}$
  2. Determine if $T_{S,P}$ satisfies φ
- Complexity: $O(2^{O(|\varphi|)}|S|)$ time
- Good news but
  - Control states (flow) only, no variables/data
  - Single service, no composition

Adding Data

- BPEL allows variables to hold XML documents

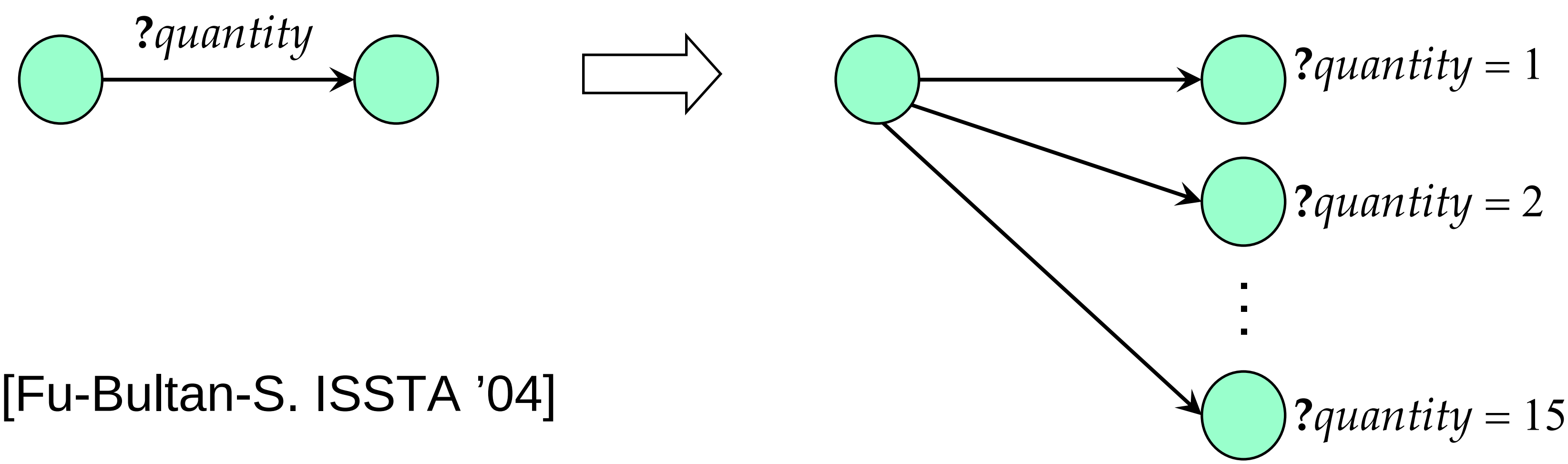
- Bad news (folklore):
  BPEL is Turing (computationally) complete

- Immediate consequence:
  It is undecidable if a given BPEL service satisfies a given LTL formula

- One possible restriction: limit variables to
  - finite domains: the number of possible values is finite

Finite Domain Variables

- Represent variable contents explicitly through states

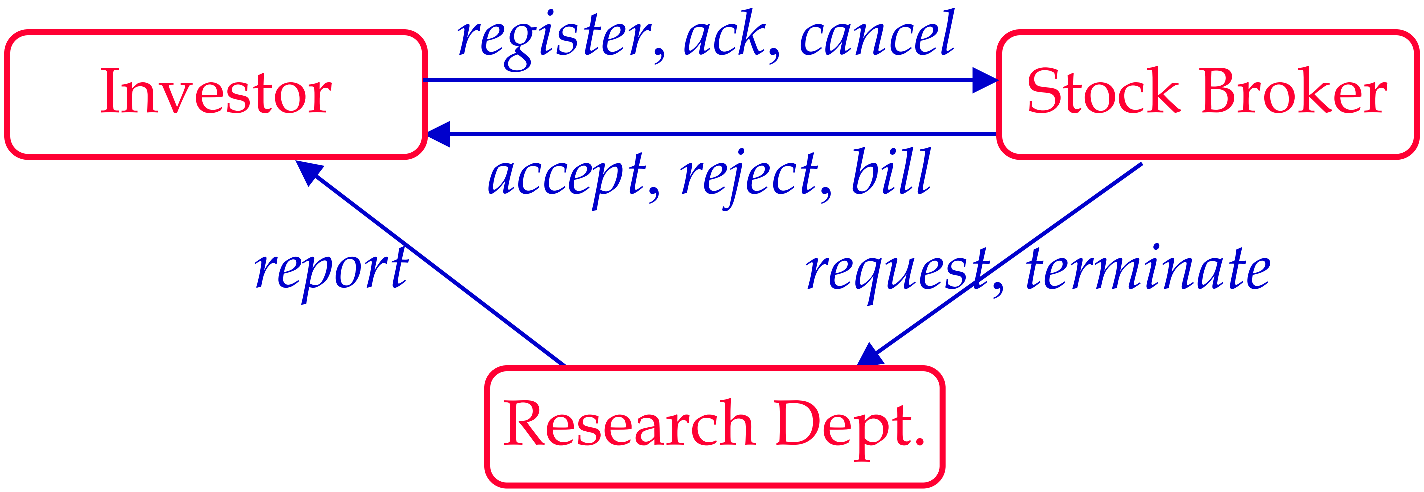


[Fu-Bultan-S. ISSTA ’04]

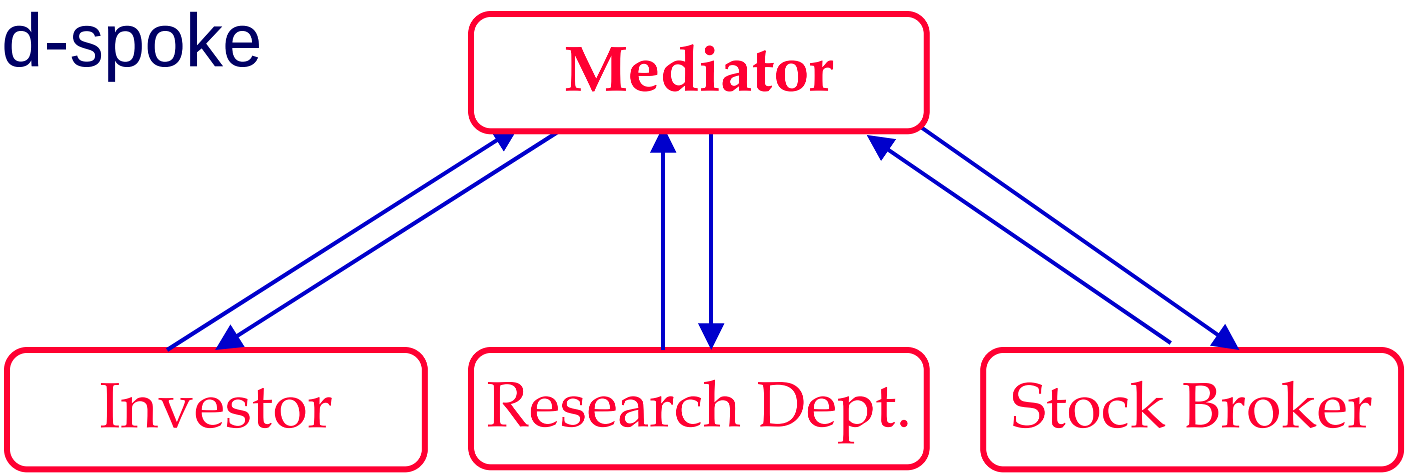
- Transition states increased by n^m times:
  n : (max) domain size, m : number of variables
- Complexity of verification: $O(2^{O(|\varphi|)}|S|n^m)$ time
  φ : LTL formula, S : BPEL service

Composition of BPEL Services

- Peer to peer

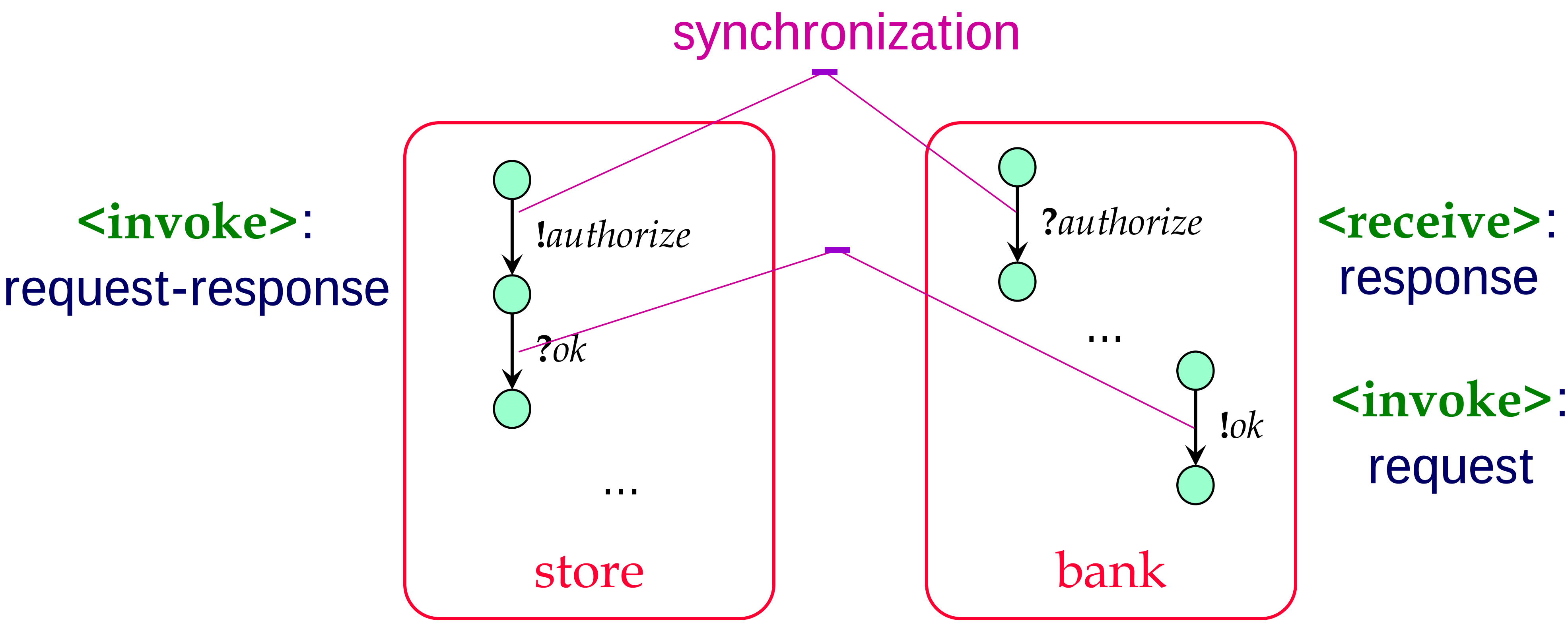


- Mediated or hub-and-spoke



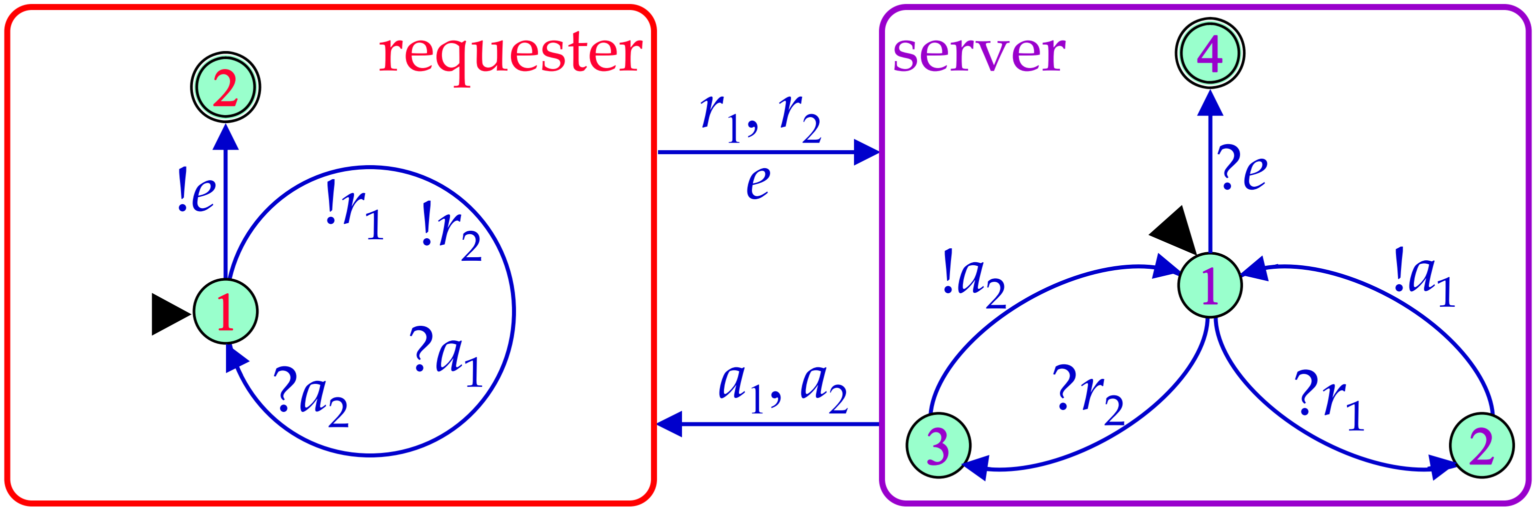
Synchronous Messaging Model

- Two specific actions:
  - Send a message (!)
  - Receive a message (?)

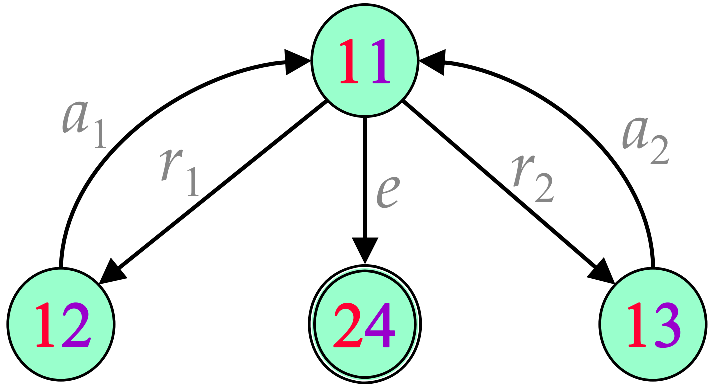


Product with Synchronous Messaging

- Two services



- Their synchronous product as a transition system:



Product with Synchronous Messaging

- In general, the composition of k BPEL services with synchronous messaging can be modeled as a transition system with r^k states where
  - r is the (max) number of states in a single service

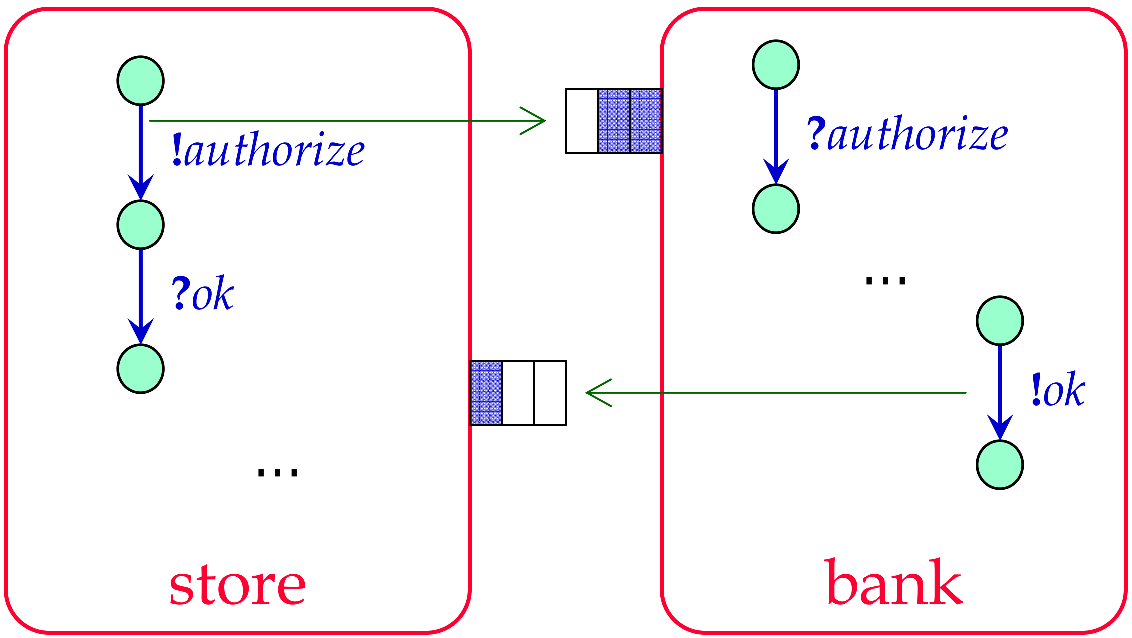
- Complexity of verification: $O(2^{O(|\varphi|)}(|S|n^m)^k)$ time
  - φ : LTL formula
  - $|S|$: size of a BPEL service
  - n : domain size
  - m : number of variables in a BPEL service
  - k : number of BPEL services

Asynchronous Messaging

[Bultan-Fu-Hull-S. WWW ’03]

- Two specific actions:
  - Send a message (!)
  - Receive a message (?)

⟨**invoke**⟩: request-response



⟨**receive**⟩: response

⟨**invoke**⟩: request

- FIFO queues are used to buffer unconsumed messages
  - One queue per service for incoming messages

Verification is Undecidable

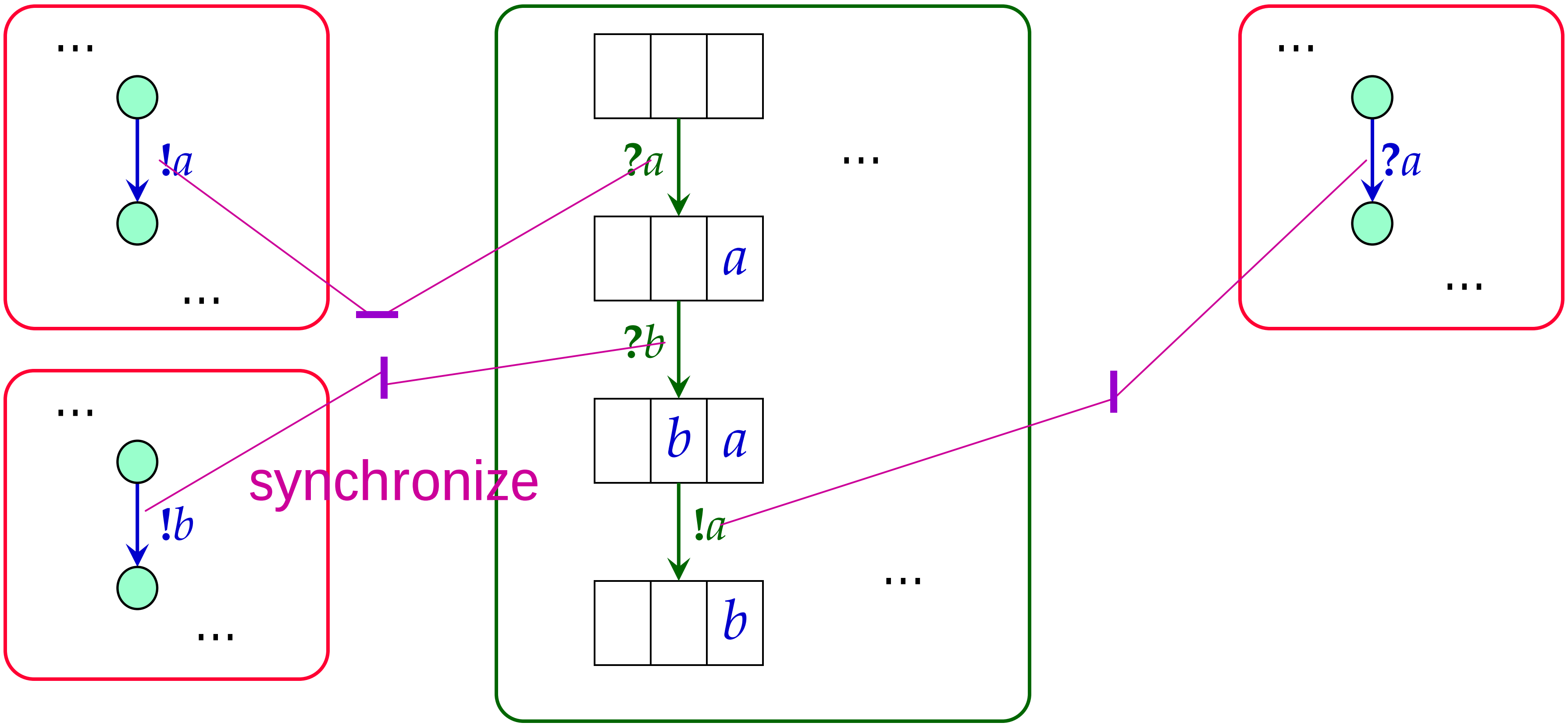
- Finite state automata with FIFO queues are Turing complete [Brand-Zafiropulo JACM'83]

- Immediate consequence:
  Verification problem is undecidable

- One possible restriction: bound queue size

Bounded Queues & Finite State Automata

- Observation: a bounded length queue has a finite number of states



- Asynchronous + bounded queue can be simulated
  - ❖ Note: Only focus on message types not content

BPEL with Asynchronous Messaging

- Number of states for queues: e^l, where
  e : number of message types, l : queue length bound
- With message contents: $e^l n^l$, where n is domain size

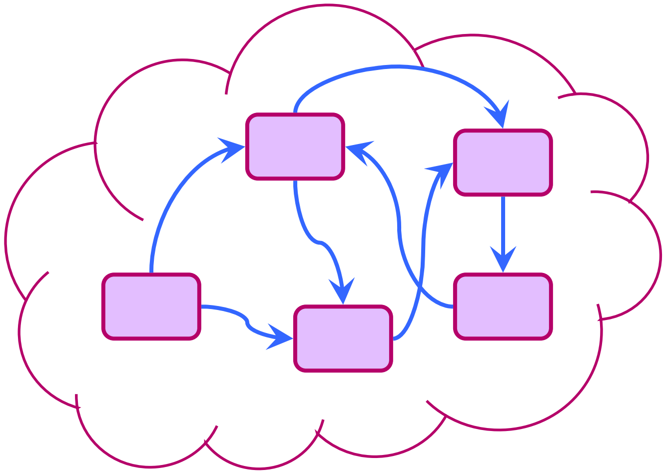
- Complexity of verification: $O(2^{O(|\varphi|)}(|S|n^m e^l n^l)^k)$ time
  - ❖ φ : LTL formula
  - ❖ $|S|$: size of a BPEL service
  - ❖ n : domain size
  - ❖ m : number of variables in a BPEL service
  - ❖ k : number of BPEL services

Summary of Verifying BPEL Services

- Focus on decidability boundary of LTL properties of BPEL + compositions (synchronous, bounded queue asynchronous messaging)
- Verification algorithms: map to exiting verifiers
  - Model checker: SPIN [Fu-Bultan-S. 2003-4] [Nakajima 2004], [Pistore-Traverso-et al 2005]
  - Process algebras: LTSA [Foster-Uchitel-Magee-Kramer 2003], CWB [vanBreugel-Koshkina 2004] [Salaun-Bordeaux-Schaef 2004], LOTOS [Ferara 2004][Salaun-Ferara-Chirichiello 2004]
  - ASM: [Farahbod-Classer-Vajihollahj 2004][Deutsch-Sui-Vianu 2004] [Fahland-Reisig 2005]

Outline

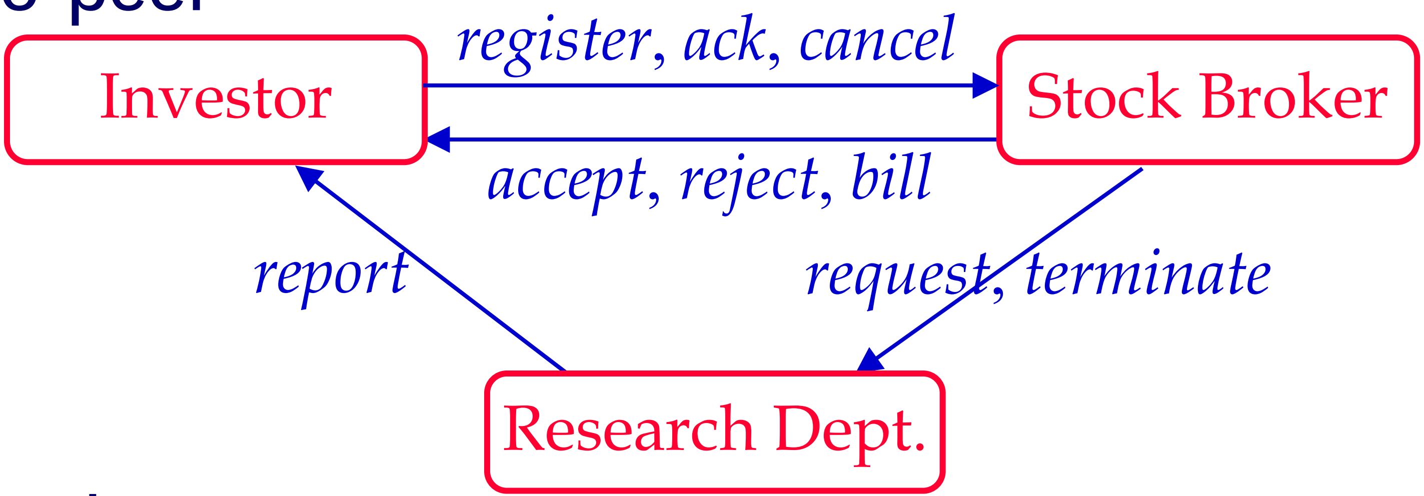
- Motivations
- Transitions systems
- BPEL services and compositions
- Choreographies (of BPEL services)
- Artifact-centric workflow
- Concluding remarks



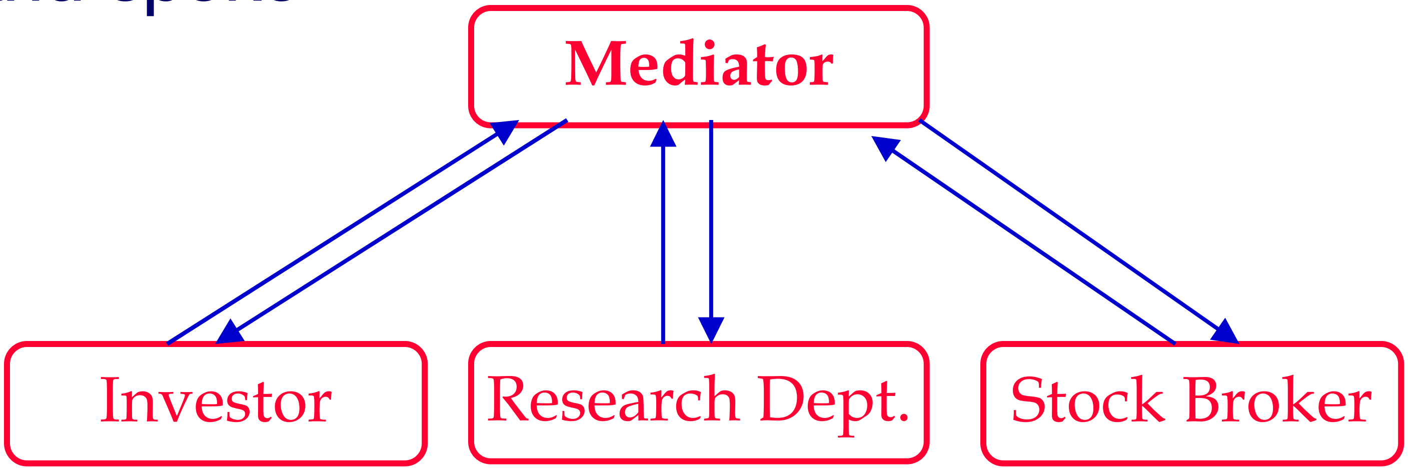
$\overset{?}{\models} \quad \varphi$

Composition: Common Topologies

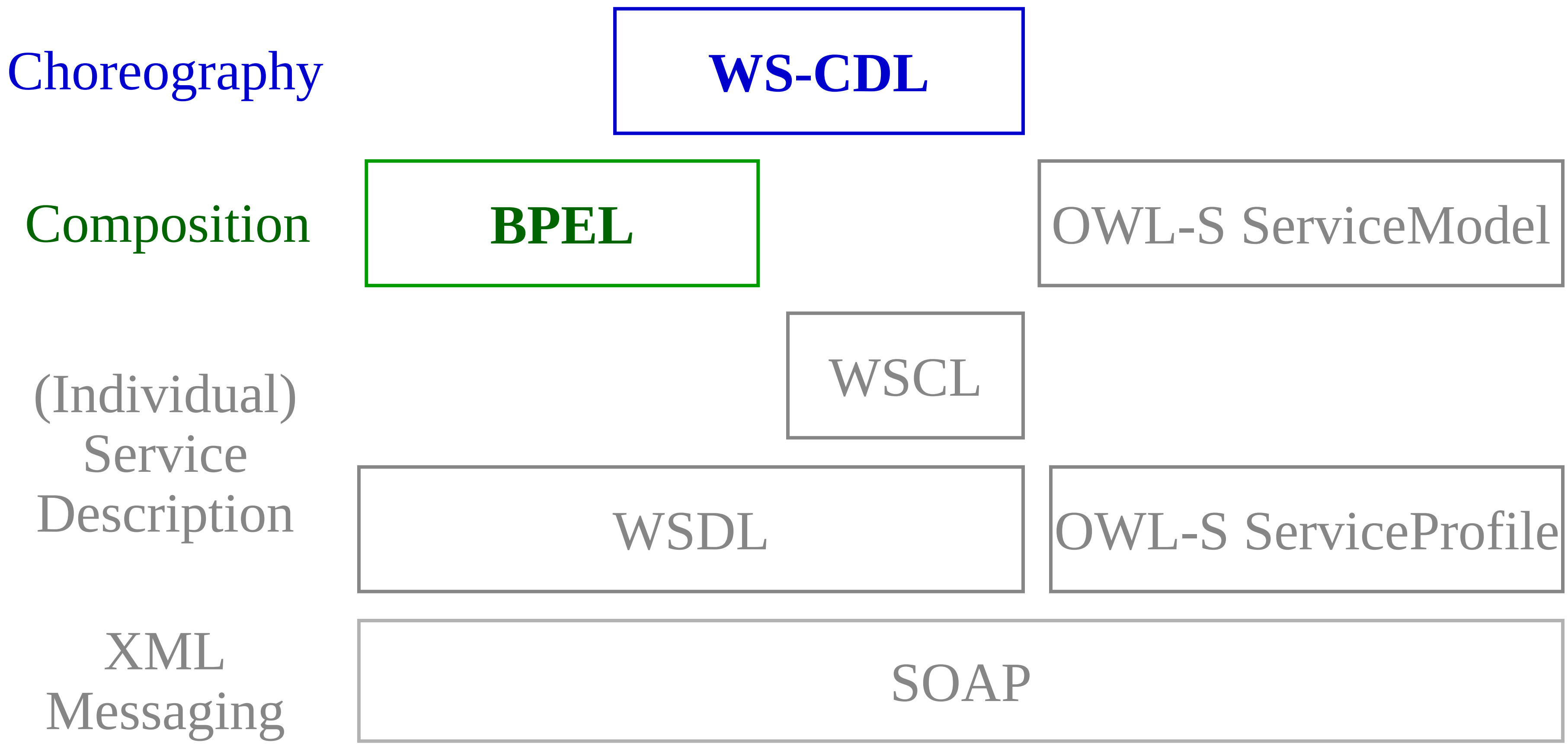
- Peer-to-peer



- Mediated, or "hub and spoke"



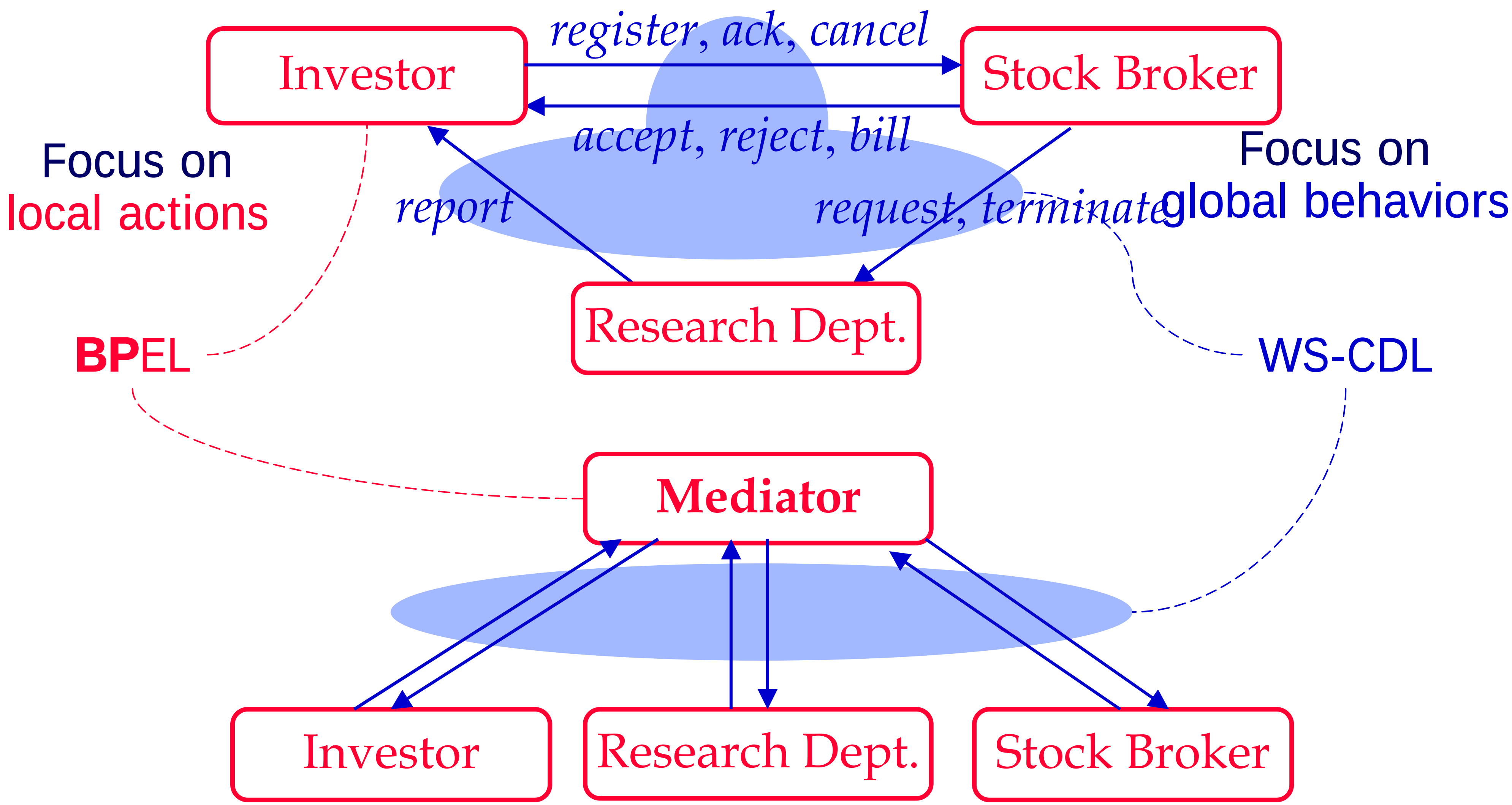
Orchestration vs Choreography



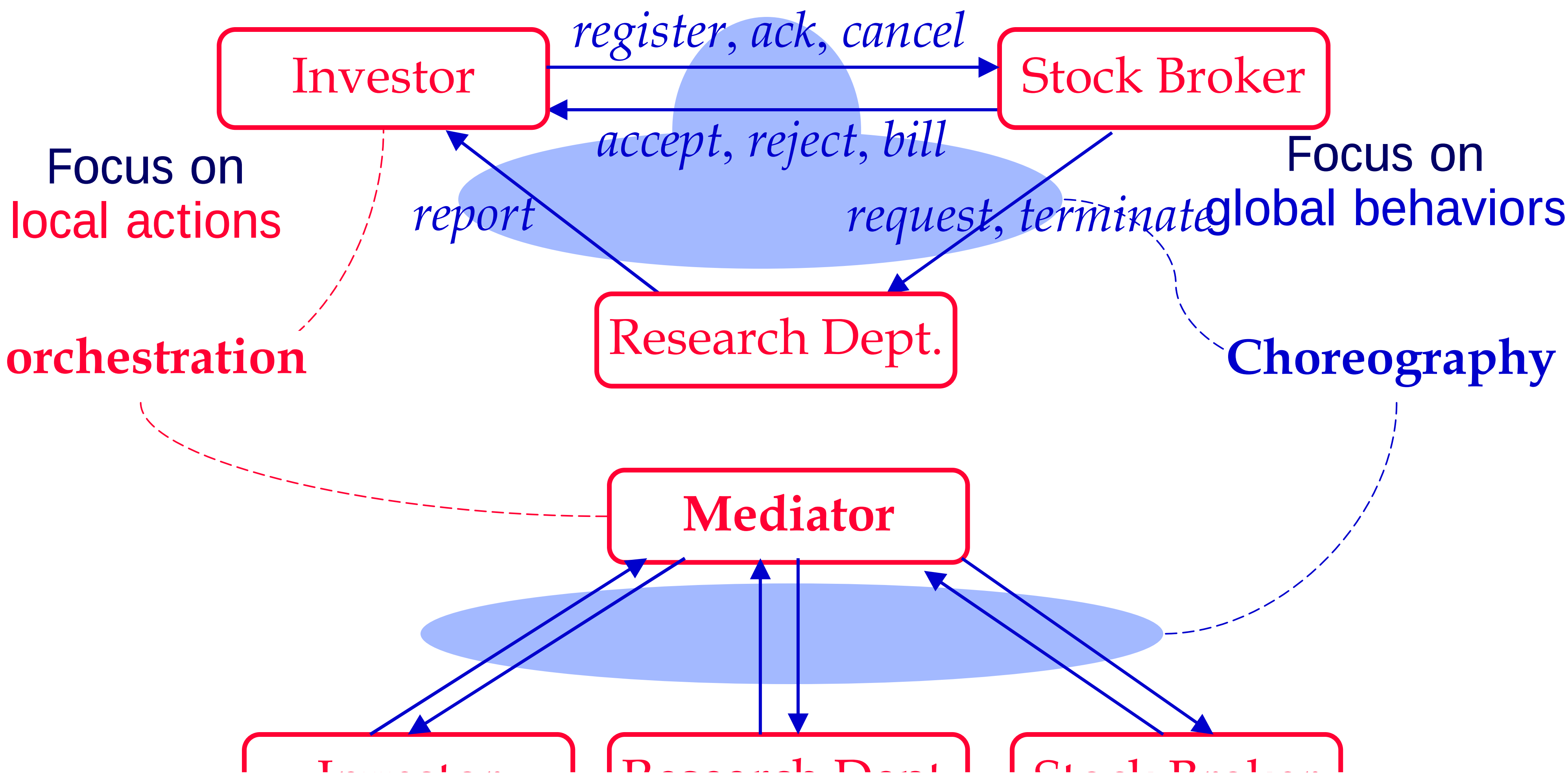
WS Choreography Definition Language

- Specification of choreography
- Model complex business protocol (e.g. order management) to enable interoperability
- Generate computational logic of individual collaborating participants
- Key concepts
  - Collaborating participants: role, relationship, participants
  - Information driven collaboration: channel, activities, workunit, choreography
- Standardization through W3C (Version 1.0: December 2004)

Composition: BPEL and WS-CDL

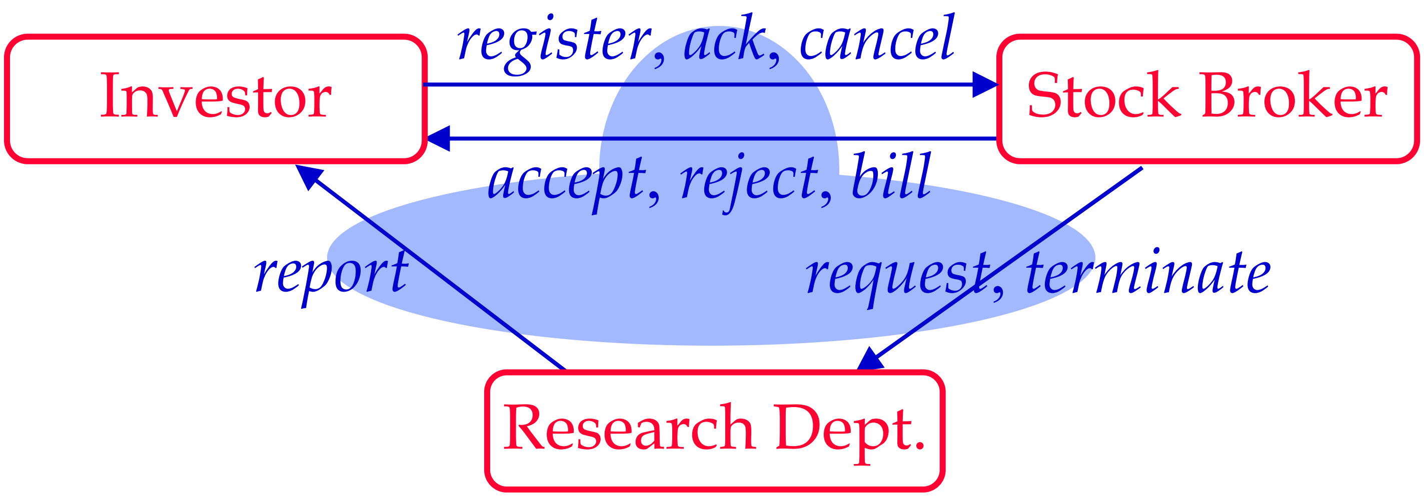


Composition: BPEL and WS-CDL



- For “hub and spoke”, the difference is small
  For “peer-to-peer”, the concept of choreography is interesting and not well understood

Automated Design: Top-down vs Bottom-up



Top-down

specification of **individual services** *orchestration*

Bottom-up

specification of **global behaviors** *Choreography*

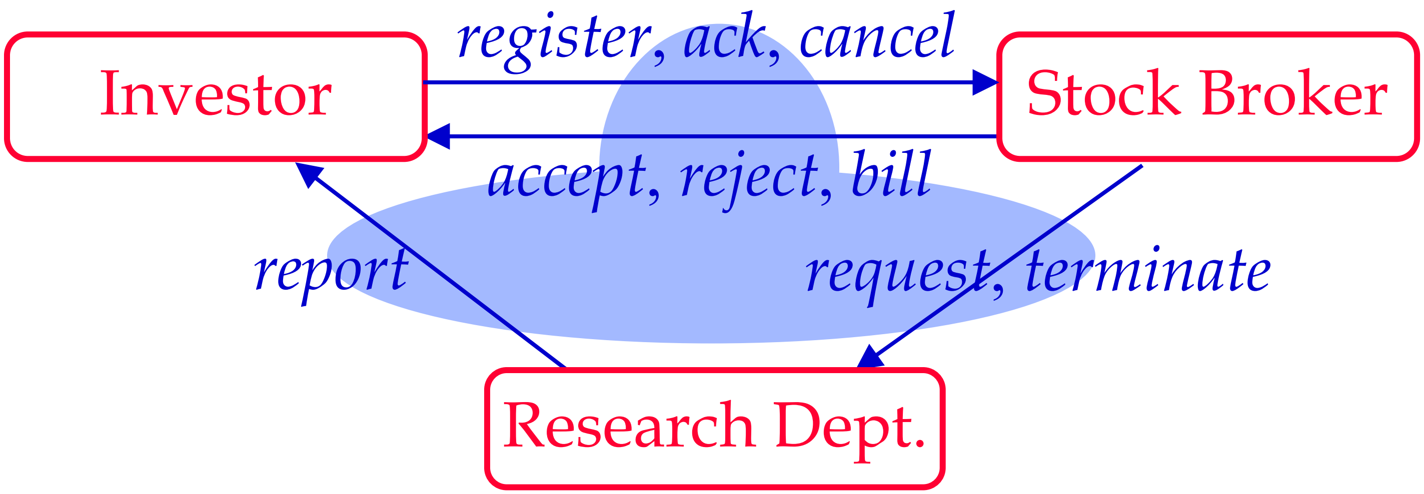
e.g., BPEL

e.g., WS-CDL

- Verification and analysis of choreography
  - Focus on the conversation model

Verification of WS Choreography

- Verification of choreography of a WS (BPEL) composition



- Services: finite transition systems on messaging actions
- Unbounded FIFO queues for messages
- Choreography: message sequences (send only)
  - How to model?
- LTL on choreography

[Fu-Bultan-S. WWW'04, ISSTA'04]

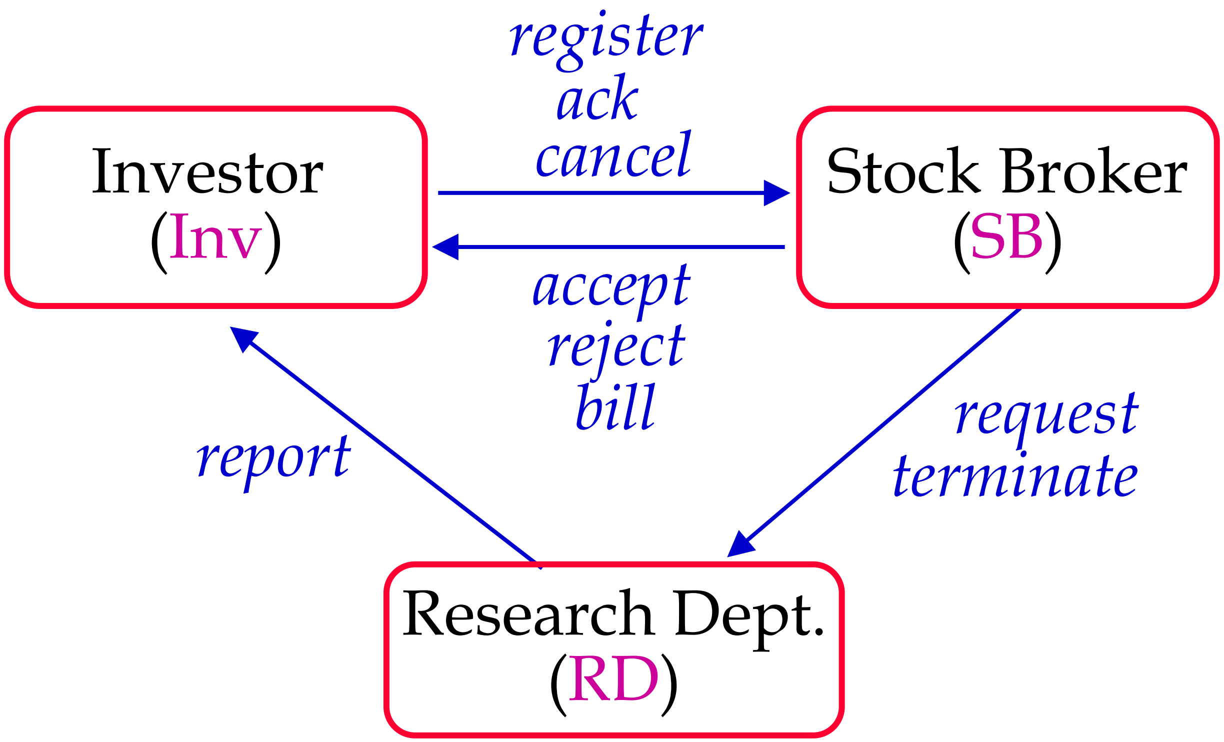
An Example: Stock Analysis Service (SAS)

Three peers: Investor, Stock Broker, and Research Dept

- Inv initiates the stock analysis service by sending a *register* message to SB
- SB may *accept* or *reject* the registration
- If the registration is accepted, SB sends an analysis *request* to the RD
- RD sends the results of the analysis directly to the Inv as a *report*
- After receiving a *report* Inv can either send an *ack* to SB or *cancel* the service
- Then, SB either sends the *bill* for the services to Inv, or continues the service with another analysis *request*

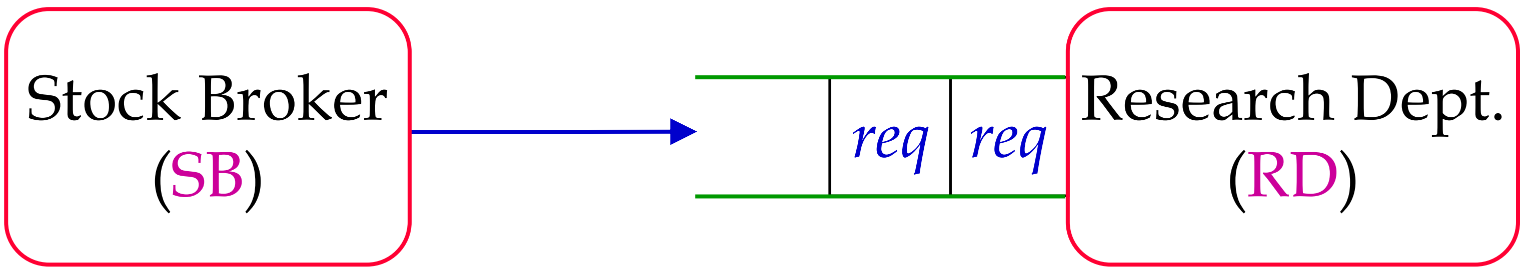
SAS Composition

- SAS is a web service composition
  - a finite set of peers: Inv, SB, RD, and
  - a finite set of message classes: *register*, *ack*, *cancel*, *accept*, ...



Asynchronous Messaging

- We assume that the messages among the peers are exchanged through reliable and asynchronous messaging
  - FIFO and unbounded message queues

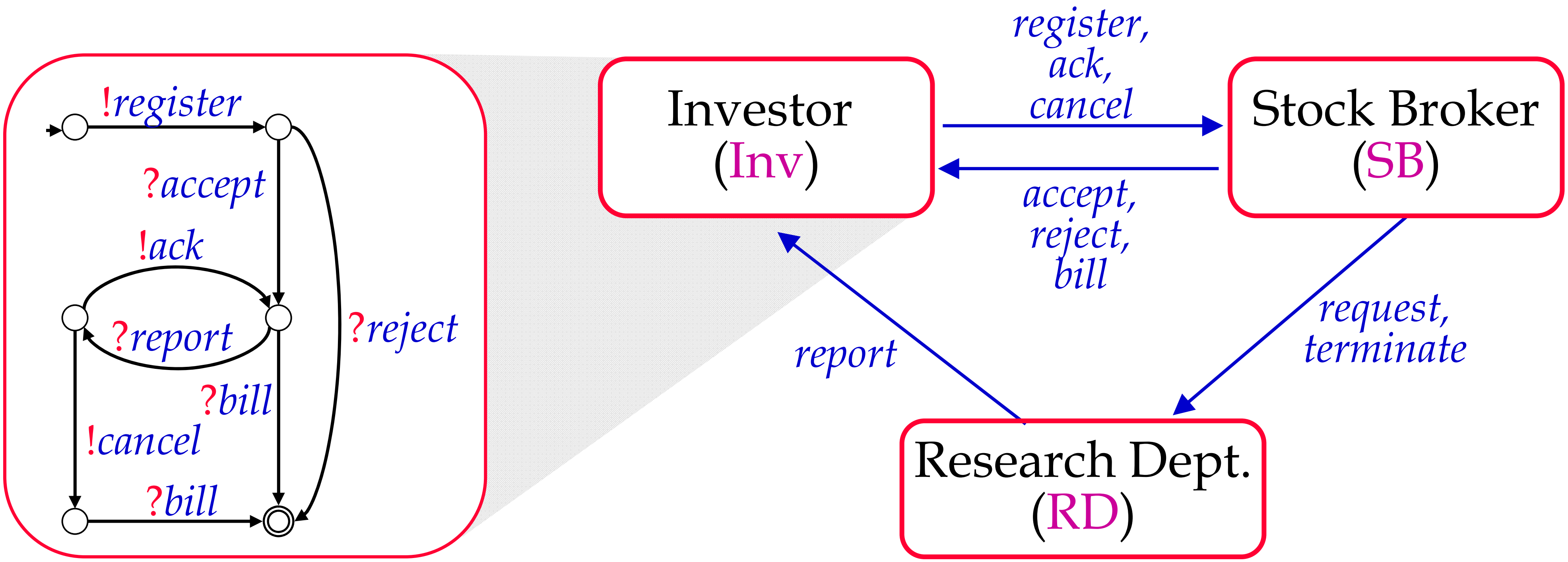


- This model is similar to industry efforts such as
  - JMS (Java Message Service)
  - MSMQ (Microsoft Message Queuing Service)

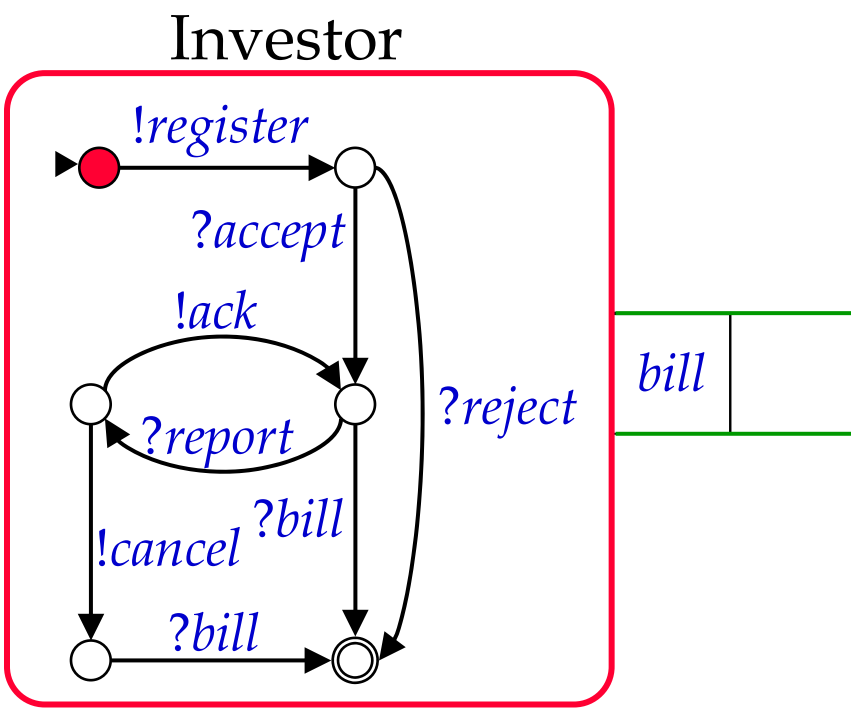
Mealy Service Model

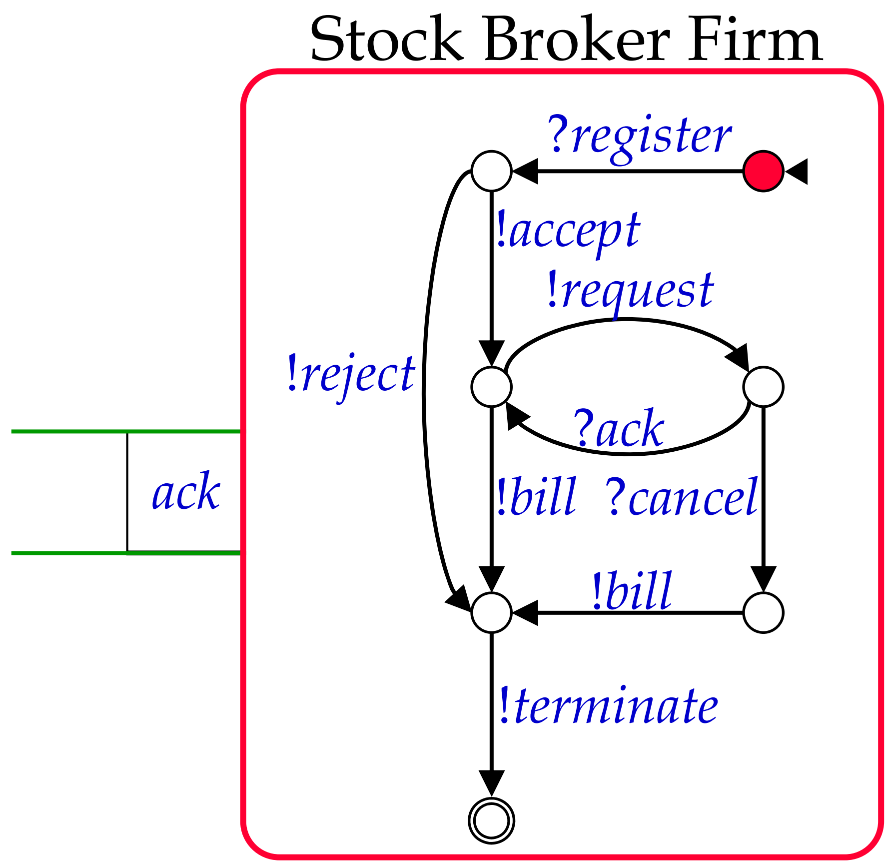
[Bultan-Fu-Hull-S. WWW'03]

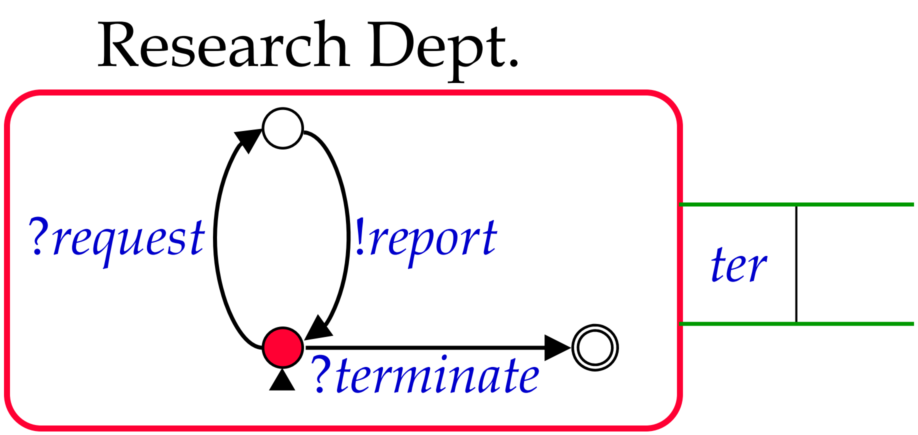
- Finite state control
- Acts on a finite set of message classes
- Transitions are based on receiving a message ?*m* or sending a message !*m*



Composite Mealy Service Execution



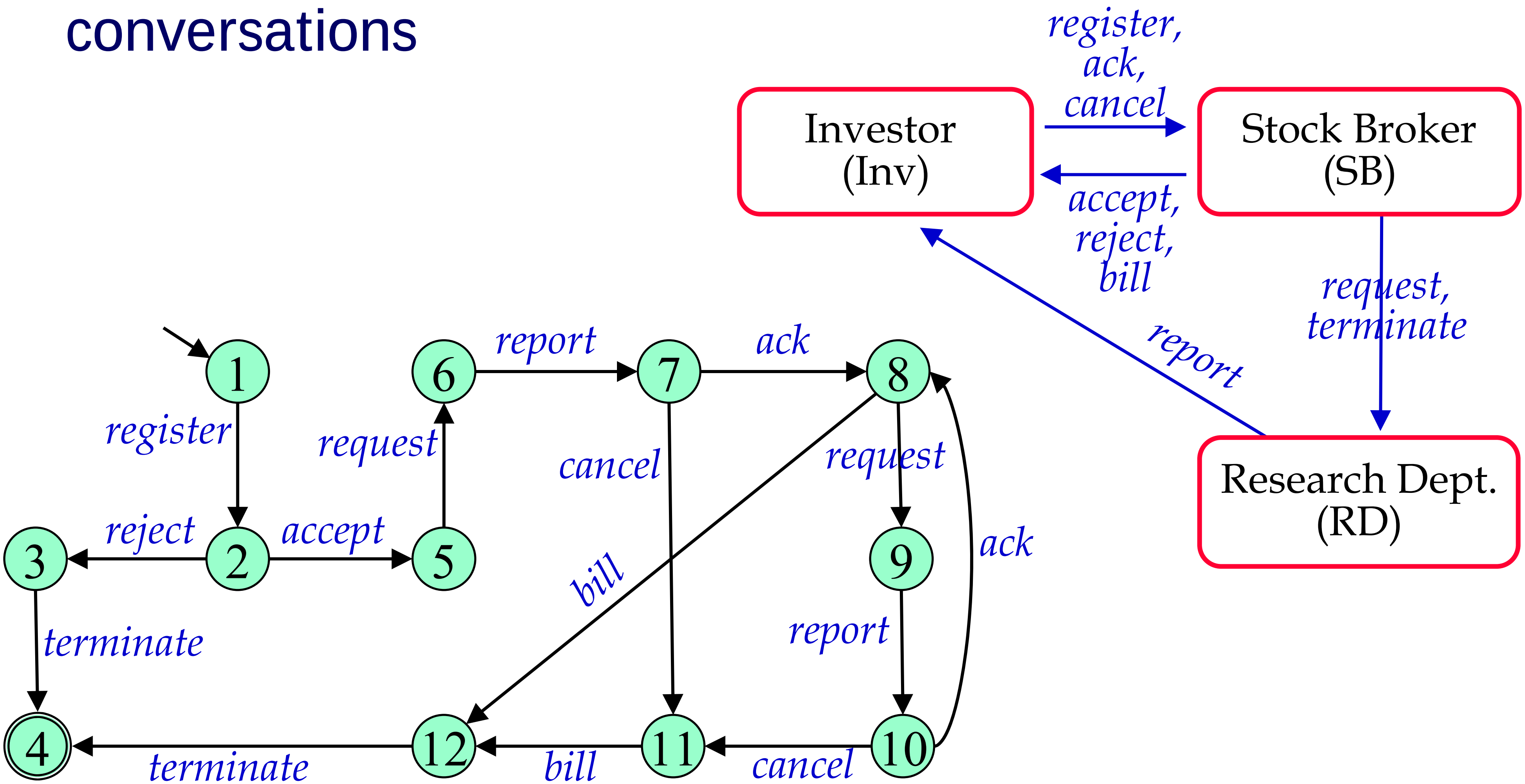




- Execution halts if
  - All Mealy services are in final states, and
  - All queues are empty

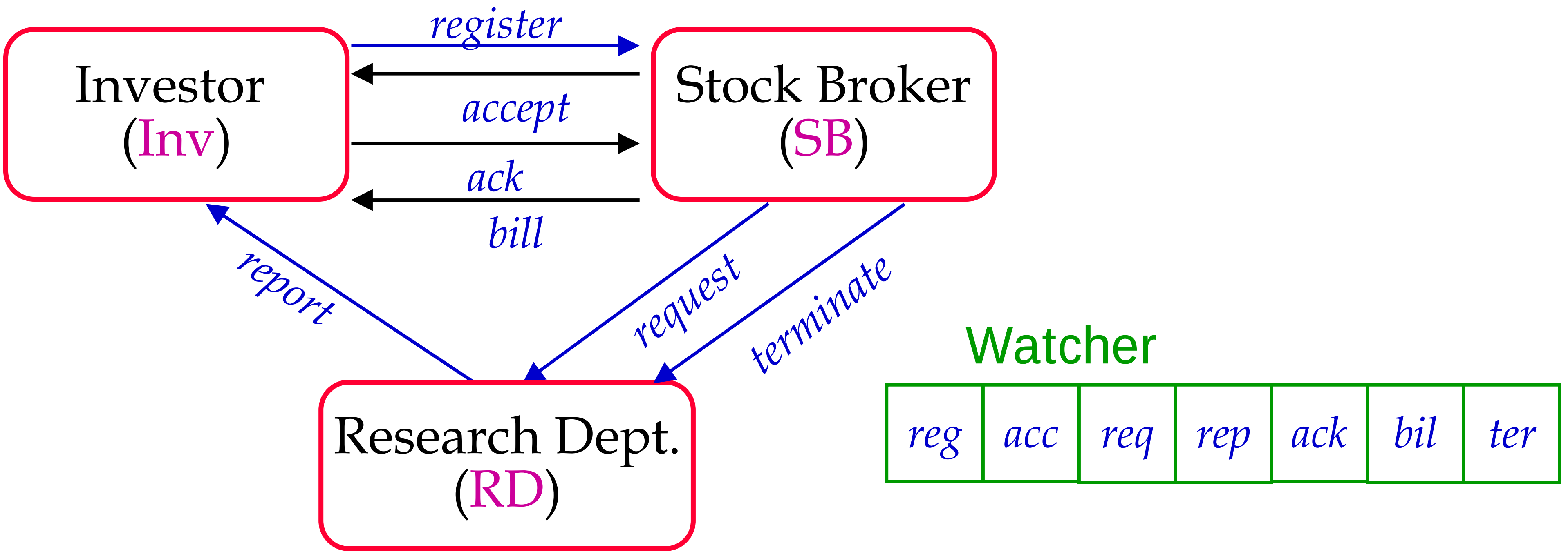
Conversations and Conversation Protocols

- Conversation: a message sequence
- A conversation protocol specifies the set of desired conversations



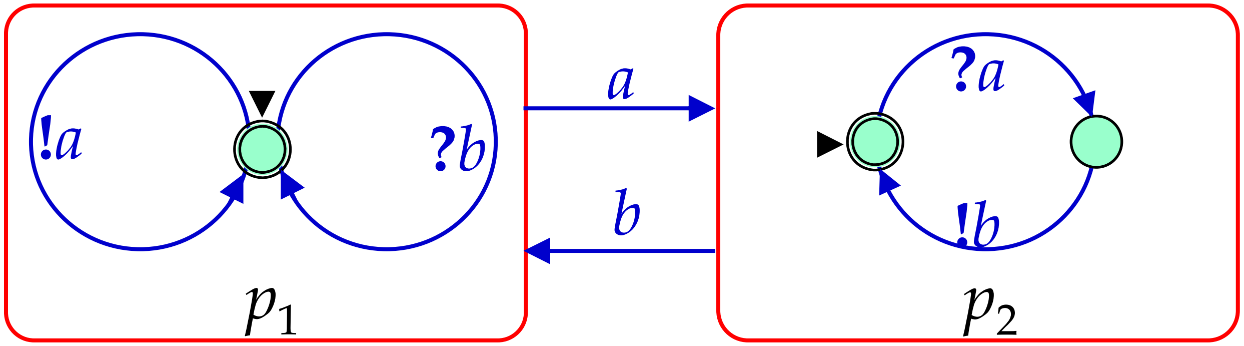
Conversations of Composite Services

- A virtual watcher records the messages as they are sent



- A conversation is a sequence of messages the watcher sees in a successful run (or enactment)
- Conversation language: the set of all possible conversations
- What properties do conversation languages have?

Conversation Languages Are Not Regular



- The set of conversations CL $\cap\, a^*b^* = a^n b^n$
- Conversation languages are not always regular
  - Some may not even be context free
- Causes: asynchronous communication & unbounded queue
- Bounded queues or synchronous: CL always regular
- CLs are always context sensitive

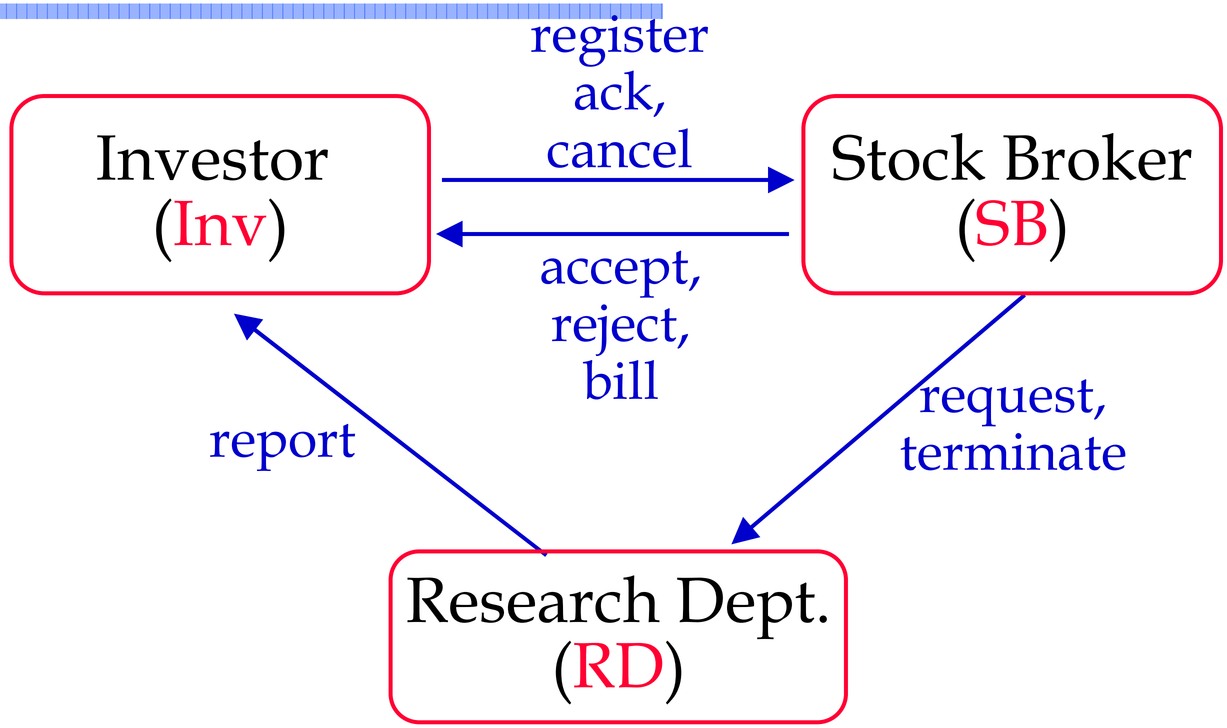
Remarks

- Communicating finite state machines with queues are computationally Turing complete
  - Conversation languages $\neq$ tracing execution states

- Why regular languages?
  - They would allow static analysis, e.g. model checking
    - Testing and debugging in SOA are harder

- Queue v.s. no queue: design time decision!

Two Key Questions



- Is the composition of (BPEL) services "correct"?
  - Verify conversations
- Automated design of services from the desired conversation protocol?

Temporal Properties of Conversations

- The notion of conversation enables reasoning about temporal properties of the composite web services
- Extend LTL extends naturally to conversations
  - LTL temporal operators

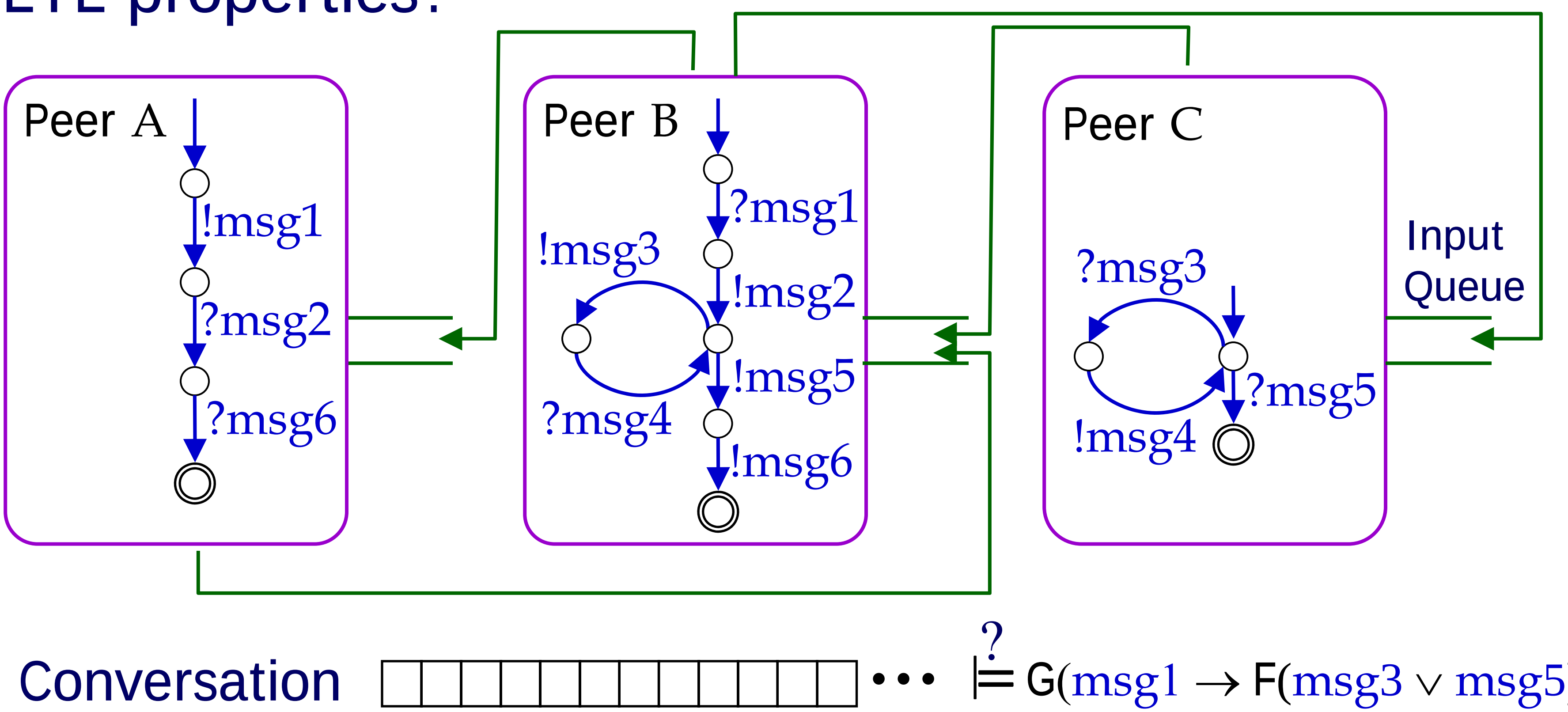
    X (neXt), U (Until), G (Globally), F (Future)
  - Atomic properties

    Predicates on message classes (or contents)
  - Example: $\mathsf{G}\,(accept \rightarrow \mathsf{F}\,bill)$

- Verification problem: Given an LTL property, does the conversation language (i.e. every conversation) satisfy the property?

Design Scenario 1: Bottom Up

- Given a composition of services, does its CL satisfy the LTL properties?



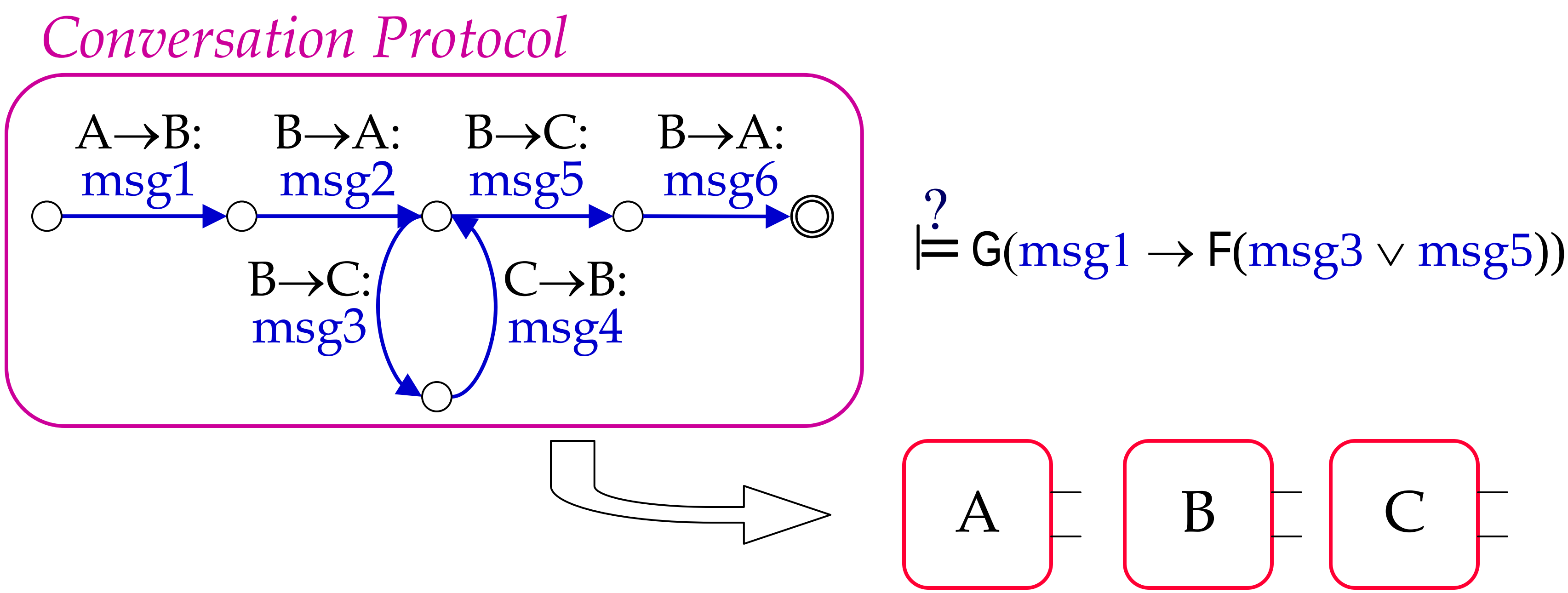
$\overset{?}{\models}$ G(msg1 $\rightarrow$ F(msg3 $\vee$ msg5))

- Problem: the general case is undecidable

[Brand-Zafiropulo JACM'83]

Design Scenario 2: Top Down

- Specify the global messaging behavior explicitly as a conversation protocol
- Determine if the conversations allowed by the protocol satisfy LTL properties



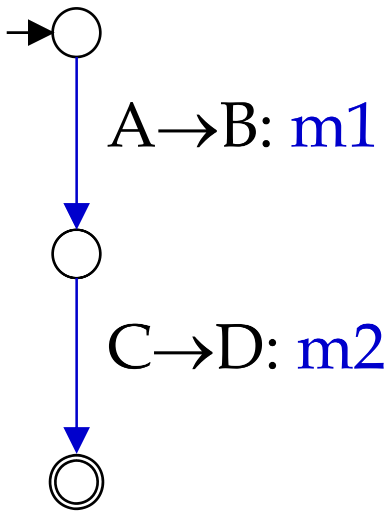
- Problem: the conversation protocol may not be realizable

Approaches

- (Bottom up) verification is undecidable
  - Approach 1: check if the conversations using bounded queue satisfy LTL property
    —partial verification
  - Approach 2: sufficient condition for bounded queue CL = unbounded queue CL
    —synchronizablility
- (Top down) protocol may be unrealizable
  - Approach 3: sufficient condition for realizability

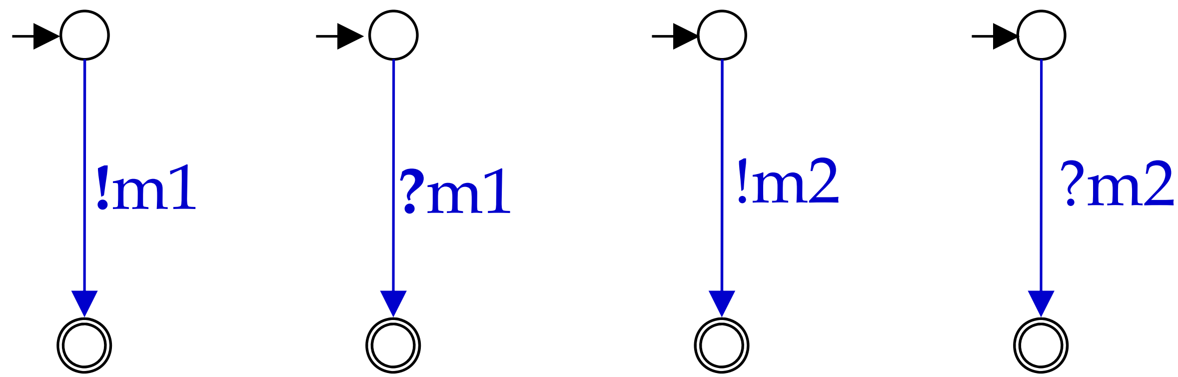
Realizability Problem

- Not all conversation protocols are realizable!



Conversation protocol

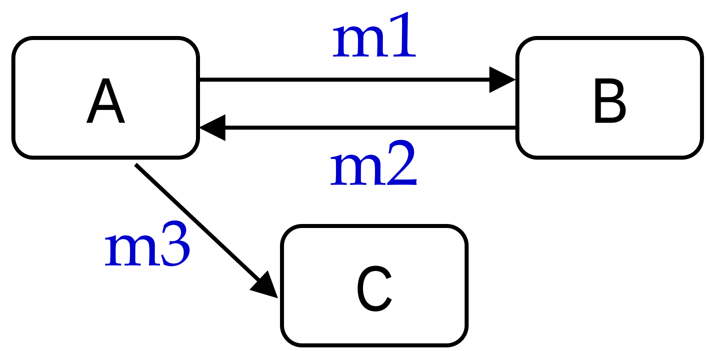
Projection of the conversation protocol to the peers

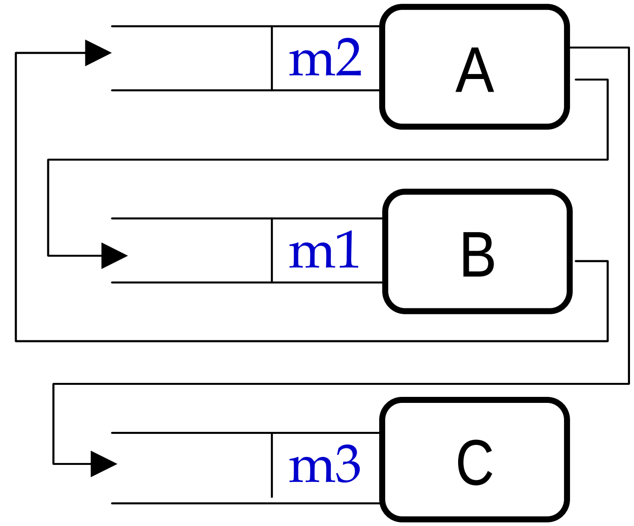


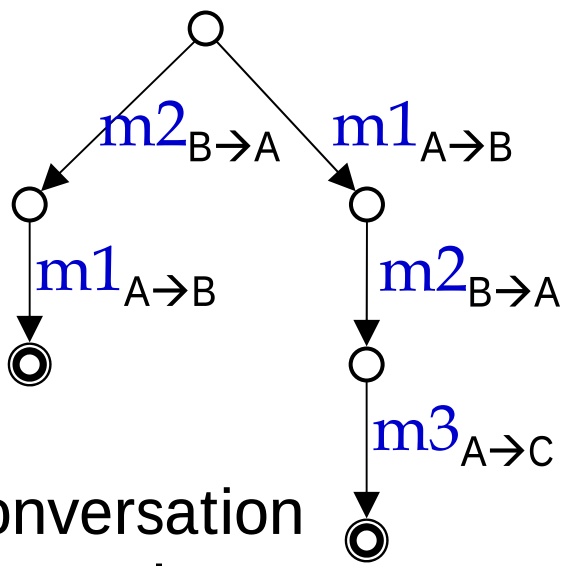
Peer A Peer B Peer C Peer D

- Conversation "m2 m1" will be generated by any legal peer implementation which follows the protocol

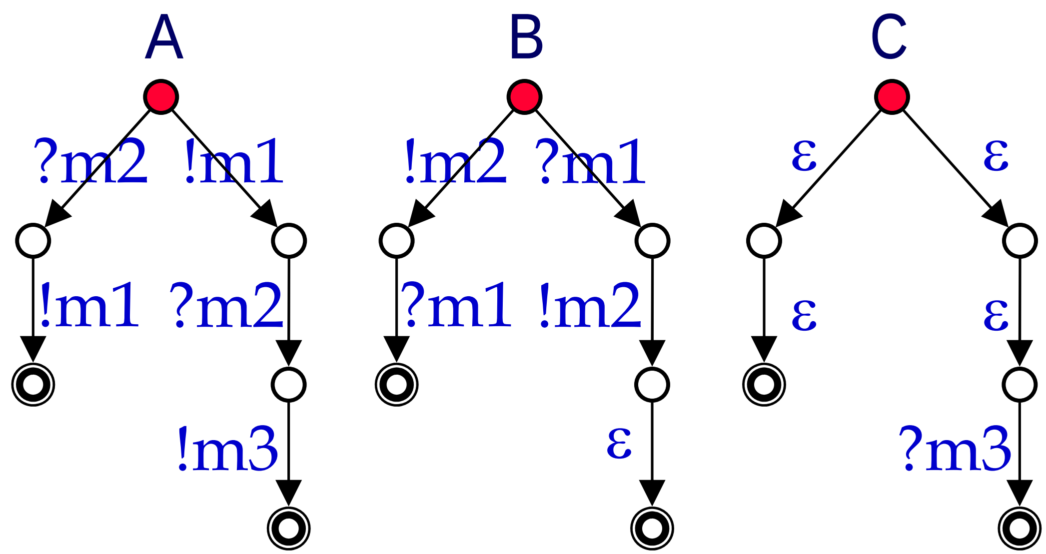
Another Non-Realizable Protocol







Conversation protocol



Generated conversation: | m2 | m1 | m3 |

A Sufficient Condition for Realizability

[Fu-Bultan-S. CIAA ’03]

- Three parts for realizability (contentless messages)
  - Lossless join
    Conversation protocol should be equal to the “join” of its projections to each peer
  - Synchronous compatible
    When the projections are composed synchronously, there should not be a state where a peer is ready to send a message while the corresponding receiver is not ready to receive
  - Autonomous
    Each peer should be able to make a deterministic decision on whether to send or to receive or to terminate

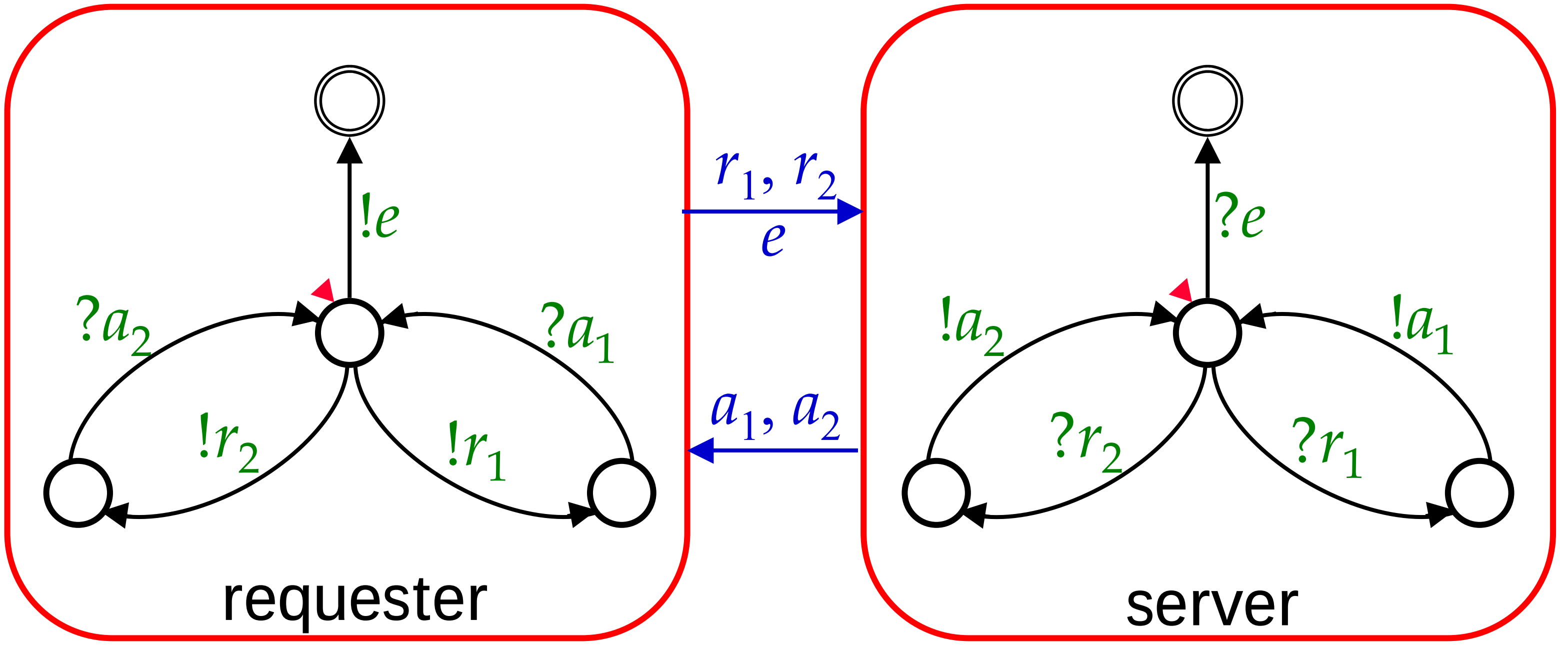
Bottom-Up Approach

- Given a composition of web services, check if its conversations satisfy some LTL properties

- General problem is undecidable due to asynchronous communication (with unbounded queues)

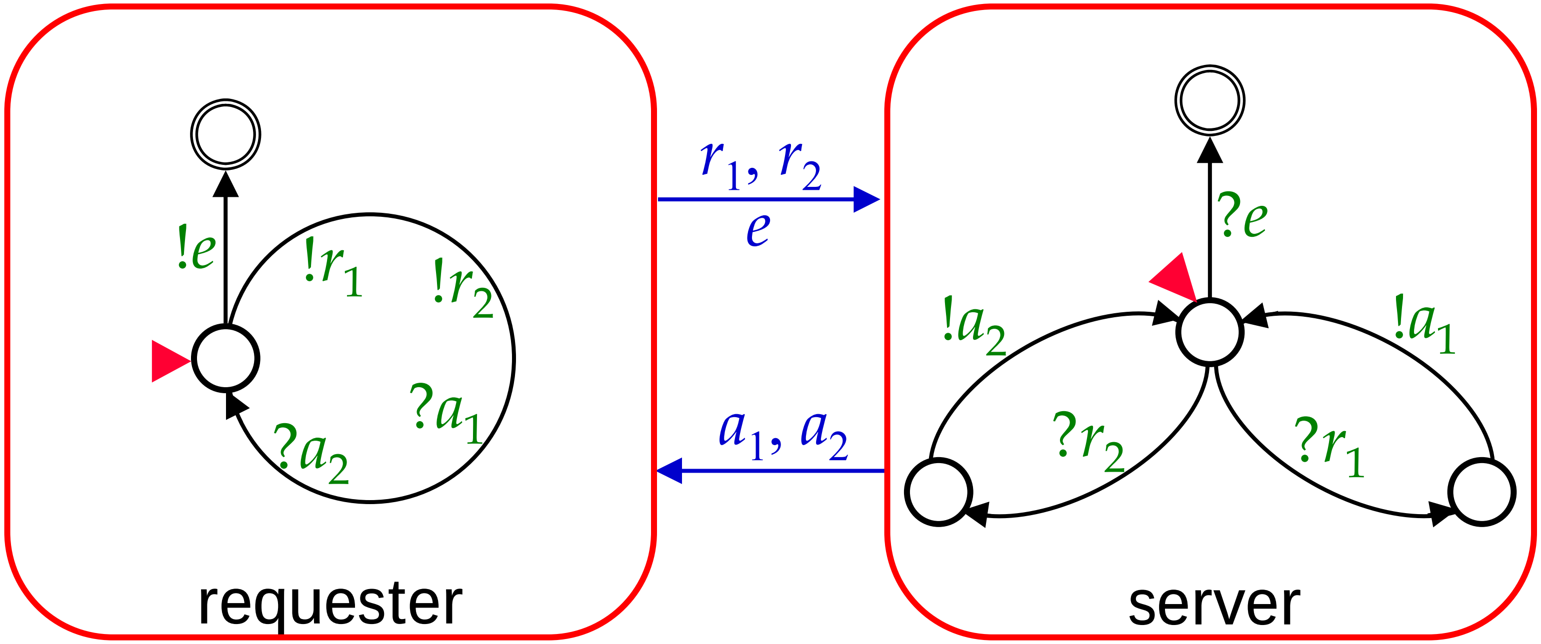
- Naïve idea: limit the queue length
  - Problem 1: only partial verification, unless we are lucky
  - Problem 2: state size explosion

Example 1: Regular CL, Bounded Queues



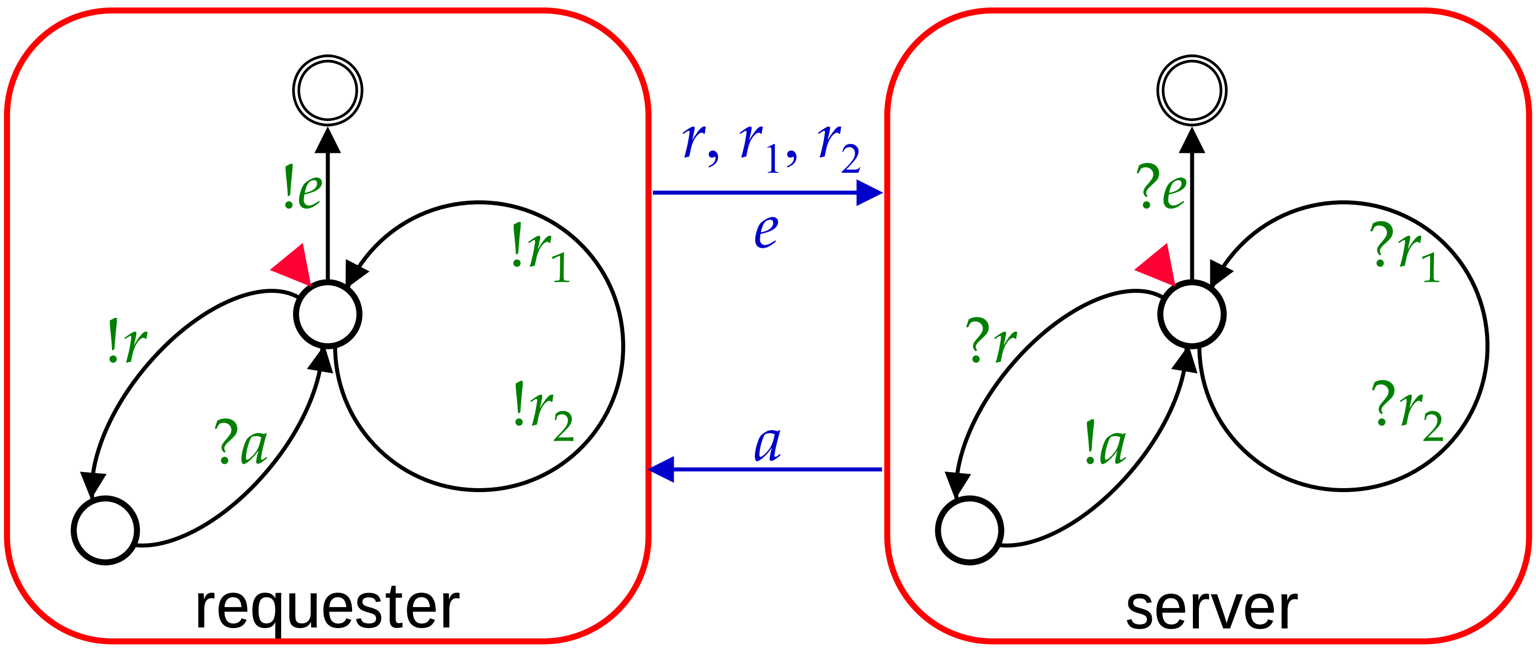
- Conversation language is regular: $(r_1 a_1 \mid r_2 a_2)^* \; e$
- During every halting run two queues are bounded

Example 2: Not Regular, Unbounded



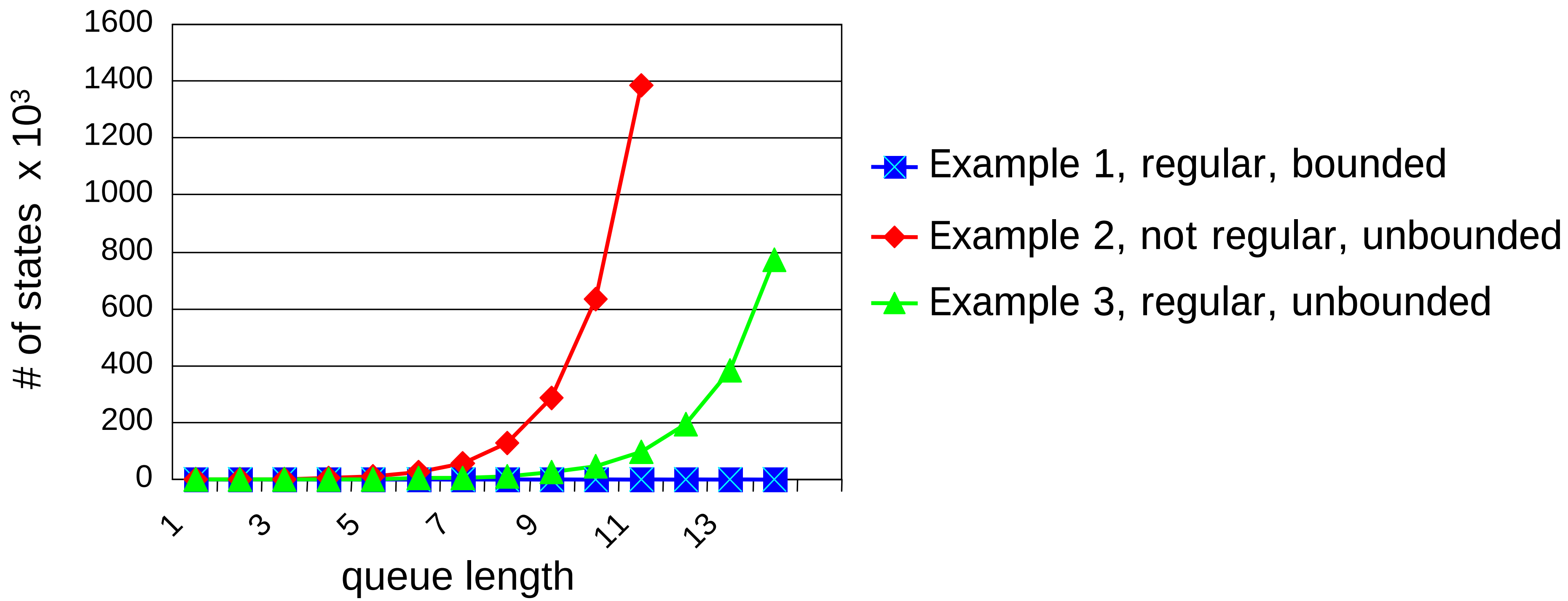
- Conversation language is not regular
- Queues are not bounded

Example 3: Regular, Unbounded



- Conversation language is regular: $(r_1 \mid r_2 \mid ra)^* \, e$
- Queues are not bounded

Three Examples



- Verification of Examples 2 and 3 are difficult even if we bound the queue length
- How can we distinguish Examples 1 and 3 (with regular conversation languages) from 2?
  - ➔ Synchronizability Analysis

Synchronizability Analysis

- A composite web service is synchronizable, if its conversation language does not change
  - when asynchronous communication with unbounded queues is replaced with synchronous communication or bounded queues

- A composite web service is synchronizable, if it satisfies the synchronous compatible and autonomous conditions [Fu-Bultan-S. WWW'04]

Are These Conditions Too Restrictive?

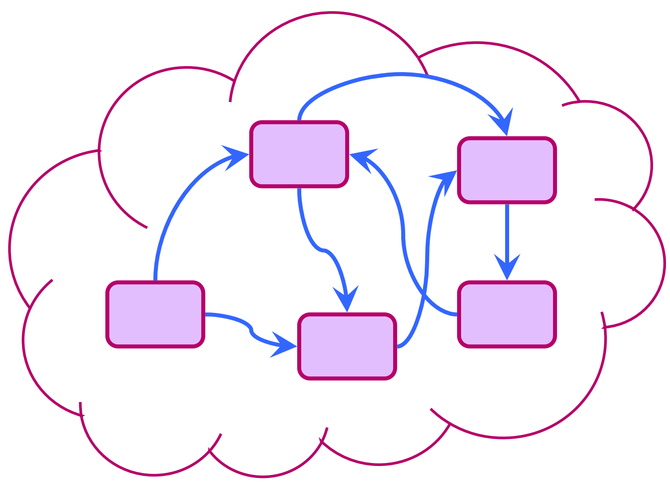
Problem Set		Size			Synchronizable?
Source	Name	#msg	#states	#trans.	
ISSTA'04	SAS	9	12	15	yes
IBM Conv. Support Project	CvSetup	4	4	4	yes
	MetaConv	4	4	6	no
	Chat	2	4	5	yes
	Buy	5	5	6	yes
	Haggle	8	5	8	no
	AMAB	8	10	15	yes
BPEL spec	shipping	2	3	3	yes
	Loan	6	6	6	yes
	Auction	9	9	10	yes
collaxa.com (Oracle)	StarLoan	6	7	7	yes
	Cauction	5	7	6	yes

Summary

- Verification of choreography is intricate
  - Choreography of composition may not be regular and does not fall into natural formal language classes
  - Must be concerned with the realizability problem
- Realizability and verification on conversations with Mealy machines [Fu-Bultan-S. 2003-6]
- Realizability on process algebras, choreography languages [many, 2005-]

Outline

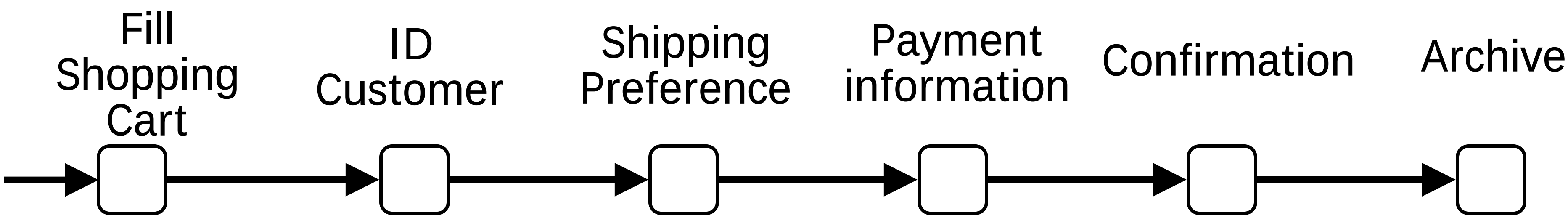
- Motivations
- Transitions systems
- BPEL services and compositions
- Choreographies (of BPEL services)
- Artifact-centric workflow
- Concluding remarks



$\overset{?}{\models} \quad \varphi$

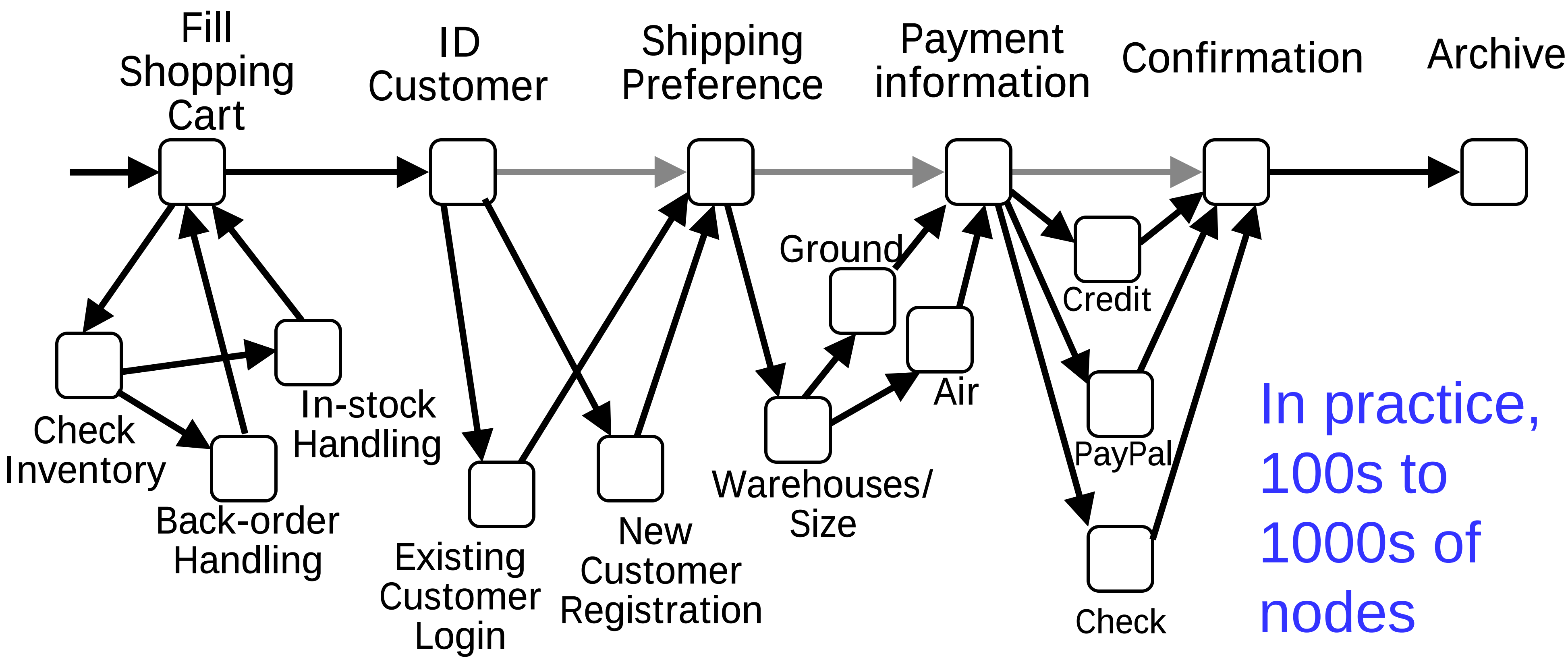
Workflow (Business Process)

- A bookseller example: Traditional control-centric model



Workflow (Business Process)

- A bookseller example: Traditional control oriented model
- Multiple steps needed for each activity

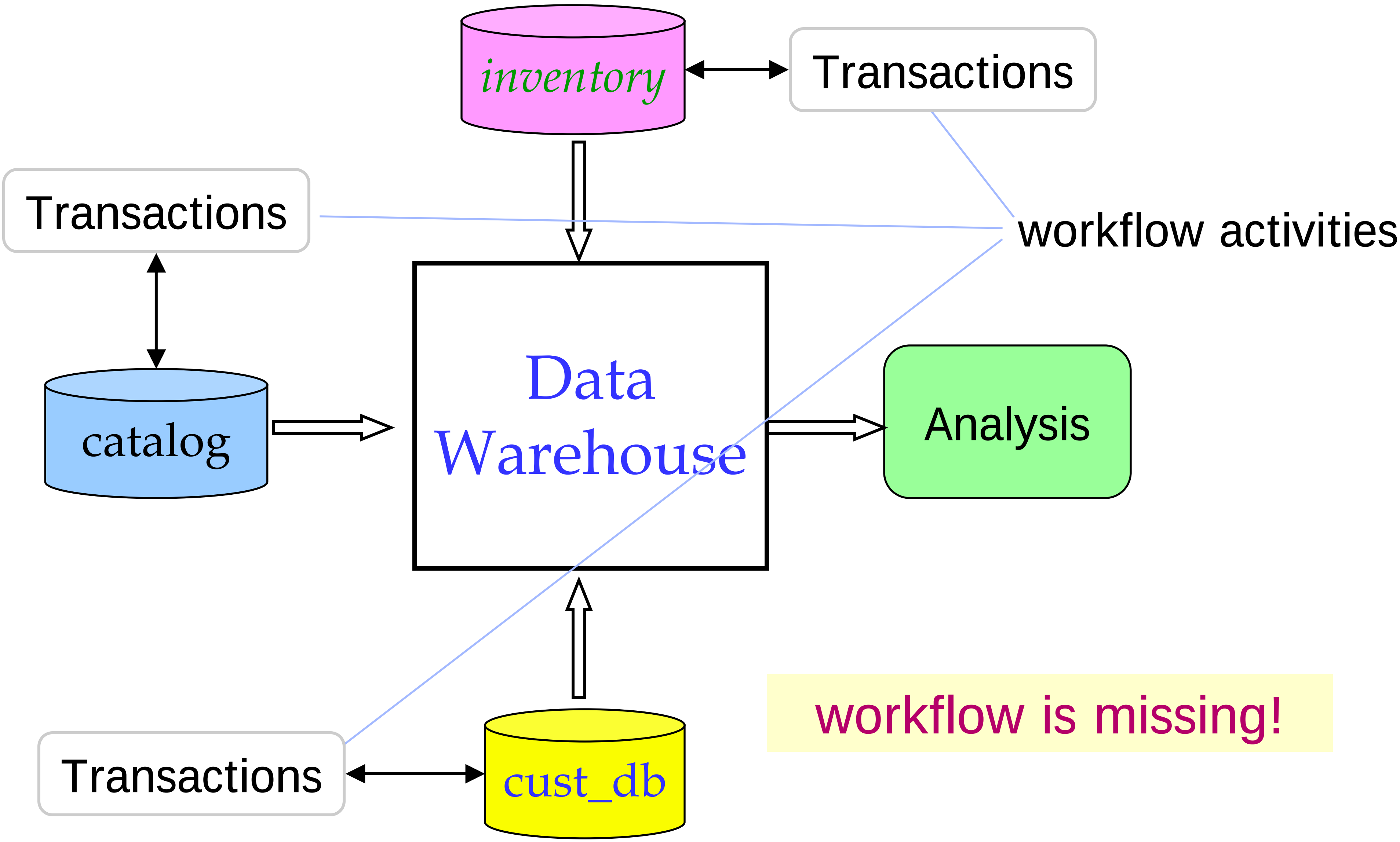


In practice, 100s to 1000s of nodes

Hard to reason, find useful views: missing data

Business Intelligence: Data View

- Extract-Transform-Load

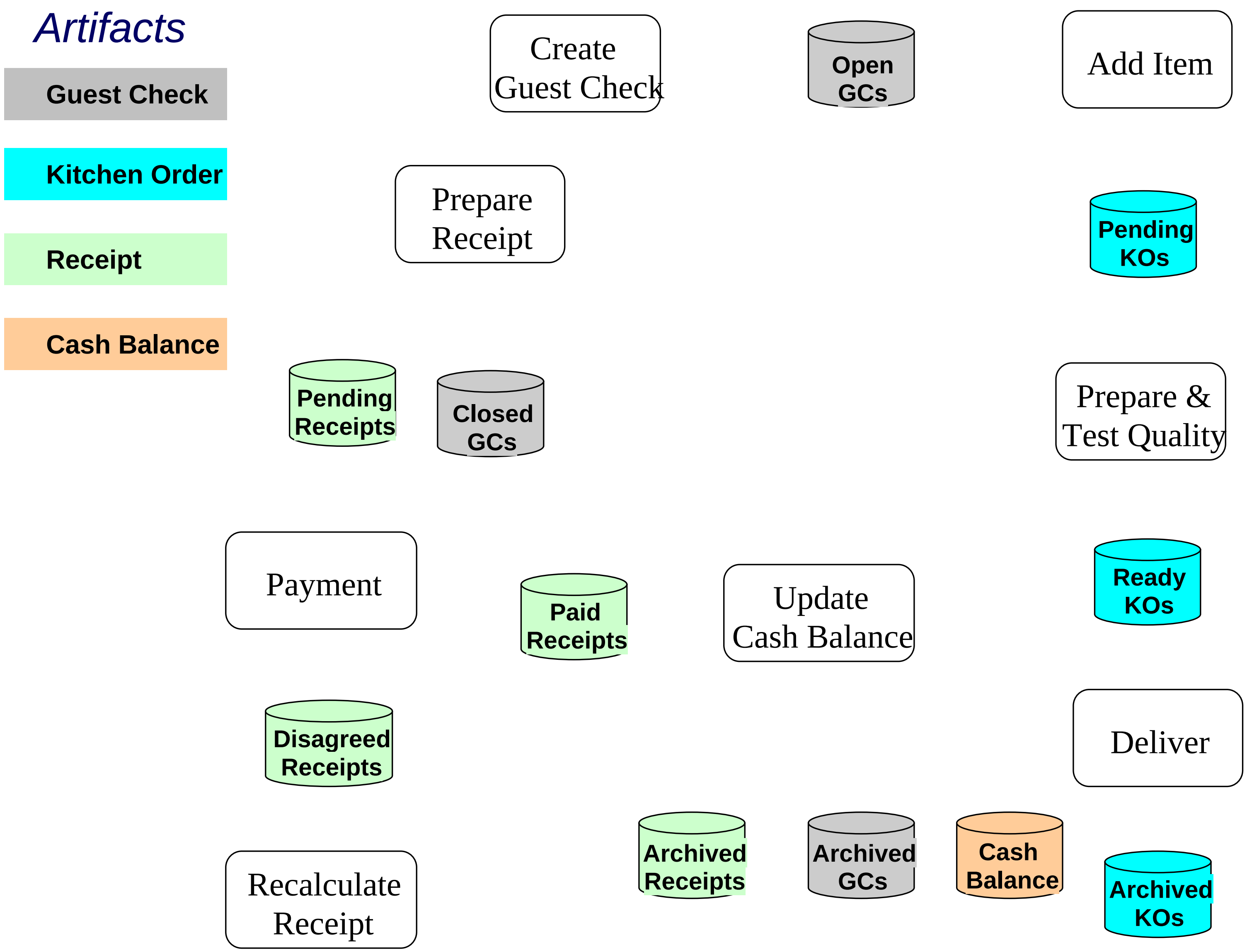


Business Artifacts !

- A business artifact is a key conceptual business entity that is used in guiding the operation of the business
  - *fedex package delivery*, *patient visit*, *application form*, *insurance claim*, *order*, *financial deal*, *registration*, ...
  - both “information carrier” and “road-maps”

- Very natural to business managers and BP modelers
- Includes two parts:
  - Information model:
    data needed to move through workflow
  - Lifecycle:
    possible ways to evolve

Example: Restaurant



Example: Restaurant

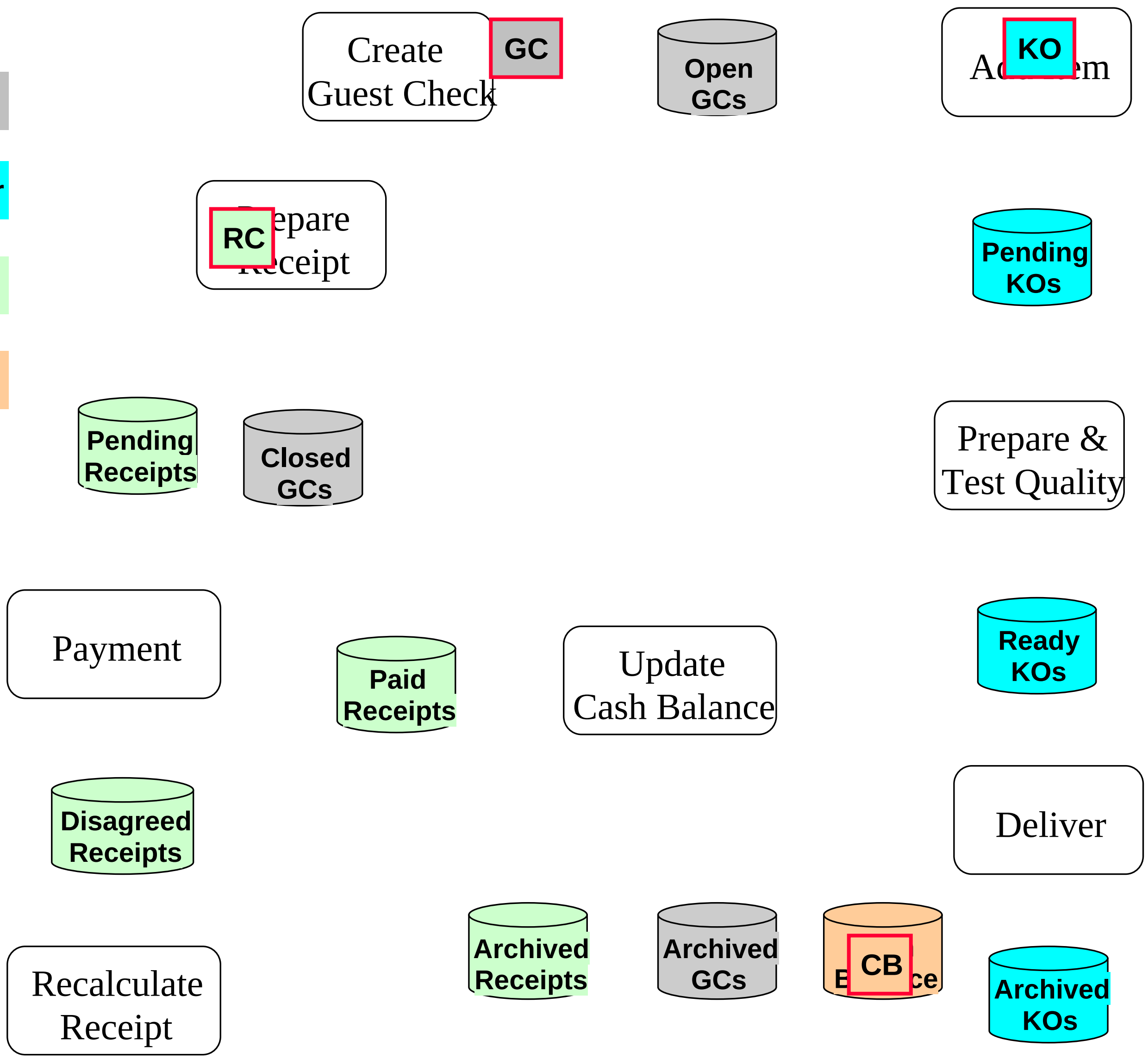
Artifacts

Guest Check

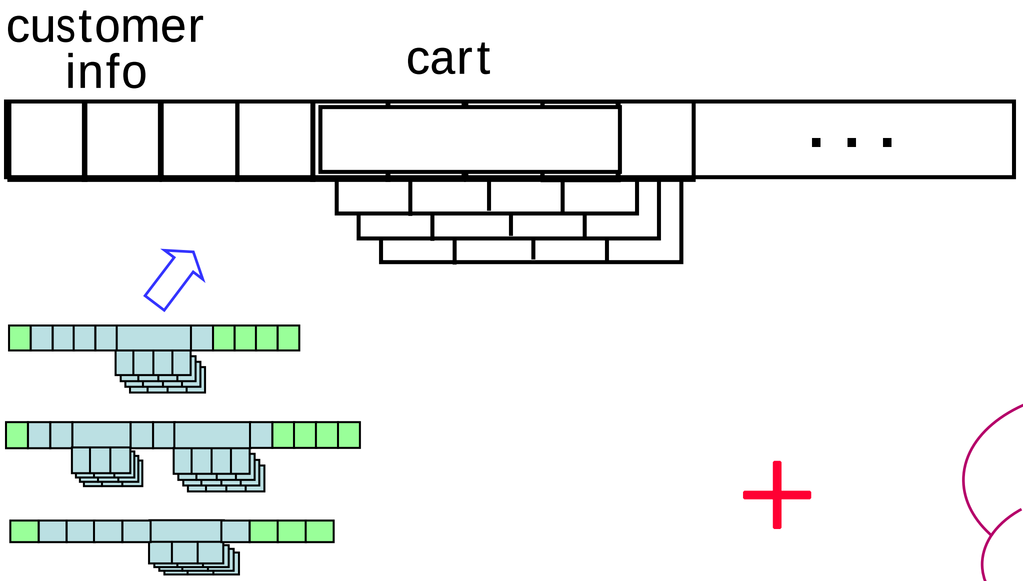
Kitchen Order

Receipt

Cash Balance



Emerging Artifact-Centric BPs

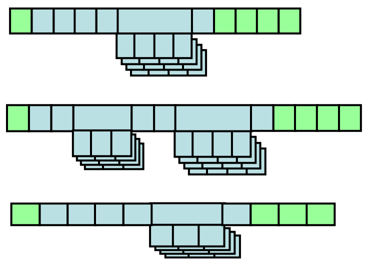


Artifacts (Info models)

+

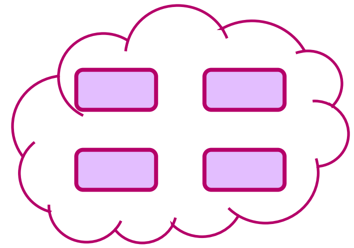
- Informal model [Nigam-Caswell IBM Sys J 03]
- Systems: BELA (IBM 2005), Siena (IBM 2007)
- Formal models
  - State machines [Bhattacharya-Gerede-S. SOCA 07] [Gerede-S. ICSOC 07]
  - Rules [Bhattacharya-Gerede-Hull-Liu-S. BPM 07]

A Logical Artifact Model for BPs



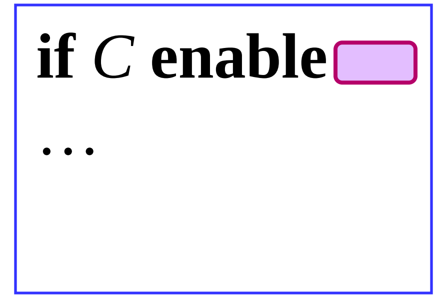
Artifacts
(info models)

+



Semantic services
(IOPEs)

+

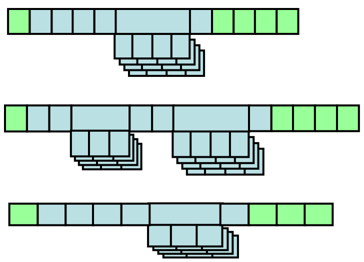


Condition-
action

- A variation of [Bhattacharya-Gerede-Hull-Liu-S. BPM 07]
- [Hull-S. 09] (in preparation)

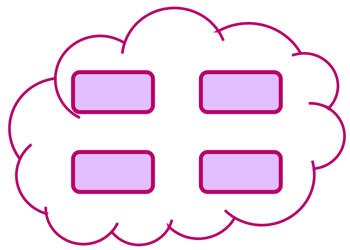
Verification Problem

- Given a workflow and a goal, do all executions of the workflow satisfy the goal?



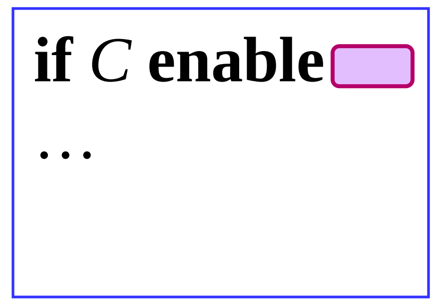
Artifacts
(Info models)

+



Semantic services
(IOPEs)

+



Condition-action

$\overset{?}{\models} \varphi$

[Bhattacharya-Gerede-S. SOCA 07] [Gerede-S. ICSOC 07]

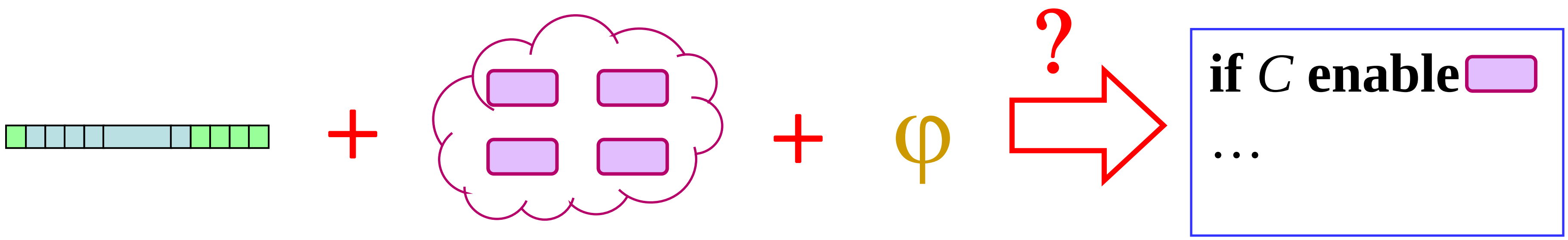
[Bhattacharya-Gerede-Hull-Liu-S. BPM 07]

[Deutsch-Hull-Patrizi-Vianu ICDT 09]

[Vianu ICDT 09]

Synthesis Problem

- Given a goal and a set of services, construct a set of rules so that every execution satisfies the goal



Artifact (Info model) | Semantic services (IOPEs) | Goal (FO) | Condition-action

[Fritz-Hull-S. ICDT 09]

(restricted to single artifact, first-order goals)

Workflow Schema

- A workflow schema is a triple

$$W = (\Gamma,\ \ S,\ \ R\ )$$

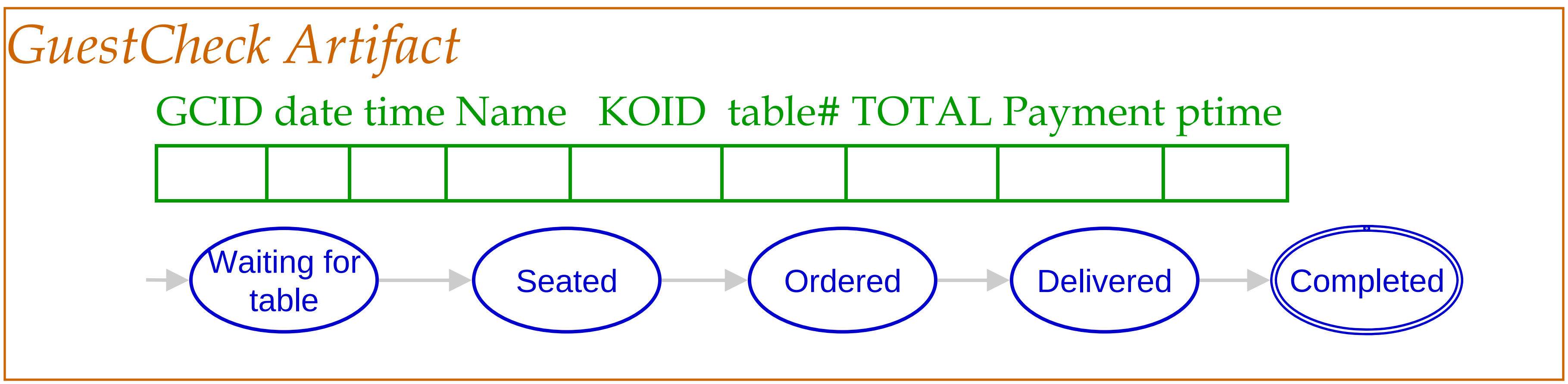
  - Γ : a set of artifacts classes (artifact schema)
  - S : a set of (semantic) services
  - R : a set of condition-action rules

A First-Order Logic + Structure

- Assuming some first order logic L with a fixed structure
  - U is the universe
- Existence of an infinite set of artifact IDs
- Existence of an infinite set of attributes

Artifact Classes

- An artifact class consists of
  - a finite set of attributes, of type U or artifacts IDs
  - a finite set of states, initial and final states (transitions not defined)
- An artifact is a pair:
  - a mapping from attributes to $U \cup \text{IDs} \cup \{\perp\}$
  - a state



Artifacts in a Workflow

- During runtime, each artifact class in Γ may have a finite set of artifacts
- The union I of sets of artifacts must be closed under "cross-referencing"

(Semantic) Services

- A service has a precondition and effects, conditions on
  - Attribute values
  - Defined-ness of attribute values
  - Equality of artifact IDs
  - An attribute holds the ID of a newly created artifact

SERVICE *SeatingGuests*

WRITE: $\{x\text{: } \mathit{GuestCheck}\}$

READ: $\{x\text{: } \mathit{GuestCheck},\ y\text{: } \mathit{Table}\}$

PRE-CONDITION: $\neg\text{Defined}(x.\mathit{table\#}) \wedge$
$\neg\text{Defined}(y.\mathit{GCID})$

EFFECTS:

- $\text{Defined}(x.\mathit{table\#}) \wedge \text{Seated}(x)$
- $\neg\text{Defined}(x.\mathit{table\#}) \wedge \text{Waiting4table}(x)$

Another Example

$$0 \le A \le 2 \quad \xrightarrow{A} \boxed{\sigma} \xrightarrow{B} \quad \begin{array}{l} 0 \le A < 1 \wedge 0 \le B \\ \vee \\ 1 \le A \le 2 \wedge 1 \le B \end{array}$$

(Semantic) Services

- A (semantic) service is a tuple $(\sigma, R, W, \pi, \rho)$, where
  - σ is a task name
  - R, W are finite sets of (resp., read, write) artifacts
  - π, ρ are quantifier-free formulas (pre- and post-condition, resp.) over attributes of artifacts in R, $R \cup W$, resp.
- allow $\mathrm{Defined}(A)$ for an attribute A

- I' is the result of executing σ on I, $I \xrightarrow{\sigma} I'$, if
  - $(I, I') \models \pi \wedge \rho$, and
  - frame conditions are satisfied

Condition-Action Rules

- Rules that define business logic
  - Invoke a service
  - Change artifact states

    states are used to organize the processing

if Waiting4Table(x) **enable** *SeaingGuest*(x)

if Defined($x.GCID$) $\wedge$ Defined($x.GCID.table$#)
change state to Taken(x) $\wedge$ Seated($x.GCID$)

Condition-Action Rules

- A condition-action rule is an expression of form "**if** φ **enable** σ" or "**if** φ **change state to** ϕ" or where
  - ❖ φ is a (quantifier-free) formula
  - ❖ σ is a semantic service
  - ❖ ϕ is a state changing formula

- I' is the result of executing a rule r : **if** φ **…** on I, $I \xrightarrow{r} I'$, if
  - ❖ $I \models \varphi$, and
  - ❖ $I \xrightarrow{\sigma} I'$ or I, I' only differ on states as specified

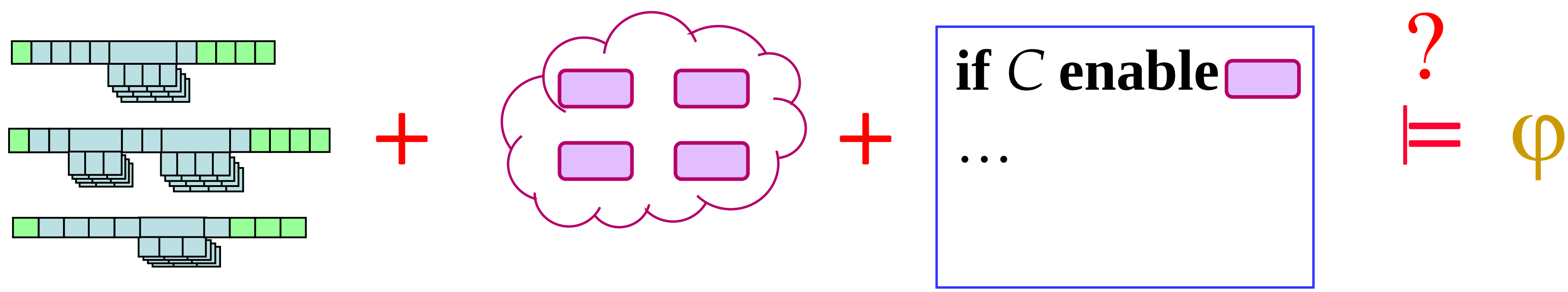
Workflow Schema

- A workflow schema is a triple $W = (\Gamma, S, R)$
  - Γ : artifact schema
  - S : a finite set of semantic services
  - R : a finite set of condition-action rules

- Denote $\xrightarrow{*}$ the closure of $\bigcup_{r \in R} \xrightarrow{r}$

Verification Problem

- Given a workflow and a goal, do all executions of the workflow satisfy the goal?



Artifacts
(Info models)

Semantic services
(IOPEs)

Condition-
action

[Bhattacharya-Gerede-Hull-Liu-S. BPM 07]

[Deutsch-Hull-Patrizi-Vianu ICDT 09]

Analysis Problems

- An artifact system $W = (\Gamma, \; S, \; R)$

  artifacts, services, rules

- Completion:

  Does W allow a complete run of some artifact?

- Dead-end:

  Does W have a dead-end path?

- Attribute redundancy:

  Does W have a redundant attribute?

No attribute value comparisons

[Bhattacharya-Gerede-Hull-Liu-S. BPM 07]

Results

- The problems are undecidable

  Primary reason: workflow language is Turing complete

- If we disallow creation of new artifacts
  - Initial: if each artifact has only initial attributes defined

  The analysis problems are PSPACE-complete
  - even for a single artifact

[Bhattacharya-Gerede-Hull-Liu-S. BPM 07]

- Consider only a single artifact

Monotonic Workflow

- Once an attribute is assigned a value, it cannot be changed

- For monotonic services:

  Complexity ranging from linear to intractable under various conditions

[Bhattacharya-Gerede-Hull-Liu-S. BPM 07]

Completion (Monotonic Workflow)

- Linear time if
  - Services are deterministic (single effect)
  - Preconditions has no negation
  - Rule conditions are positive and does not check state information
- NP-complete if the above conditions are slightly relaxed

(single artifact)

[Bhattacharya-Gerede-Hull-Liu-S. BPM 07]

Dead-End & Redundancy (Monotonic Workflow)

- Checking if there is a dead end path is Π_p^2-complete, even with various restrictions
- Checking redundant attributes is co-NP-complete, even with various restrictions

(single artifact)

[Bhattacharya-Gerede-Hull-Liu-S. BPM 07]

Three Analysis Problems: Review

- An artifact system $W = (\Gamma, S, R)$

  artifacts, services, rules
- Completion: Does W allow a complete run of an artifact?
- Dead-end: Does W have a dead-end path?
- Attribute redundancy: Does W have a redundant attribute?
- Undecidable in general, PSPACE if no artifact creation, intractable for monotonic workflows

  [Bhattacharya-Gerede-Hull-Liu-S. BPM 07]
- Ad hoc properties, restricted to defined-ness
- How to verify LTL properties?

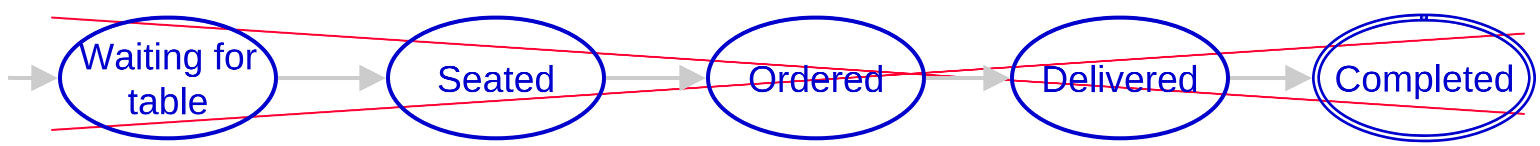
  [Deutsch-Hull-Patrizi-Vianu ICDT 09]

Adding Infinite States to Artifacts

- An artifact is a pair:
  - a mapping from attributes to $U \cup$ IDs $\cup \{\perp\}$
  - a state relation

GuestCheck Artifact

GCID	date	time	Name	KOID	table#	TOTAL	Payment	ptime



Items

ItemNo	Qty	cookingReq	Table#

Services Can Update State Relations

- Model operations on artifacts
  - updates of the artifact attributes
  - insertions/deletions in artifact states
- Insertions & updates can draw values from ...
  - current artifacts, state relations
  - external inputs (by programs or humans), computation that returns new values

Service Specification

Consists of

- pre-condition: a Boolean query on current snapshot of artifact system
- post-condition : constraints on the updated artifacts
- for each state relation, state insertion/deletion rules
  - specify tuples to add to (remove from) state relations
  - Defined as queries (over current snapshot)

queries, constraints: FO logic formulas

LTL(FO) to Express Properties

- LTL with propositions replaced by FO formulas (statements on individual snapshots)
- Classic LTL temporal operators

X p — p holds in next snapshot

p U q — p is true in every snapshot until q is

F p — p is eventually true

G p — p is always true

- Example (with slight abuse of notation) :

$$\mathsf{G}\,\neg(\neg \mathrm{Defined}(\mathit{table\#}) \wedge \exists \mathbf{z}\, \mathit{Items}(\mathbf{z}))$$

- The domain is dense order without endpoints

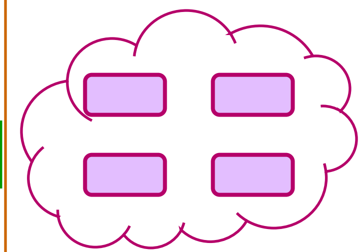
Verification Problem

GuestCheck Artifact

GCID	date	time	Name	KOID	table#	TOTAL	Payment	ptime

Items

ItemNo	Qty	cookingReq	Table#



Services $\overset{?}{\models}$ φ LTL(FO)

- In general, it is undecidable [Deutsch-Hull-Patrizi-Vianu ICDT 09]
- Need restrictions to turn it into decidable

Guarded FO

[Deutsch-Hull-Patrizi-Vianu ICDT 09]

- Guarded FO formulas restrict quantifications:

$$\exists x\, \varphi(x) \;\Rightarrow\; \exists x\, (A(\ldots,x,\ldots) \wedge \varphi(x))$$
$$\forall x\, \varphi(x) \;\Rightarrow\; \forall x\, (A(\ldots,x,\ldots) \rightarrow \varphi(x))$$

  $A(\ldots,x,\ldots)$: x is an attribute value and x cannot appear in any state atoms in φ

- All formulas used to update states are guarded FO
- Guarded LTL(FO): only allow guarded FO formulas

- Originated from input boundedness of [Spielmann 2003]

Guardedness is a Serious Limitation

- Not guarded:

$$\mathsf{G}\,\neg(\neg\mathrm{Defined}(\mathit{table\#}) \wedge \exists \mathbf{z}\; \mathit{Items}(\mathbf{z}))$$

- Guarded:

$$\mathsf{G}\,\neg(\neg\mathrm{Defined}(\mathit{table\#}) \wedge \mathit{Items}(\mathit{fish}, 1, x, 12))$$

Decidability Result

[Deutsch-Hull-Patrizi-Vianu ICDT 09]

- It can be decided in PSPACE if a guarded artifact schema satisfies a (guarded) LTL(FO)
- Actually complete in PSPACE

Summary

- Biz workflow a very promising application area for WS—tremendous impact (potentially)
- Analysis is hard but could be helped with modeling choices
- Artifact-centric workflow models: right intuition and positive experiences in practice (IBM)
- "Report on 2009 NSF Workshop on Data Centric Workflows" dcw2009.cs.ucsb.edu
  - More than 20 contributors, experts from CS, MIS, digital government, healthcare, scientific workflow

Concluding Remarks

- WS analysis and verification is important & interesting
  - Modeling
  - Design
- Current results: a good starting point
- SOA themes are yet to emerge, many open issues related to analysis
- Dynamic analysis

Acknowledgements

- Collaborators:
  - Tevfik Bultan (U C Santa Barbara)
  - Xiang Fu (Hofstra University)
  - Richard Hull, Kamal Bhatacharya, Rong Liu (IBM TJ Watson)
- Others:
  - Victor Vianu, Alin Deutsch (UCSD)
  - Fabio Patrizi (U of Rome)

References

K. Bhattacharya, C. Gerede, R. Hull, R. Liu, and J. Su: *Towards Formal Analysis of Artifact-Centric Business Process Models*. BPM 2007: 288-304

D. Brand and P. Zafiropulo: *On Communicating Finite-State Machines*. J. ACM 30(2): 323-342 (1983)

T. Bultan, X. Fu, R. Hull, and J. Su: *Conversation specification: a new approach to design and analysis of e-service composition*. WWW 2003: 403-410

A. Deutsch, R. Hull, F. Patrizi, and V. Vianu: *Automatic verification of data-centric business processes*. ICDT 2009: 252-267

A. Deutsch, L. Sui, and V. Vianu: *Specification and Verification of Data-driven Web Services*. PODS 2004: 71-82

D. Fahland and W. Reisig: *ASM-based Semantics for BPEL: The Negative Control Flow*. Abstract State Machines 2005: 131-152

R. Farahbod, U. Glässer, and M. Vajihollahi: *Specification and Validation of the Business Process Execution Language for Web Services*. Abstract State Machines 2004: 78-94

A. Ferrara: *Web services: a process algebra approach*. ICSOC 2004: 242-251

H. Foster, S. Uchitel, J. Magee, J. Kramer: *Model-based Verification of Web Service Compositions*. ASE 2003: 152-163

C. Fritz, R. Hull, and J. Su: *Automatic construction of simple artifact-based business processes*. ICDT 2009: 225-238

X. Fu, T. Bultan, and J. Su: *Conversation Protocols: A Formalism for Specification and Verification of Reactive Electronic Services*. CIAA 2003: 188-200

X. Fu, T. Bultan, and J. Su: *Analysis of interacting BPEL web services*. WWW 2004: 621-630

X. Fu, T. Bultan, and J. Su: *Model checking XML manipulating software*. ISSTA 2004: 252-262

C. Gerede and J. Su: *Specification and Verification of Artifact Behaviors in Business Process Models*. ICSOC 2007: 181-192

C. Gerede, K. Bhattacharya, and J. Su: *Static Analysis of Business Artifact-centric Operational Models*. SOCA 2007: 133-140

M. Koshkina and F. van Breugel: *Modelling and verifying web service orchestration by means of the concurrency workbench*. ACM SIGSOFT Software Engineering Notes 29(5): 1-10 (2004)

S. Merz: Model Checking: *A Tutorial Overview*. MOVEP 2000: 3-38

S. Nakajima: *Model-Checking of Safety and Security Aspects in Web Service Flows*. ICWE 2004: 488-501

A. Nigam and N. Caswell: *Business artifacts: An approach to operational specification*. IBM Systems Journal 42(3): 428-445 (2003)

M. Pistore, P. Traverso, P. Bertoli, and A. Marconi: *Automated Synthesis of Composite BPEL4WS Web Services*. ICWS 2005: 293-301

G. Salaun, L. Bordeaux, and M. Schaerf: *Describing and Reasoning on Web Services using Process Algebra*. ICWS 2004: 43-51

G. Salaun, A. Ferrara, and A. Chirichiello: *Negotiation Among Web Services Using LOTOS/CADP*. ECOWS 2004: 198-212

M. Spielmann: *Verification of relational transducers for electronic commerce*. J. Comput. Syst. Sci. 66(1): 40-65 (2003)

V. Vianu: *Automatic verification of database-driven systems: a new frontier*. ICDT 2009: 1-13