Synthesis of State Machine Diagrams from Communication Diagrams by Using Petri Nets

T. Miyamoto†, H. Kurahata†, T. Fujii‡, R. Hosokawa‡

†Osaka University, Japan
‡Osaka Gas Information System Research Institute, Japan

SOA (Service Oriented Architecture)

Service Unit providing a specific function

Example Customer management, Shipping, etc.

Key Principles by Thomas Erl (2004)

- Loose coupling
- Service contract
- Autonomy
- Reusability
- Composability
- Discoverability

System Composition of services

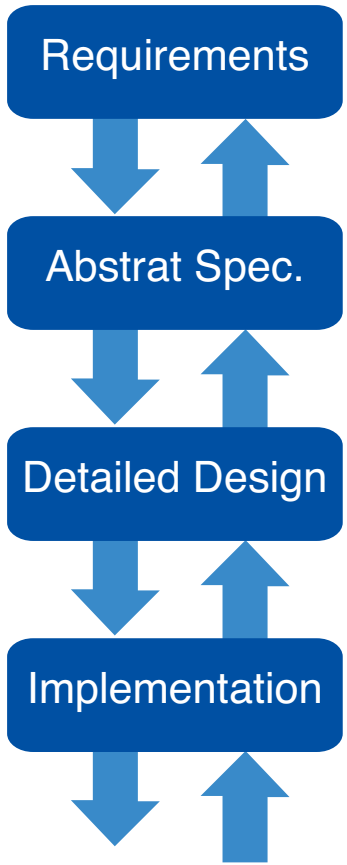
Application Business processes, Embedded system

Motivation

- Not easy to draw a detailed design from req.
- Much easier to write an abstract spec.
- Great help if a detailed design is generated from abstract spec.
- Improve quality of the developed system if automatically generated from a good spec.

Objective

Construct a detailed design from an abstract spec.



Choreography Realization Problem

- (C, I) : Choreography model
- C : Choreography
  - Messages between services
  - Observable events of services
- I : Set of service implementations
- $\mathcal{L}(C)$: Behavior allowed be choreography C
- $\mathcal{L}(I)$: Behavior generated by service implementations I

Realizability problem

Given: C

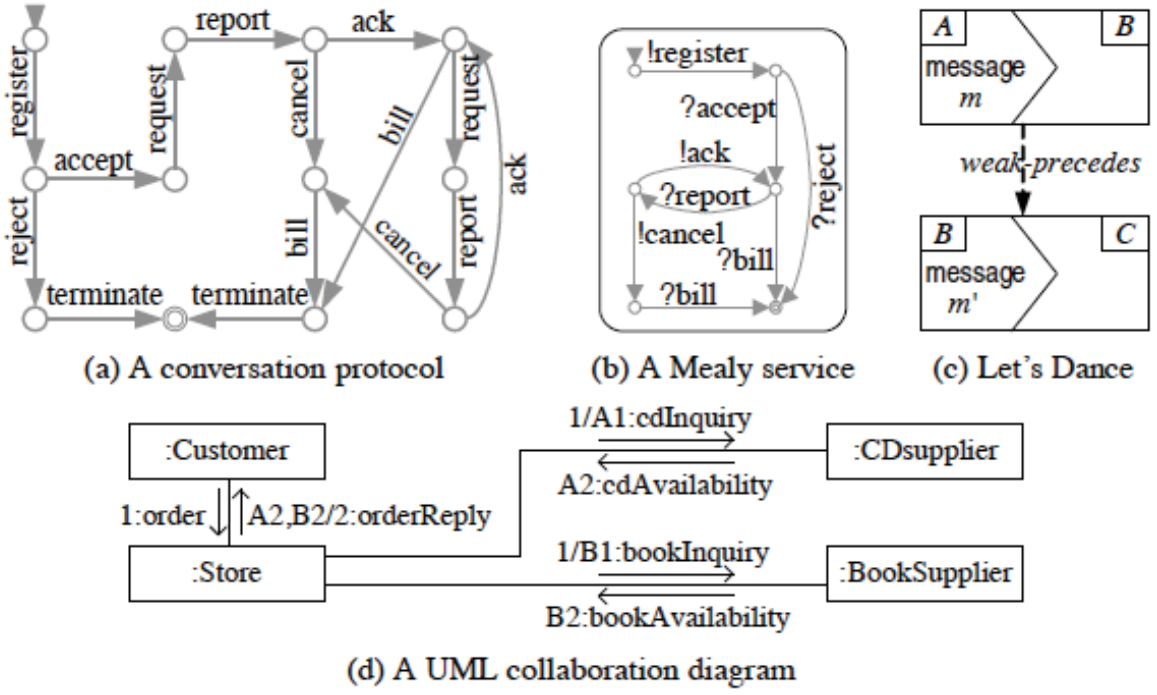
Is C realizable? When realizable, obtain service implementations I.

- C is realizable : $\exists I$, such that $\mathcal{L}(C) = \mathcal{L}(I)$
- C is weakly realizable : $\exists I$ such that $\mathcal{L}(C) \supseteq \mathcal{L}(I)$

Choreography Description Languages

- Conversation Protocol
- Sequence chart
- Communication (Collaboration) diagram
- Process algebra
- Petri net

Partial order among messages



(a) A conversation protocol (b) A Mealy service (c) Let's Dance

(d) A UML collaboration diagram

Fig. 4. Automata based Choreography Modeling Languages

[J. Su, T. Bultan, X. Fu, and X. Zhao: Towards a Theory of Web Service Choreographies, 2007]

Service Implementation Description Language

- Statechart
- Mealey automaton
- UML state machine
- Labelled transition system
- Petri net

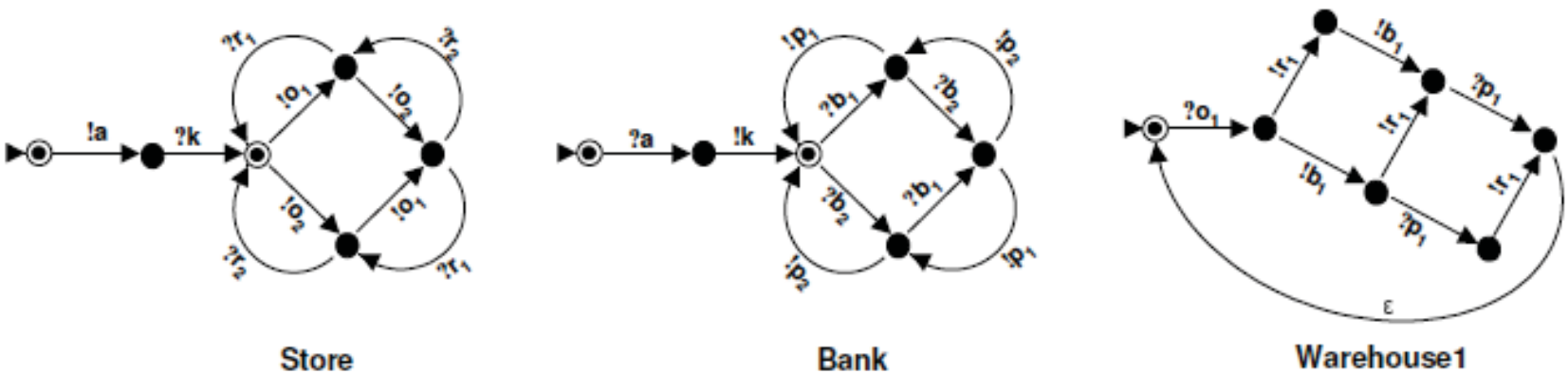


Figure 4: A Mealy implementation for the Warehouse example

[T. Bultan, X. Fu, R. Hull, and J. Su: Conversation Specification: A New Approach to Design and Analysis of E-Service Composition, 2003]

Related Works on Service Implementation Construction

Authors	Choreography	Service Impl.
T. Bultan et al.[9]	Collabo. Diagram	Mealy Automaton
G. Salaün et al.	Process Algebra	-
J. Hanson et al.	Mealy Automaton	Mealy Automaton
D. Harel[2]	LSC	Statechart
J. Whittle et al.[3]	MSC & OCL	Statechart
S. Leue et al.[4]	MSC	ROOM model
R. Lorenz et al.[7]	Po-set	Petri net

Conversation Protocol [J. Hanson et al.]

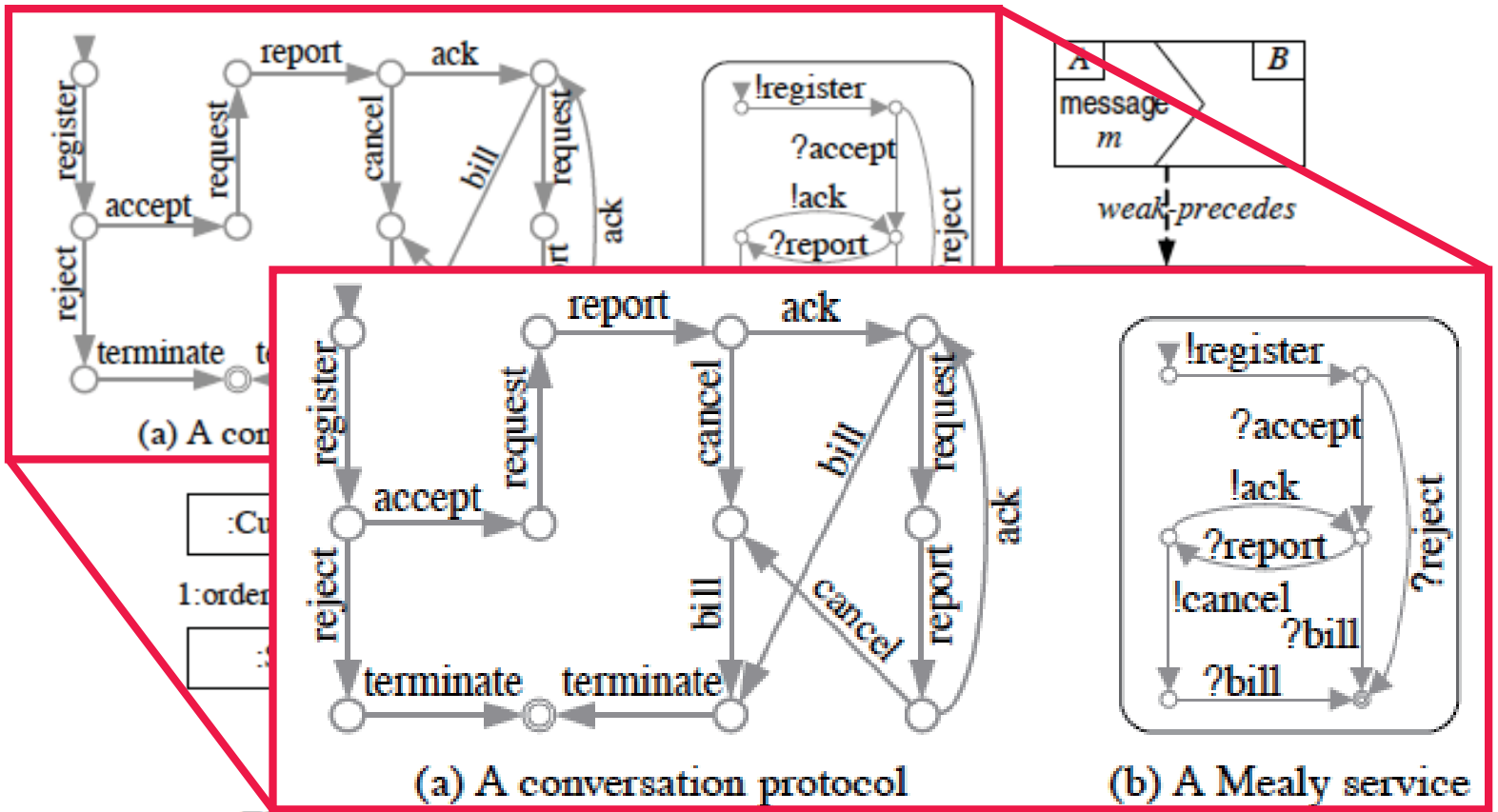
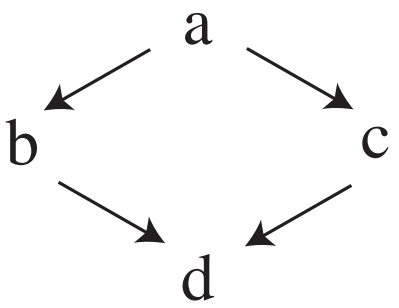


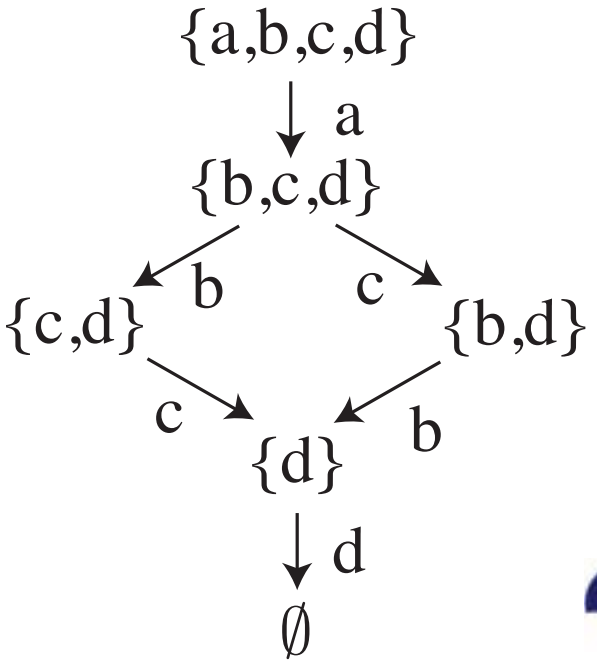
Fig. 4. Automata based Choreography Modeling Languages

[J. Su, T. Bultan, X. Fu, and X. Zhao: Towards a Theory of Web Service Choreographies, 2007]

Direct Projection [T. Bultan et al.[9]]



- M^s : a set of messages and observable events of service s
- $\Rightarrow^s$: partial order on M^s
- State machine $(\Sigma, S, s_0, \delta, F)$
  - $\Sigma = M^s$
  - $S \subseteq 2^{M^s}$
  - $s_0 = M^s$
  - $\delta : S \times \Sigma \rightarrow S$
  - $F = \{\emptyset\}$



Our Method

Problem

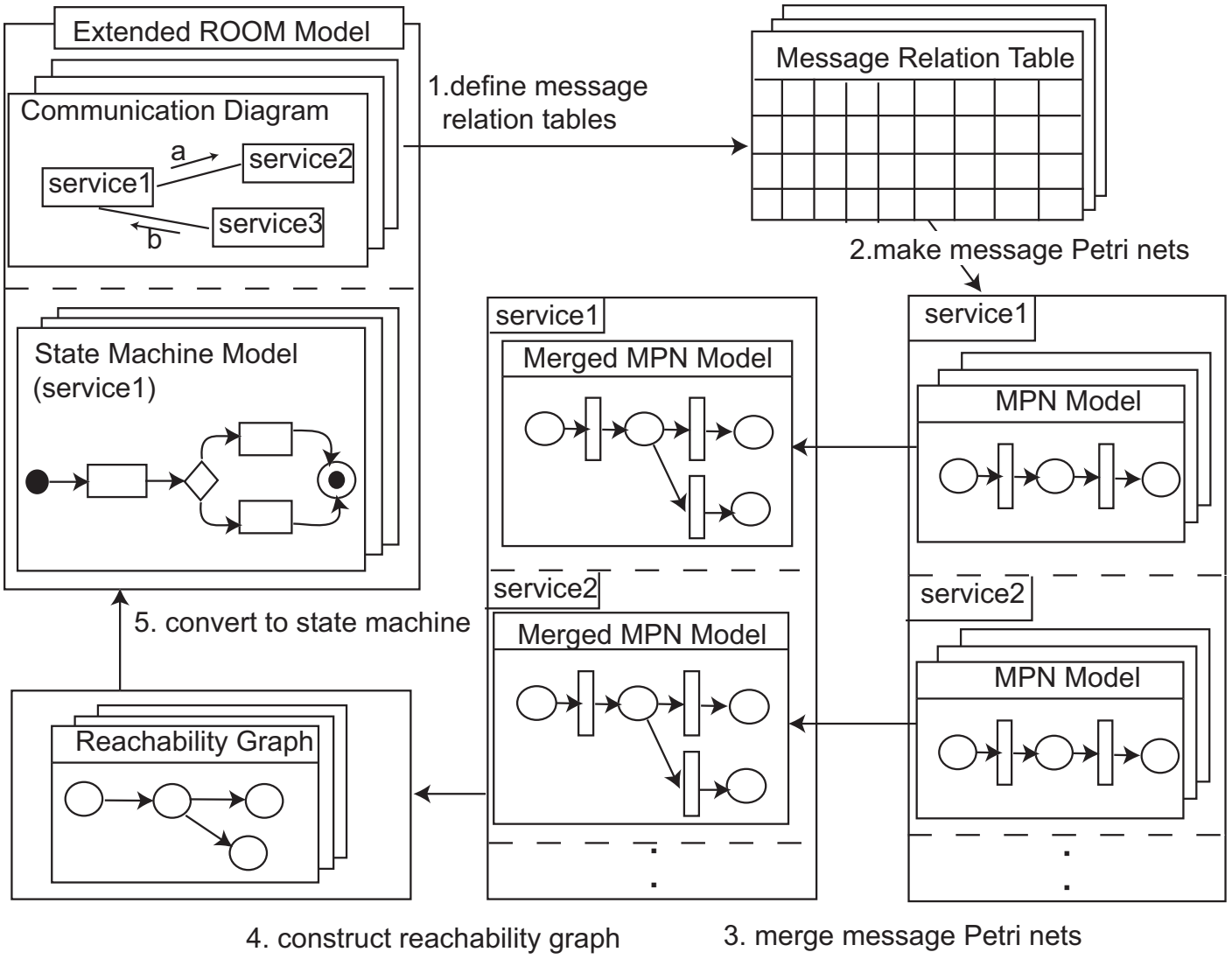
Input $C = \{(M, \Rightarrow)\}$: choreography
M : a set of messages
$\Rightarrow$: a partial order relation on M

Output $I = SM$: a set of state machines

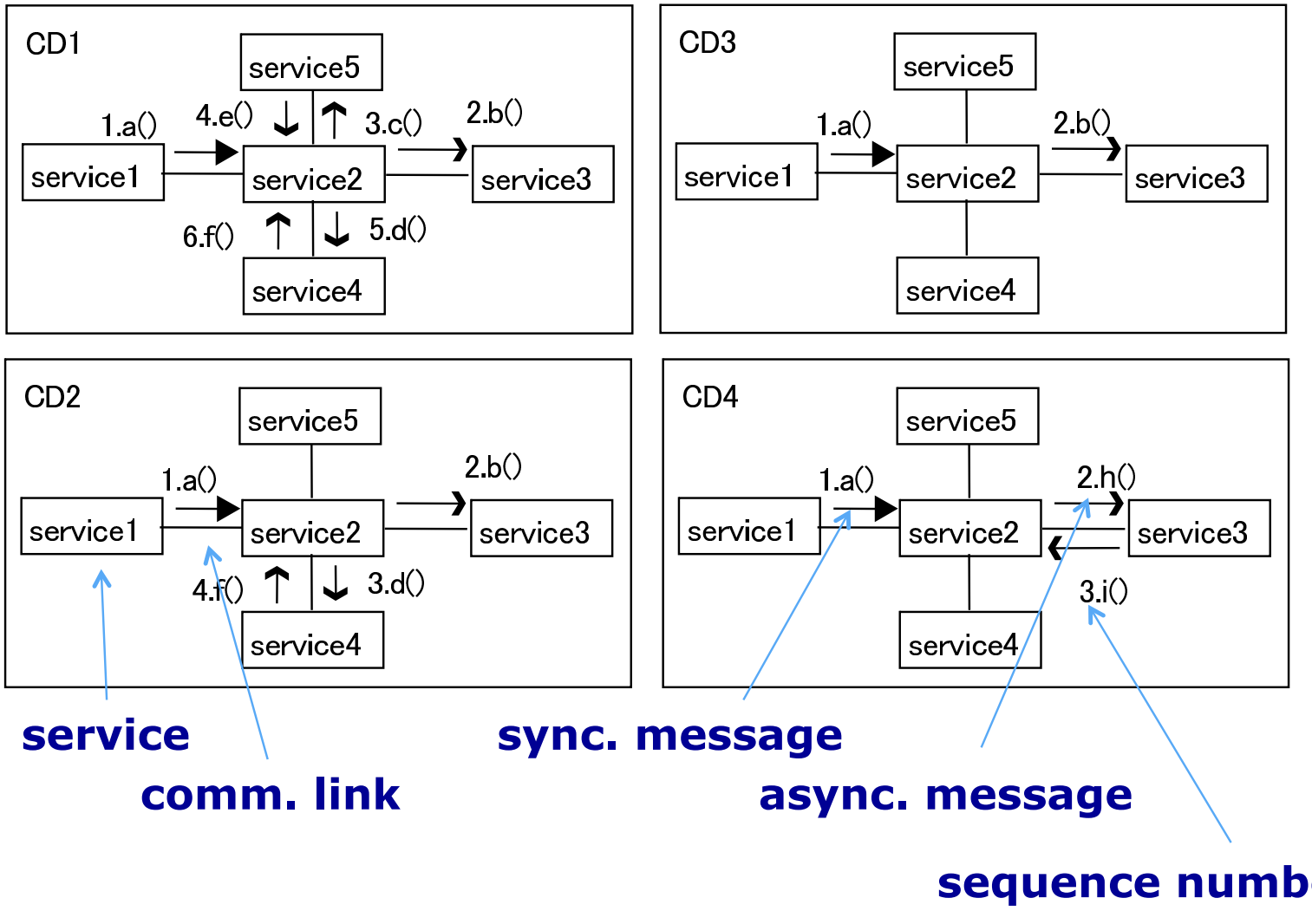
Subject to $\mathcal{L}(C) \subseteq \mathcal{L}(I)$

- Use communication diagram and message relation table to define partial order
- Use Petri nets to marge partial orders
- Try to use composite state of state machine for readability

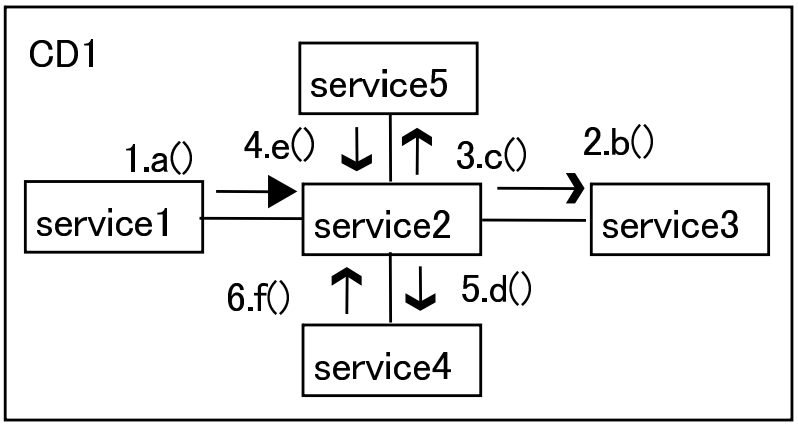
Construction Flow



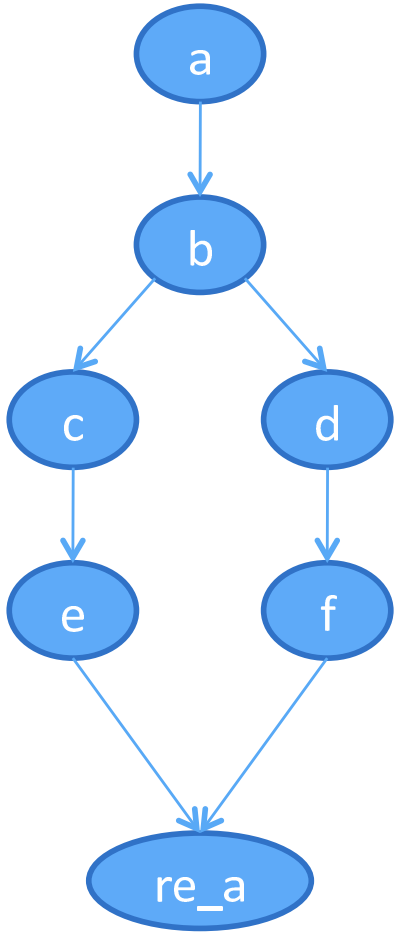
Communication Diagram to Draw Global Behavior



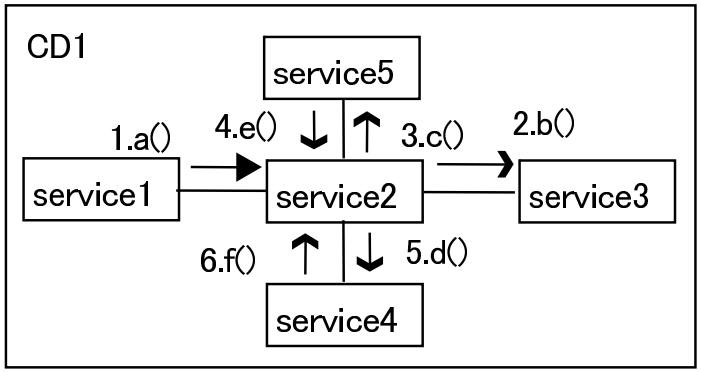
Use Table to Define Partial Order



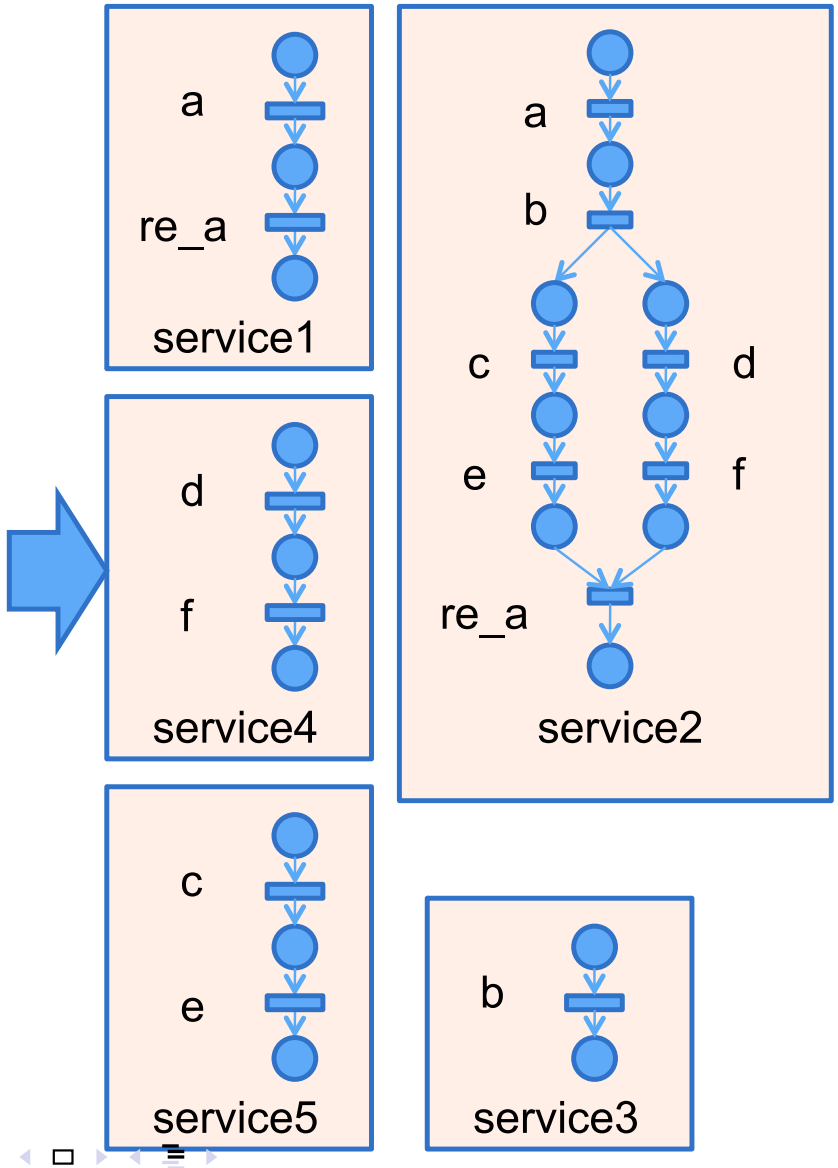
	a	b	c	d	e	f	re_a
a		→	→	→	→	→	→
b			→	→	→	→	→
c					→		→
d						→	→
e							→
f							→
re_a							



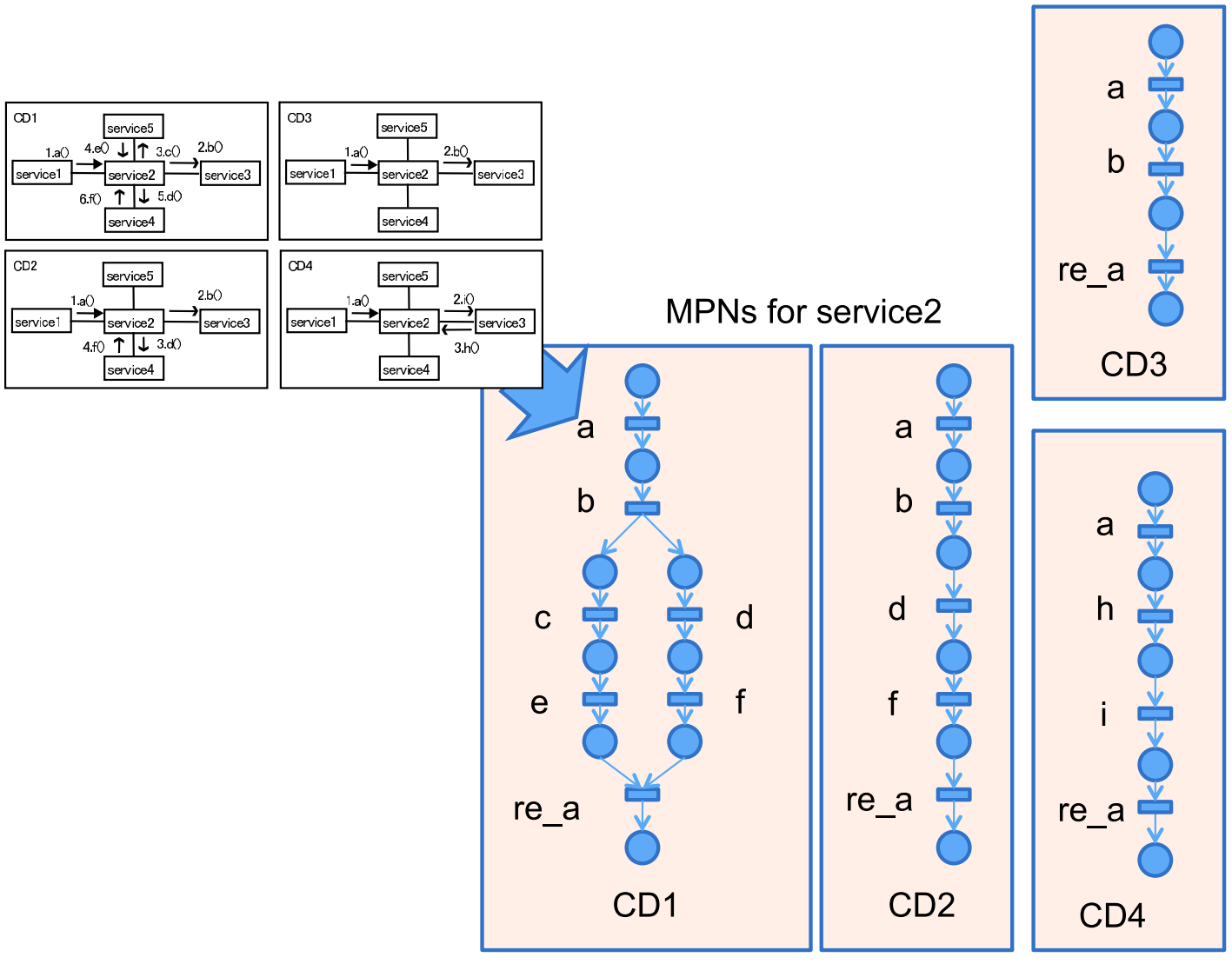
Derive a Petri Net for Each Service from a CD



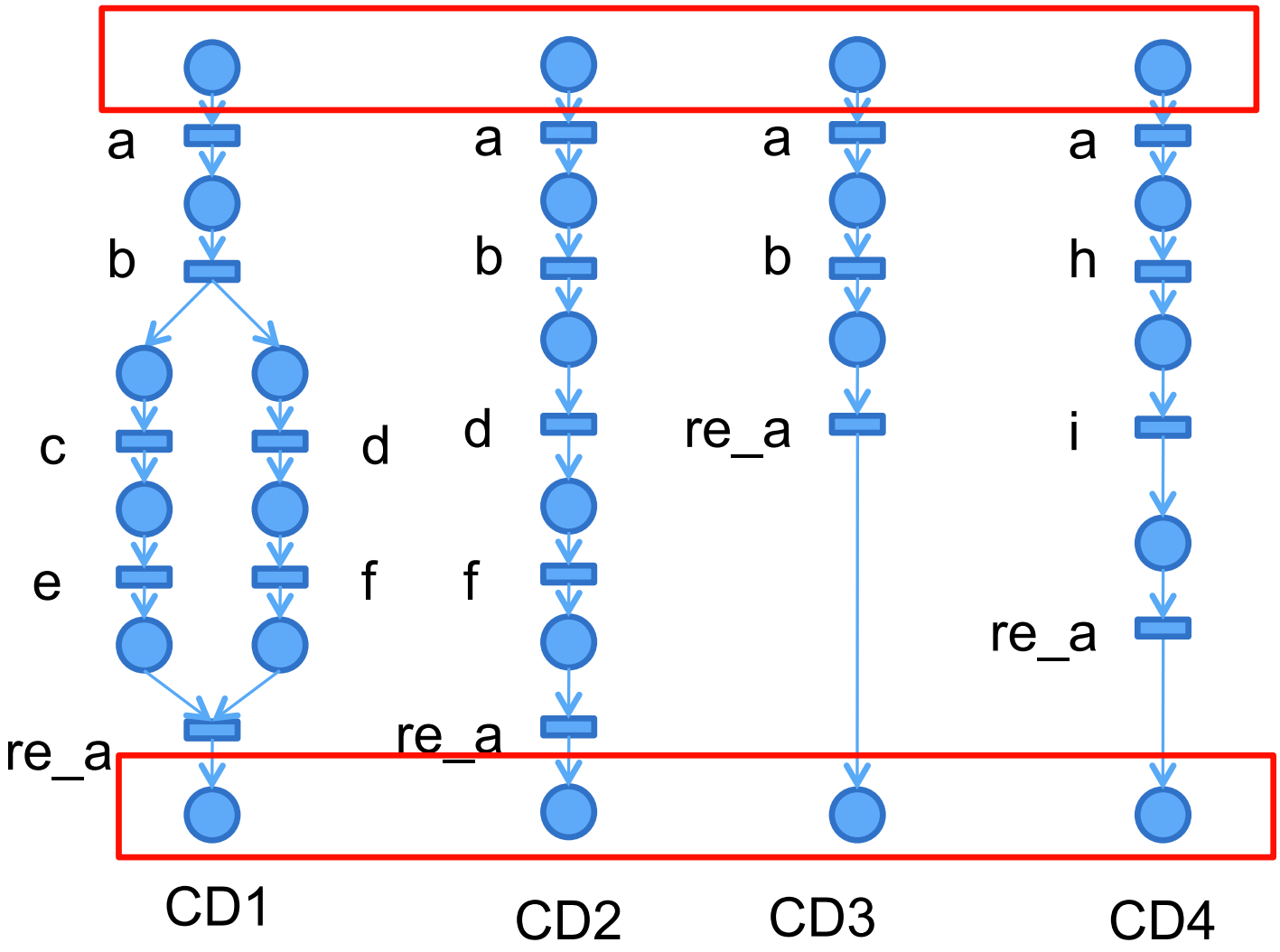
	a	b	c	d	e	f	re_a
a		→	→	→	→	→	→
b			→	→	→	→	→
c					→		→
d						→	→
e							→
f							→
re_a							



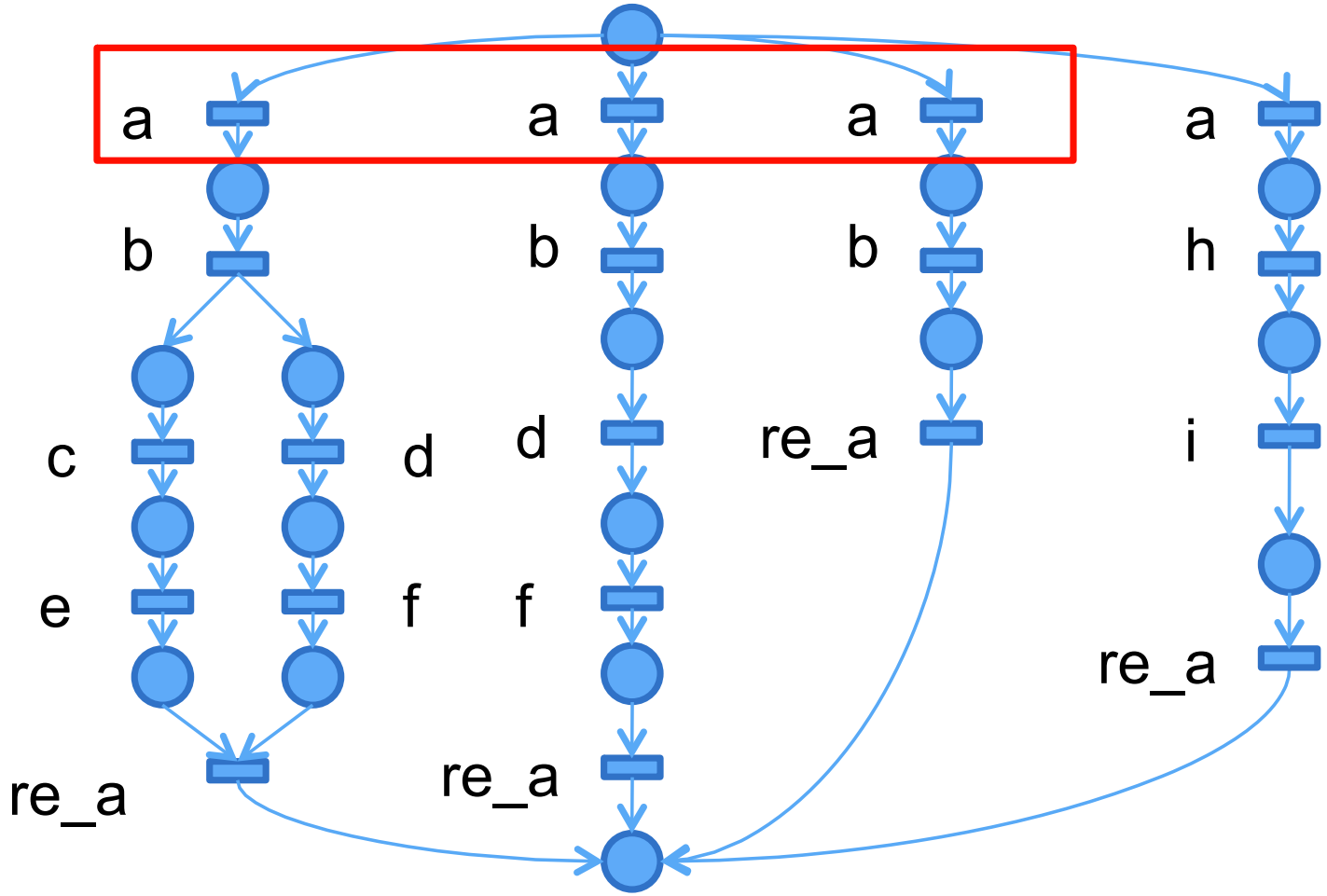
Gather Petri Nets for Each Service



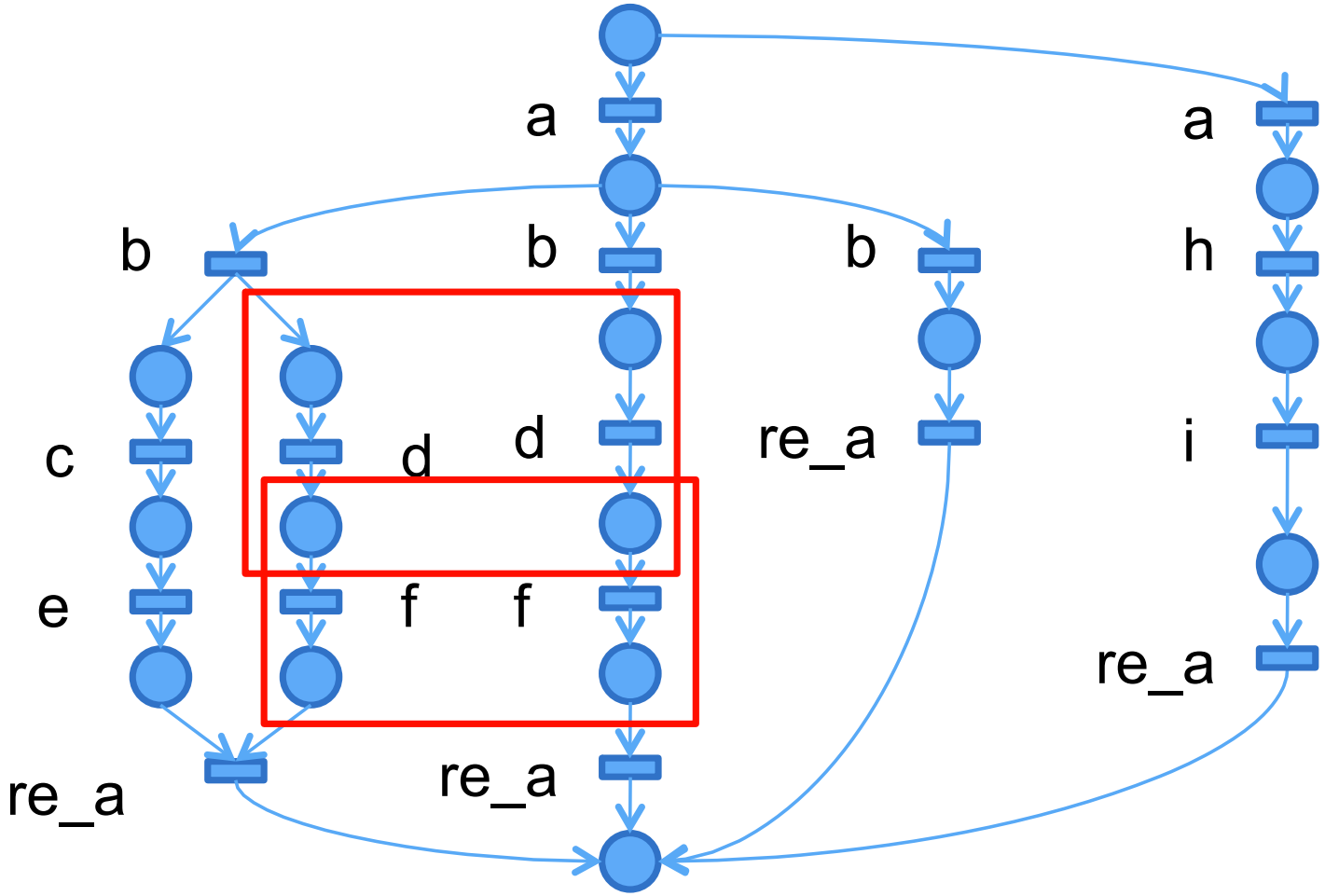
Marge Nodes on Petri Net



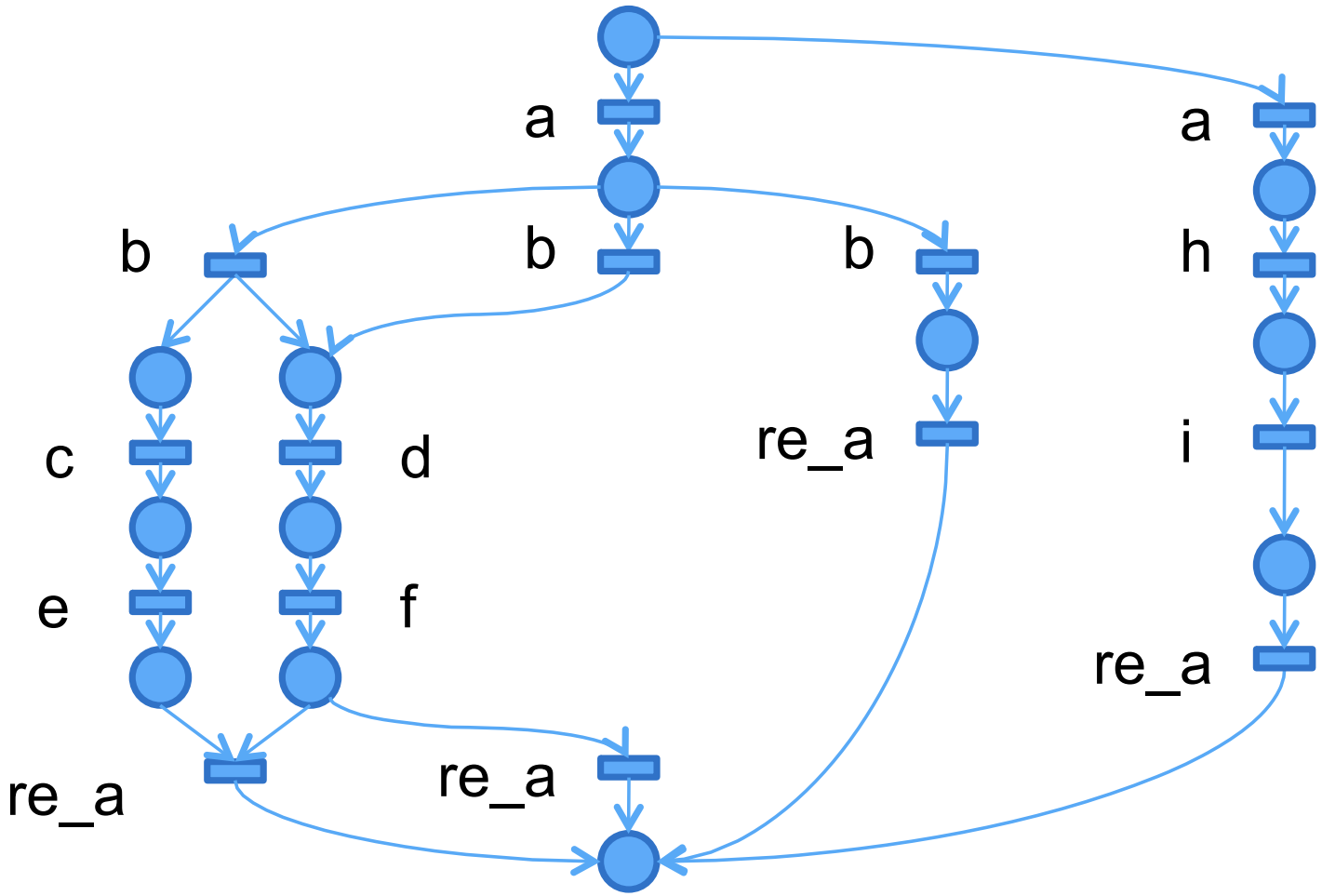
Marge Nodes on Petri Net



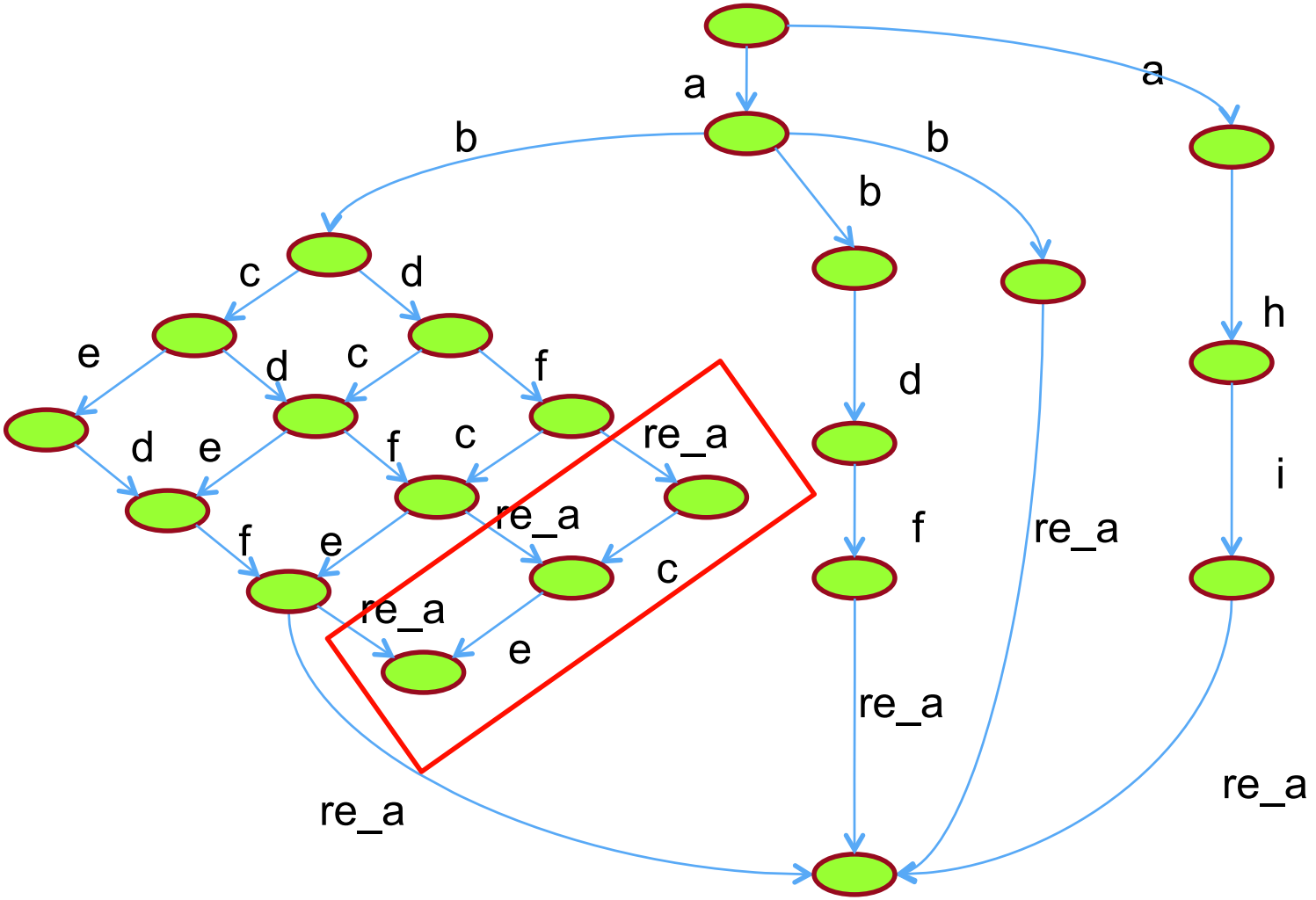
Marge Nodes on Petri Net



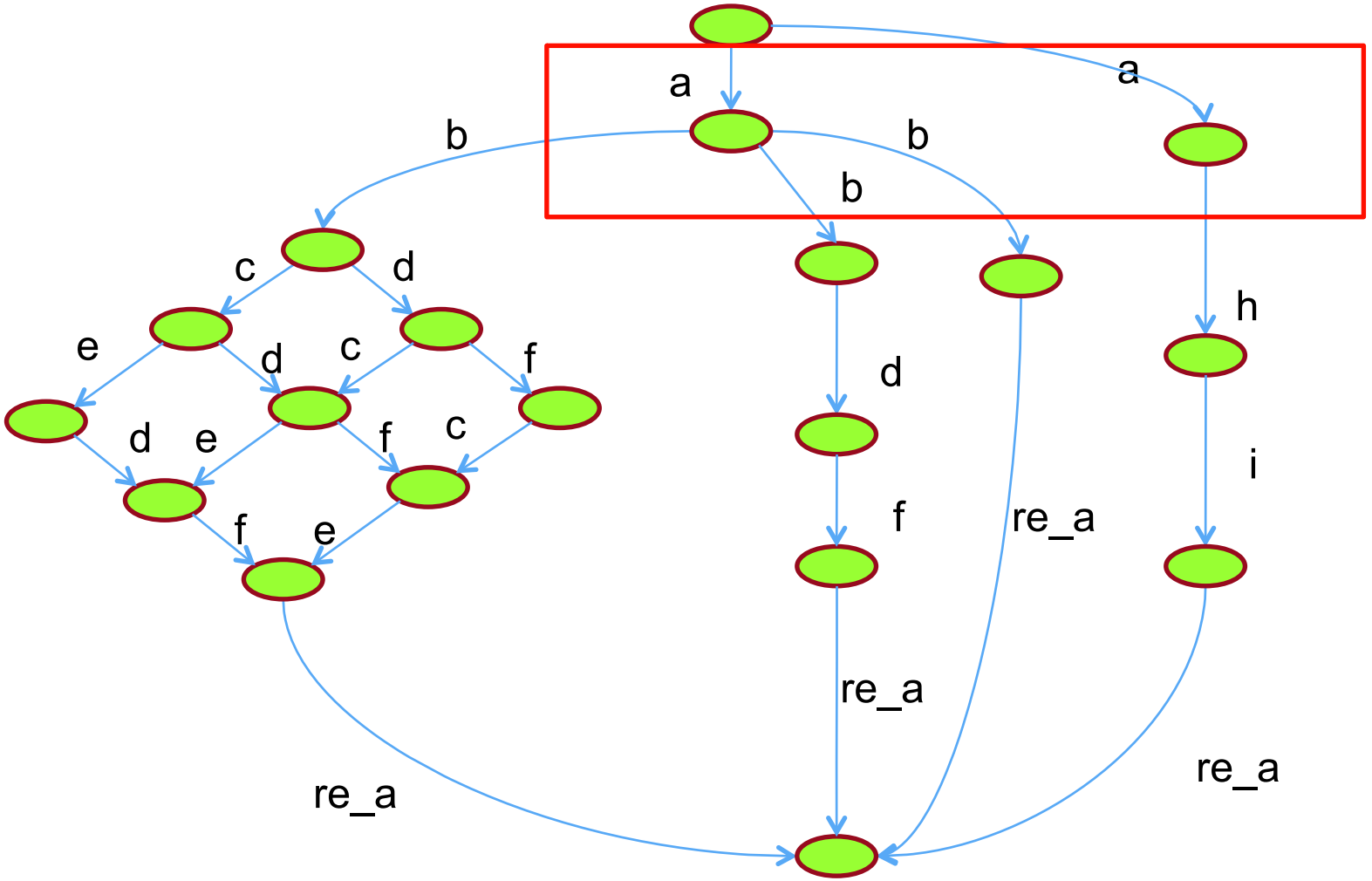
Marge Nodes on Petri Net



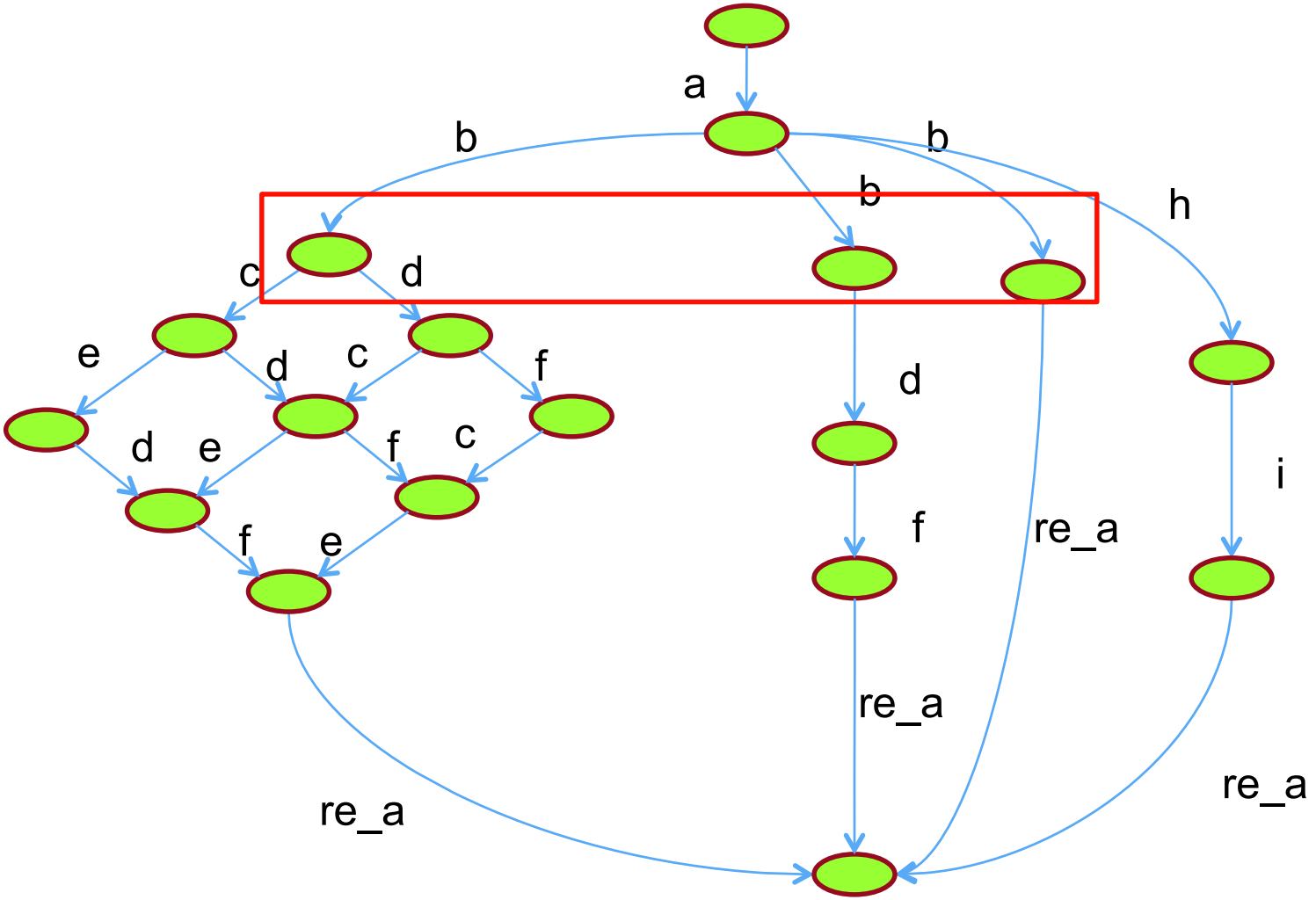
Construct Reachability Graph of Petri Net



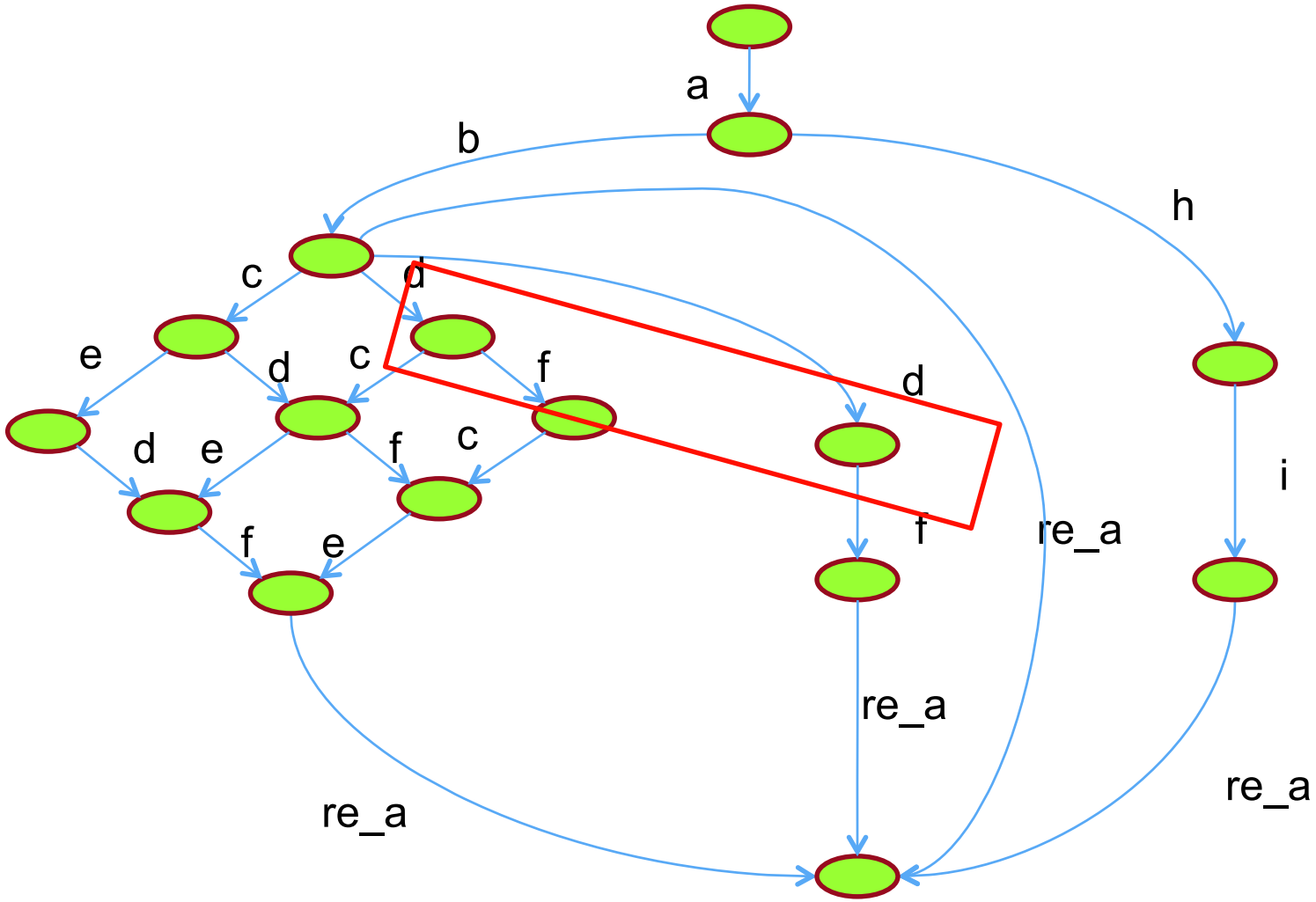
Simplify the Reachability Graph



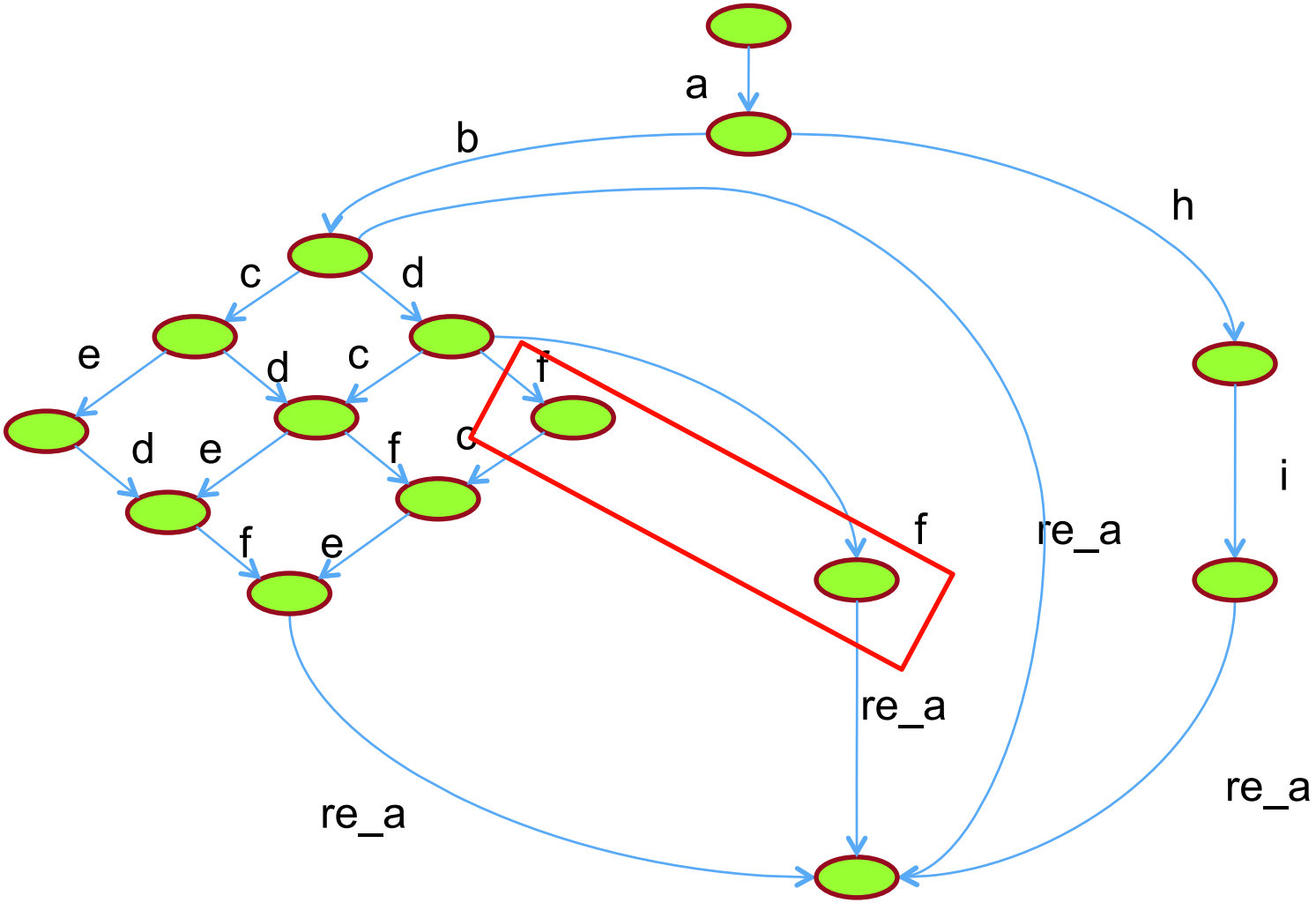
Simplify the Reachability Graph



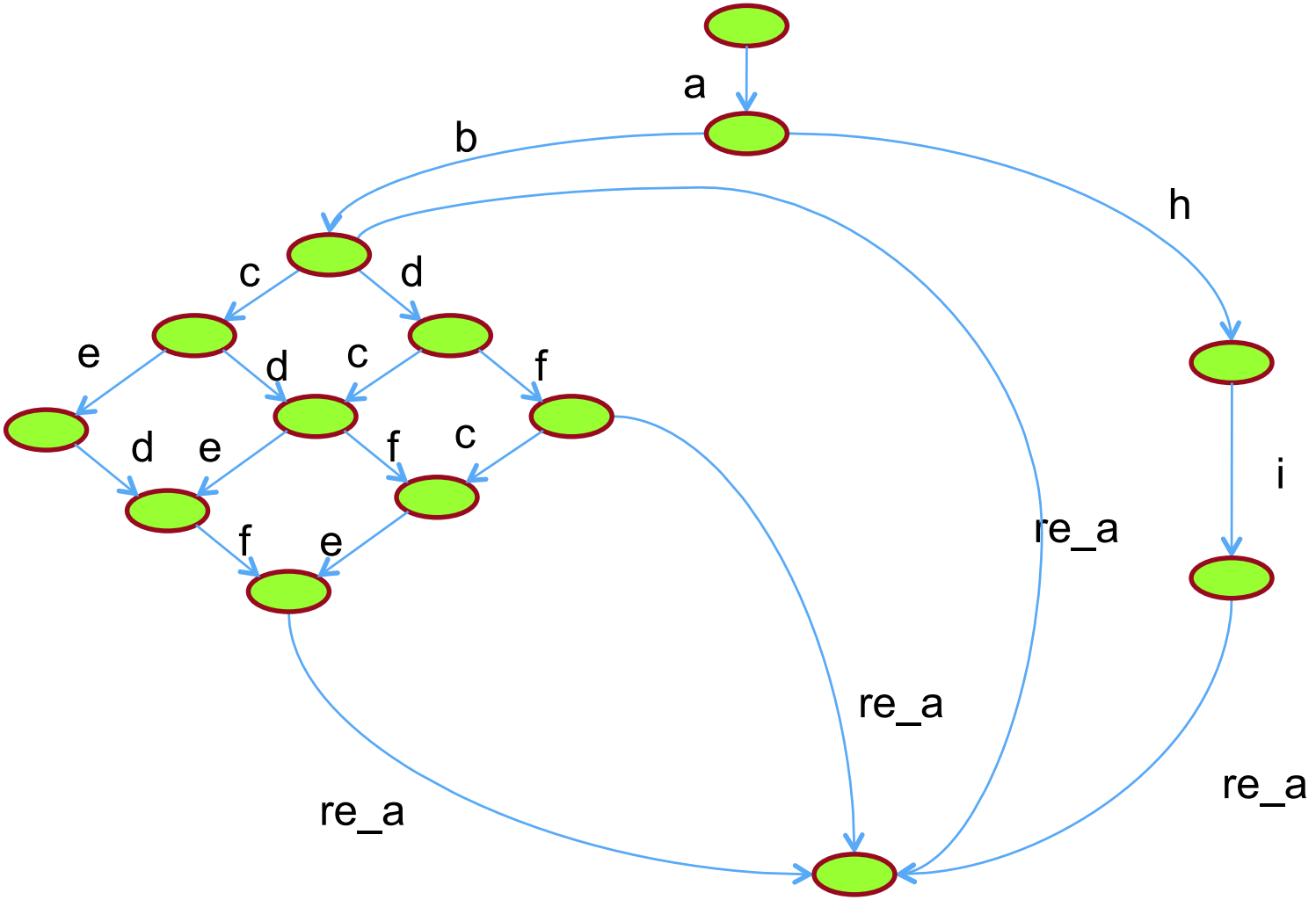
Simplify the Reachability Graph



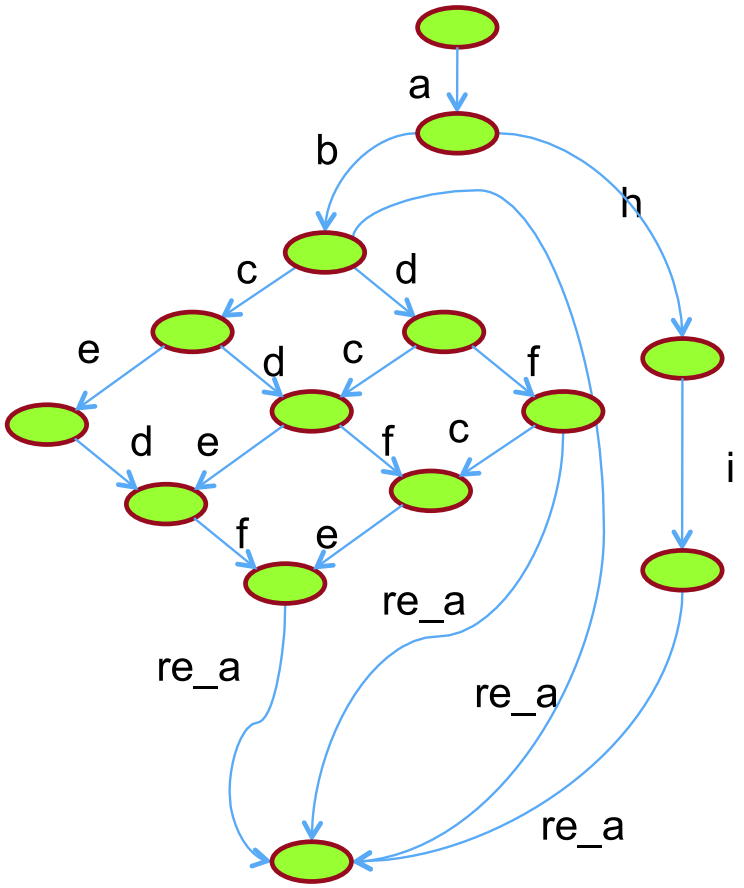
Simplify the Reachability Graph

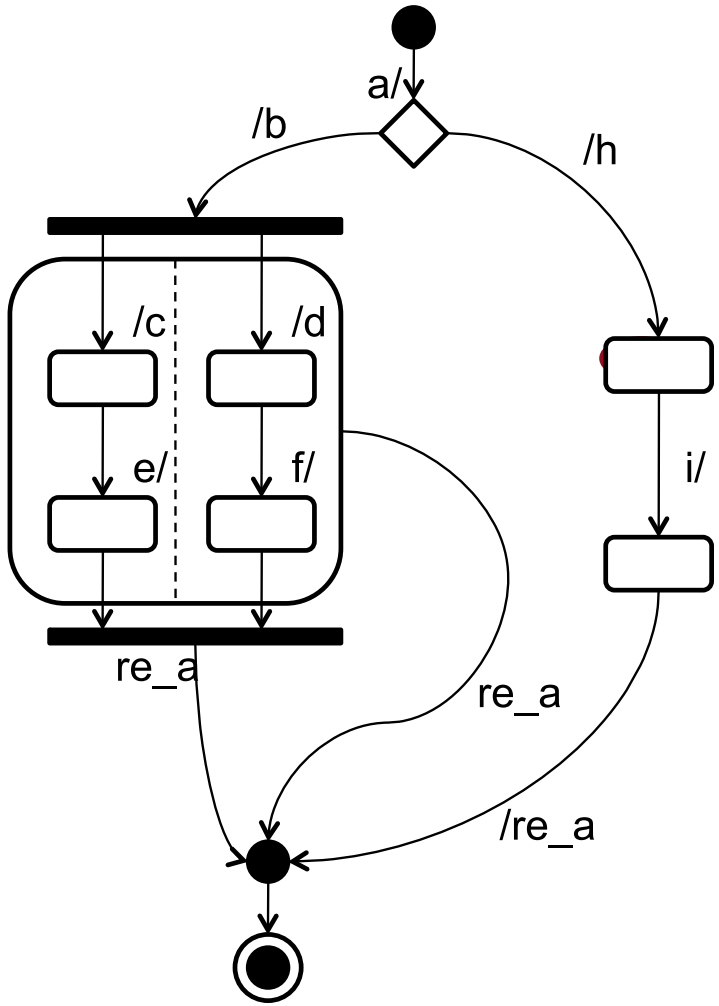


Simplify the Reachability Graph



Convert the Reachability Graph to State Machine





Conclusions

- Construct State Machine from Communication Diagrams
- Use communication diagram and message relation table to define partial order
- Use Petri nets to marge partial orders
- Try to use composite state of state machine for readability

Communication Diagram

	Communication diagram	Collaboration diagram
UML predecessor	2.x No	1.x Yes

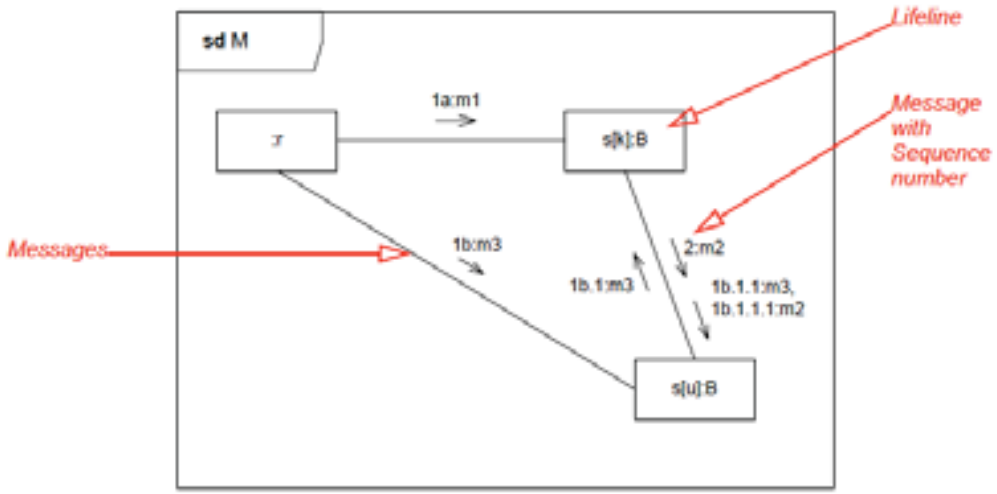
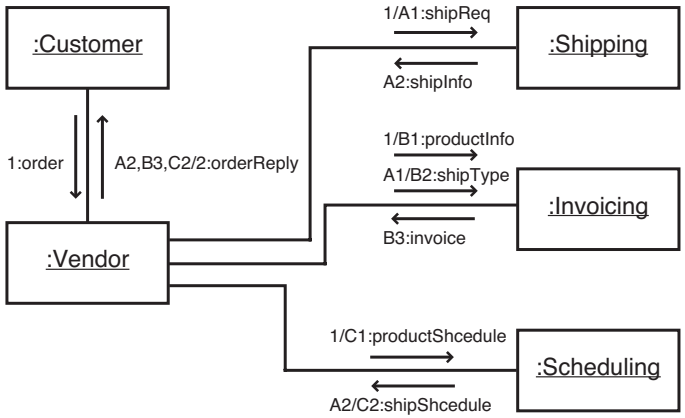


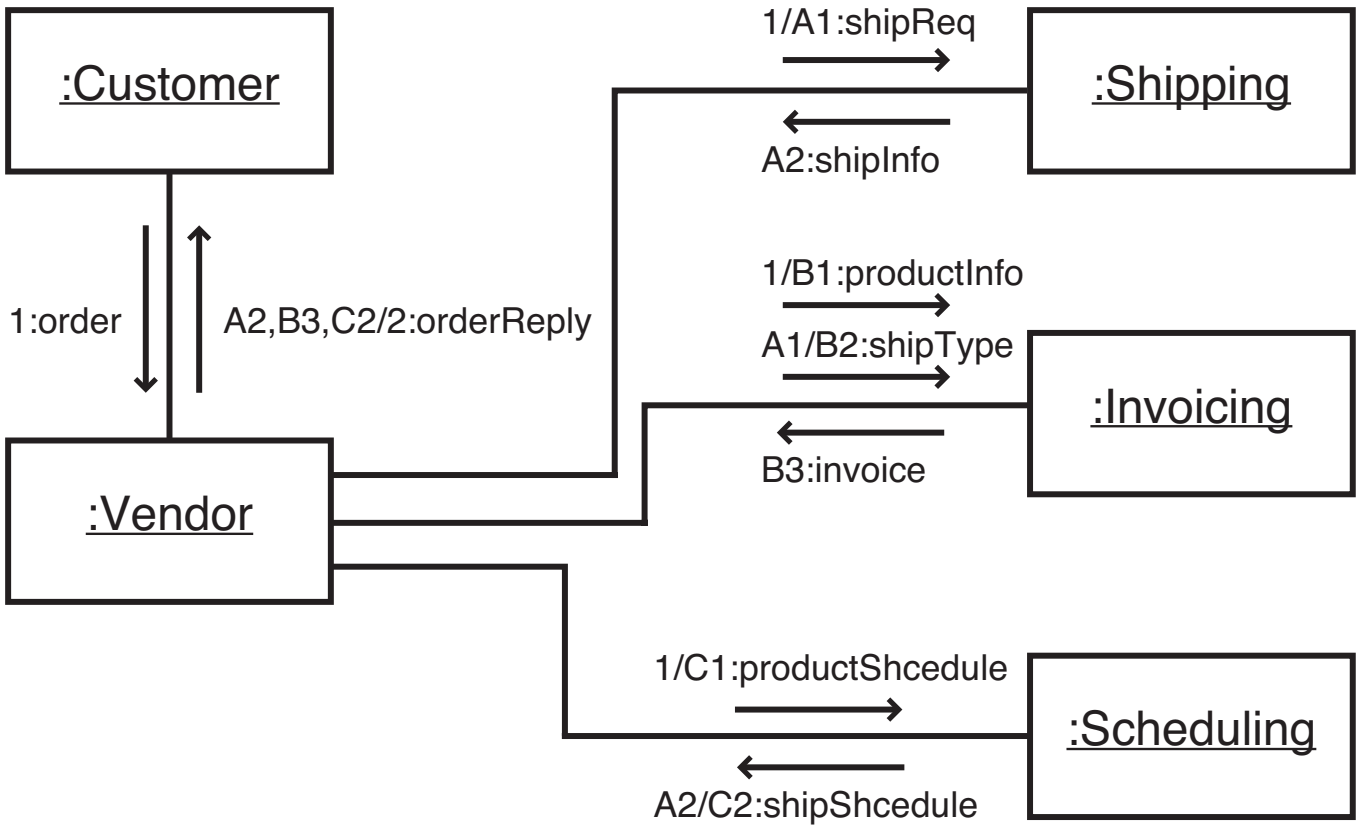
Figure 14.27 - Communication diagram

[OMG Unified Modeling Language, Superstructure]



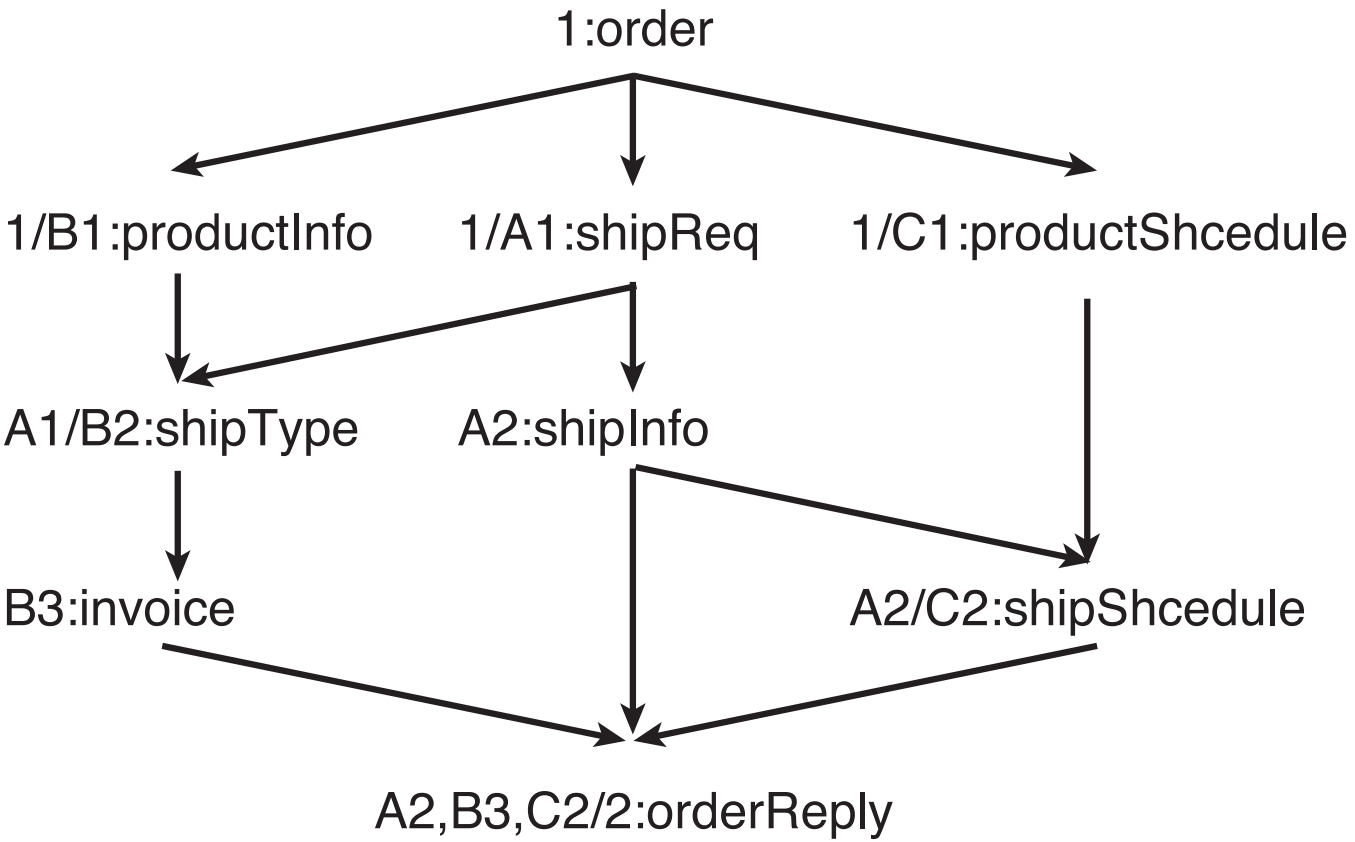
[T. Bultan and X. Fu, Specification of Realizable Service Conversations Using Collaboration Diagrams, 2008]

Purchase Order Process [WS-BPEL 2.0]



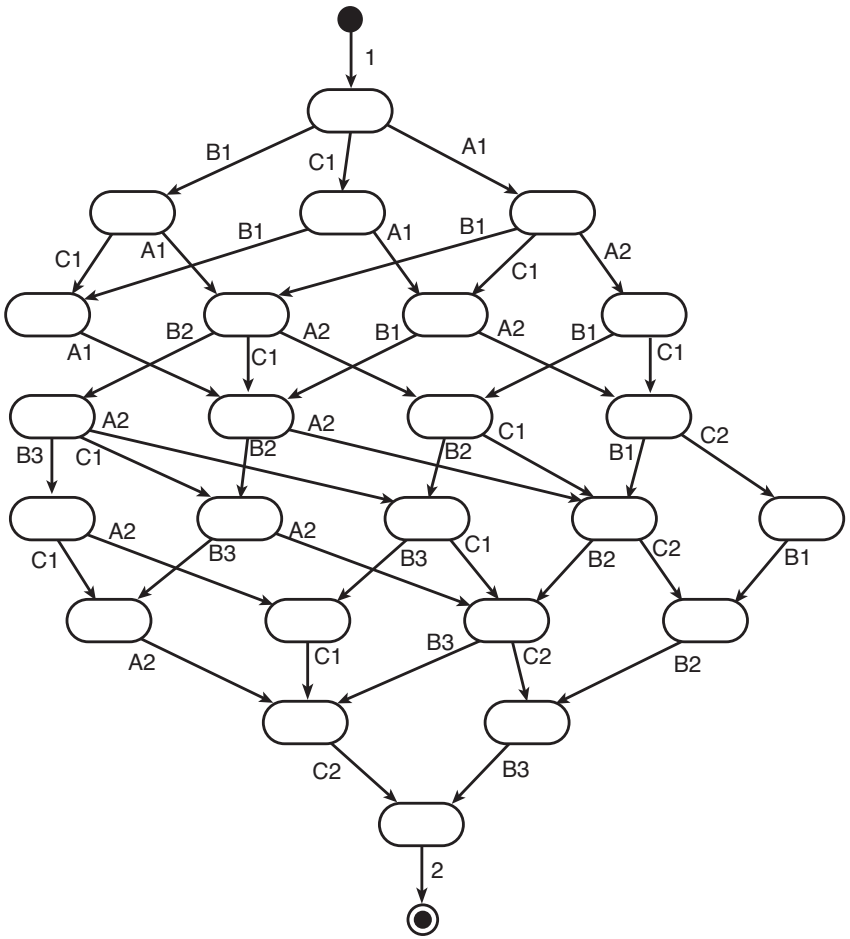
[T. Bultan and X. Fu, Specification of Realizable Service Conversations Using Collaboration Diagrams, 2008]

Partial Order for Vendor Process



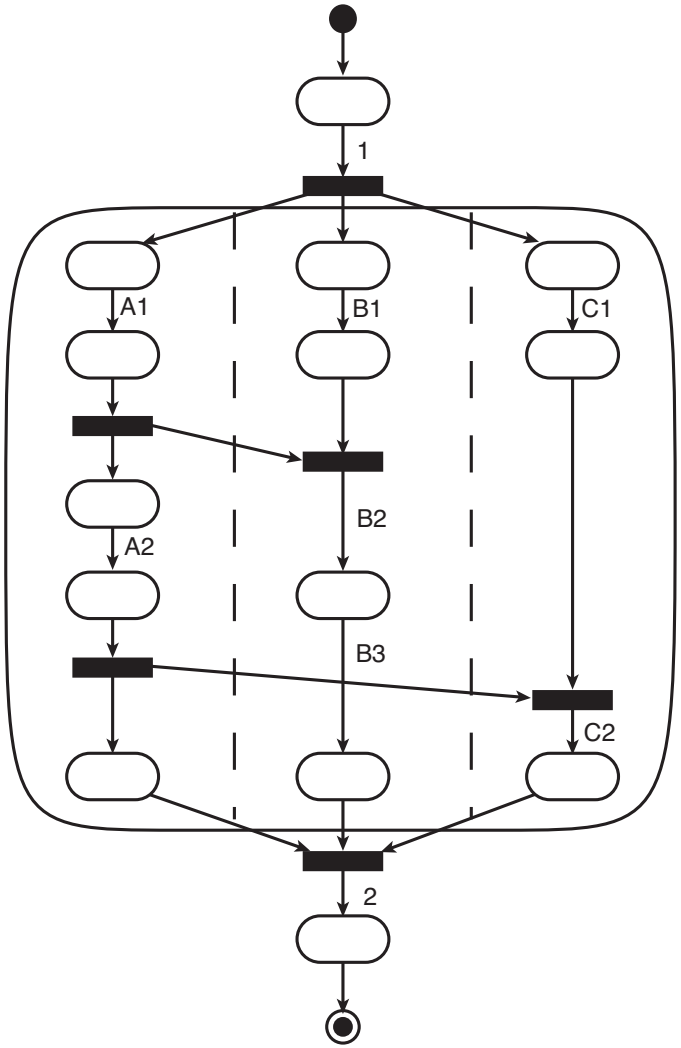
[T. Bultan and X. Fu, Specification of Realizable Service Conversations Using Collaboration Diagrams, 2008]

Service Implementation for Vendor Process



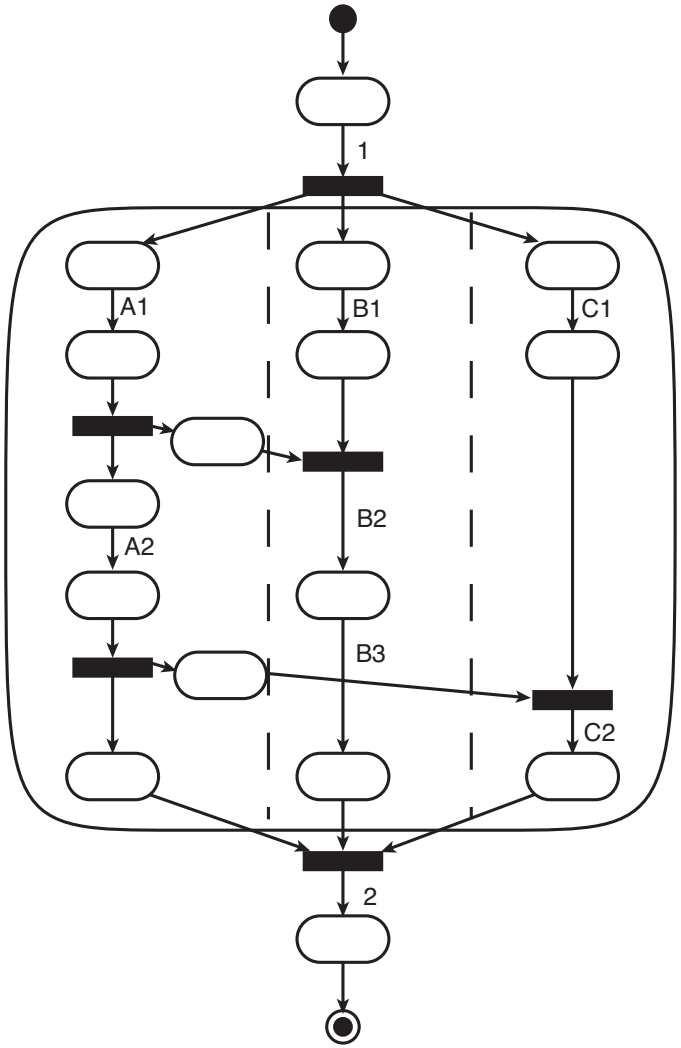
- by direct projection
- same by our method
- not adequate for revise

Use Composite States




```
(source.oclIsKindOf(Pseudostate)
 and source.kind = #fork) implies
   (target.oclIsKindOf(State)
```

Use Composite States




```
(self.kind = #fork)  implies
 self.outgoing->forAll (t1, t2 |
  t1<>t2 implies
  (self.stateMachine.LCA(t1.target,
                         t2.target).
   container.isOrthogonal))
```

State Machine Diagram and Activity Diagram

	State Machine Diagram	Activity Diagram
Concurrency	Orthogonal Regions	Petri Nets
Node	State	Transition
	(Place)	(Transition)

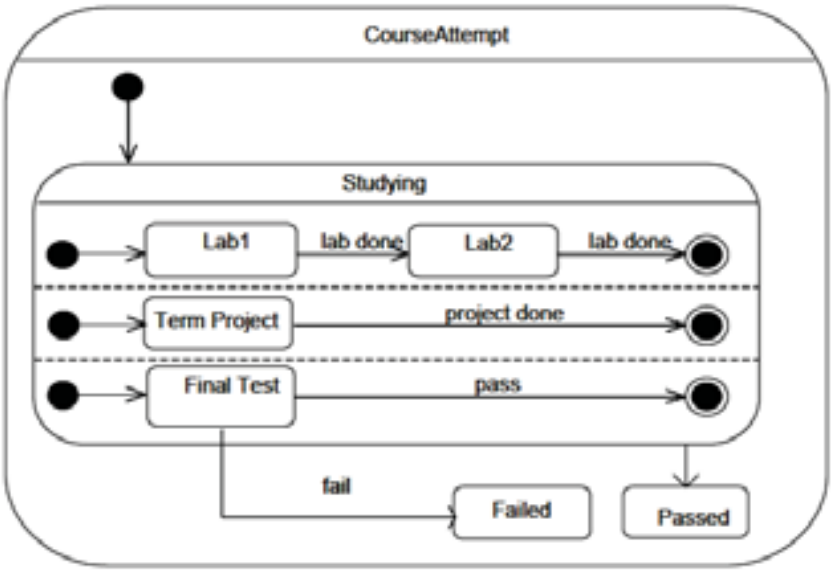


Figure 15.35 - Orthogonal state with regions

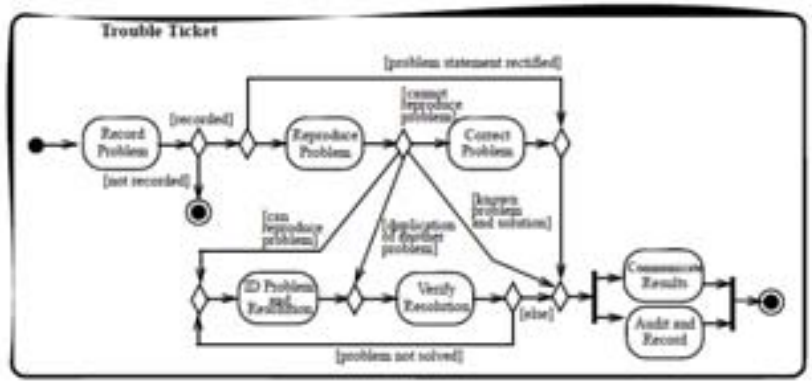


Figure 12.37 - Workflow example

[OMG Unified Modeling Language, Superstructure]