Service Interaction:
Patterns, Formalization, and Analysis

9th International School on Formal Methods for the Design of Computer, Communication and Software Systems: Web Services (SFM-09:WS), Bertinoro, Italy, June 1-6, 2009.

prof.dr.ir. Wil van der Aalst

www.vdaalst.org

Joint work with Arjan Mooij, Christian Stahl, and Karsten Wolf

TU/e Technische Universiteit **Eindhoven** University of Technology

Where innovation starts

BEST: Berlin - Rostock- Eindhoven Service Technology Program

Outline

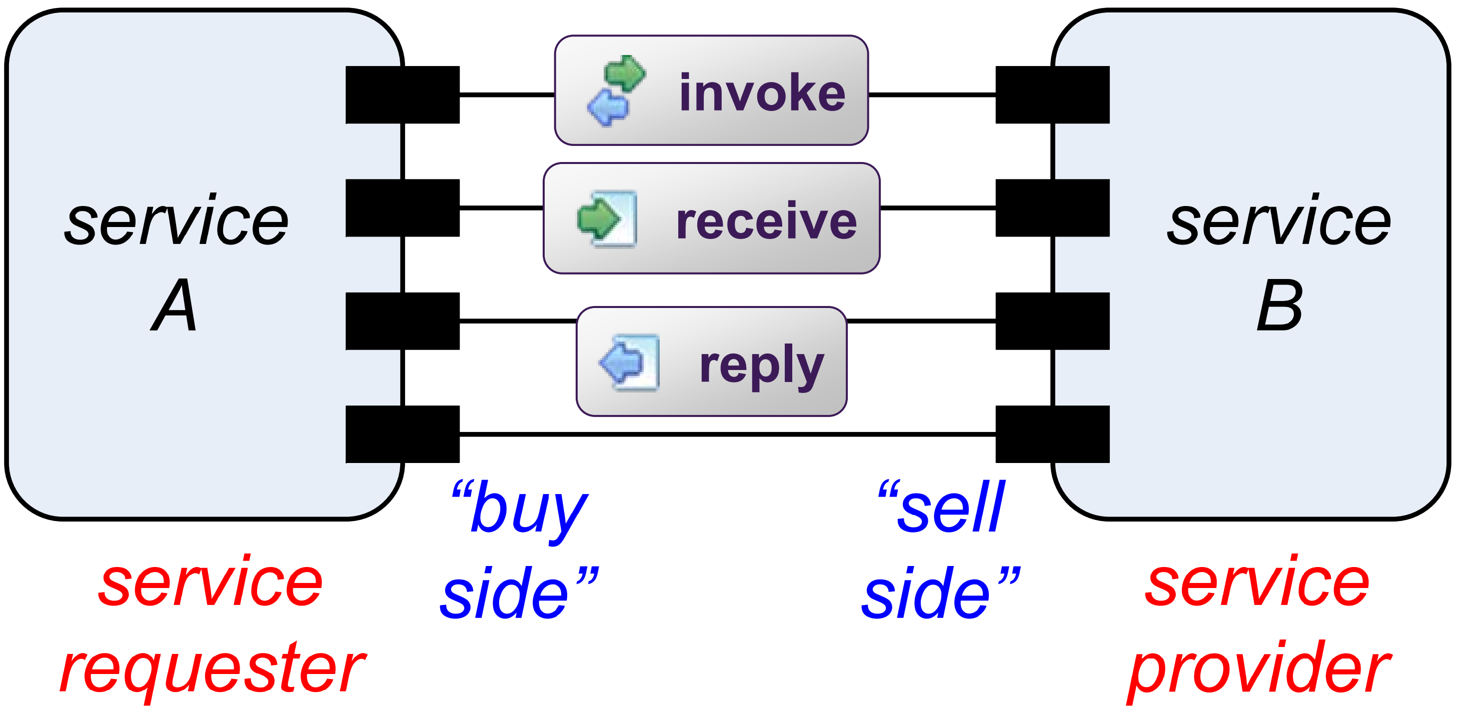
- Introduction to Service Interaction
- Workflow and Service Interaction Patterns
- Challenging Analysis Questions
- A "Crash Course" in Petri Nets
- Exposing Services
- Replacing and Refining Services
- Integrating Services Using Adapters
- Service Mining
- Conclusion

Introduction to Service Interaction

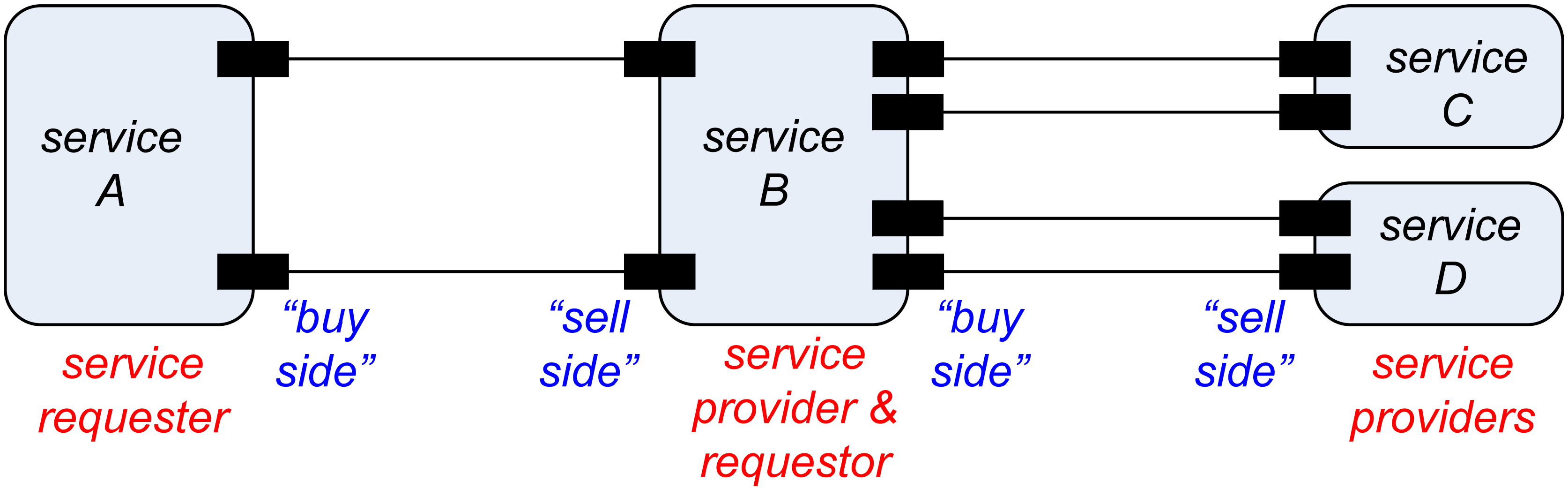
Technische Universiteit **Eindhoven** University of Technology

Where innovation starts

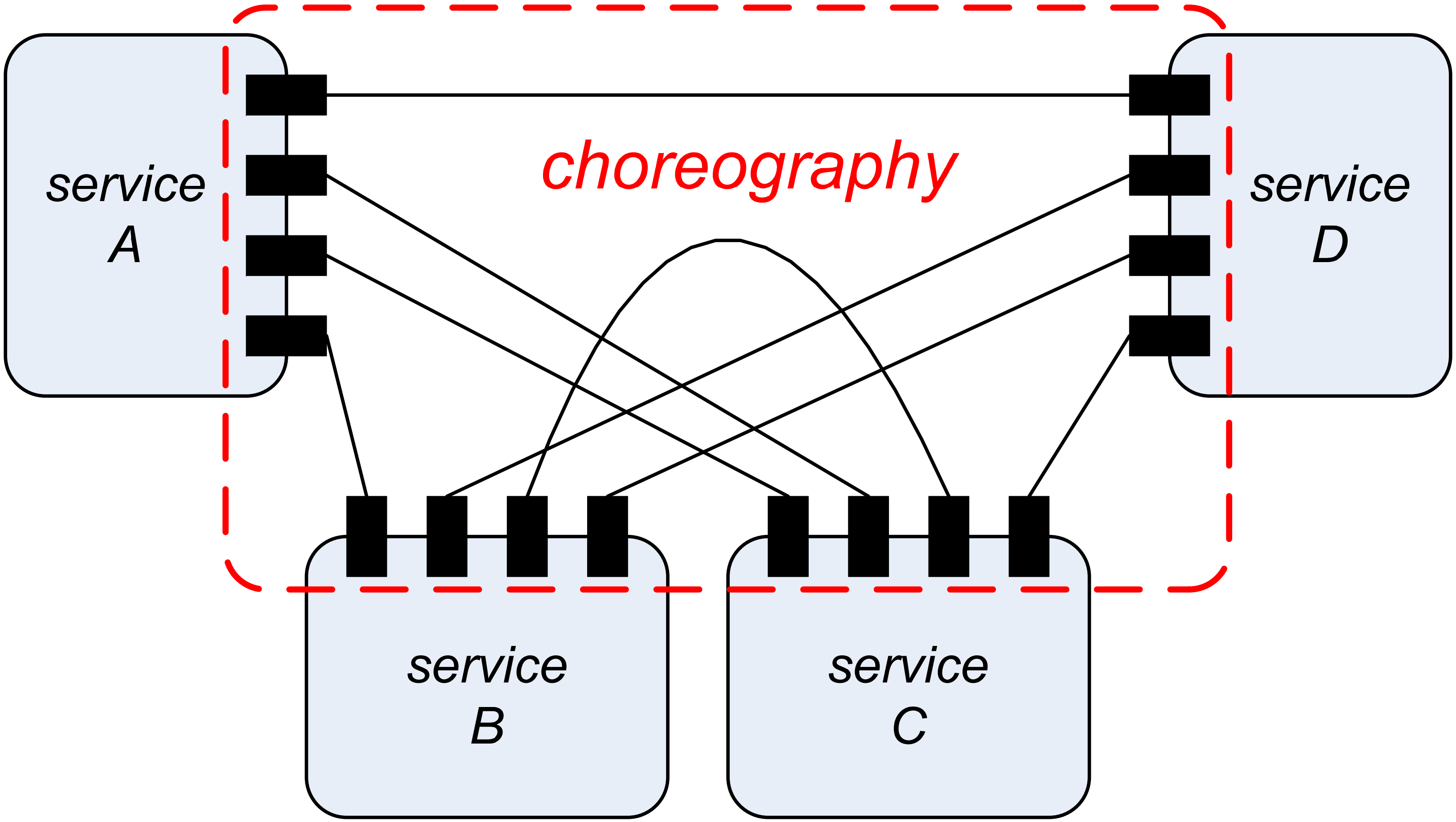
Service-Orientation: Basic Idea



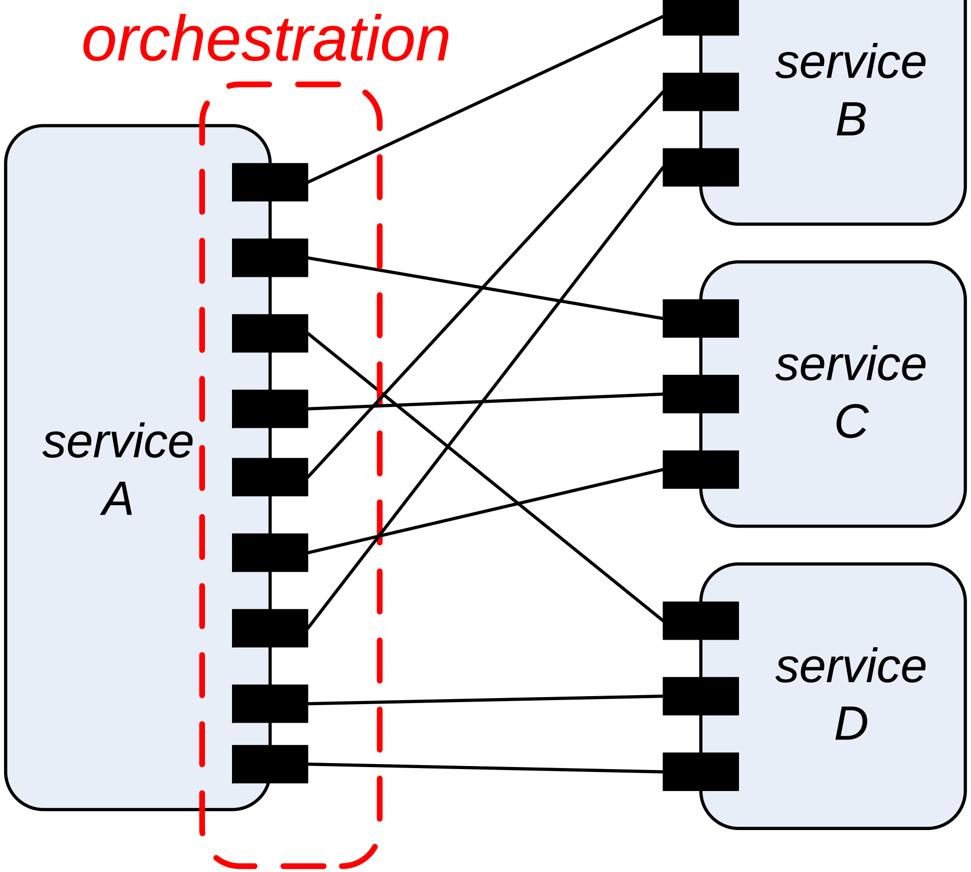
Service Networks



Choreography

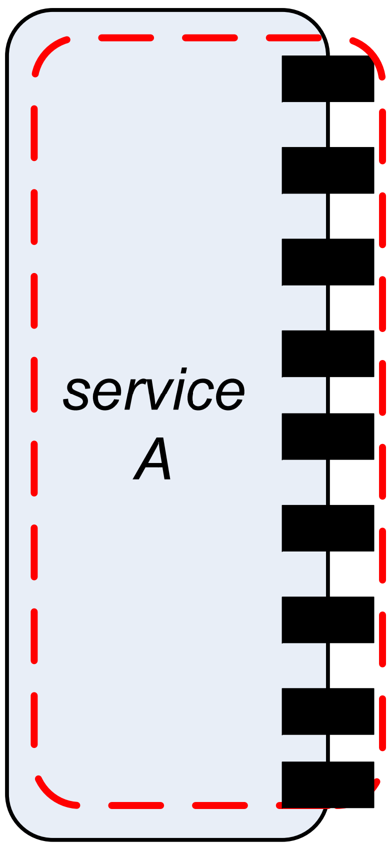


Orchestration

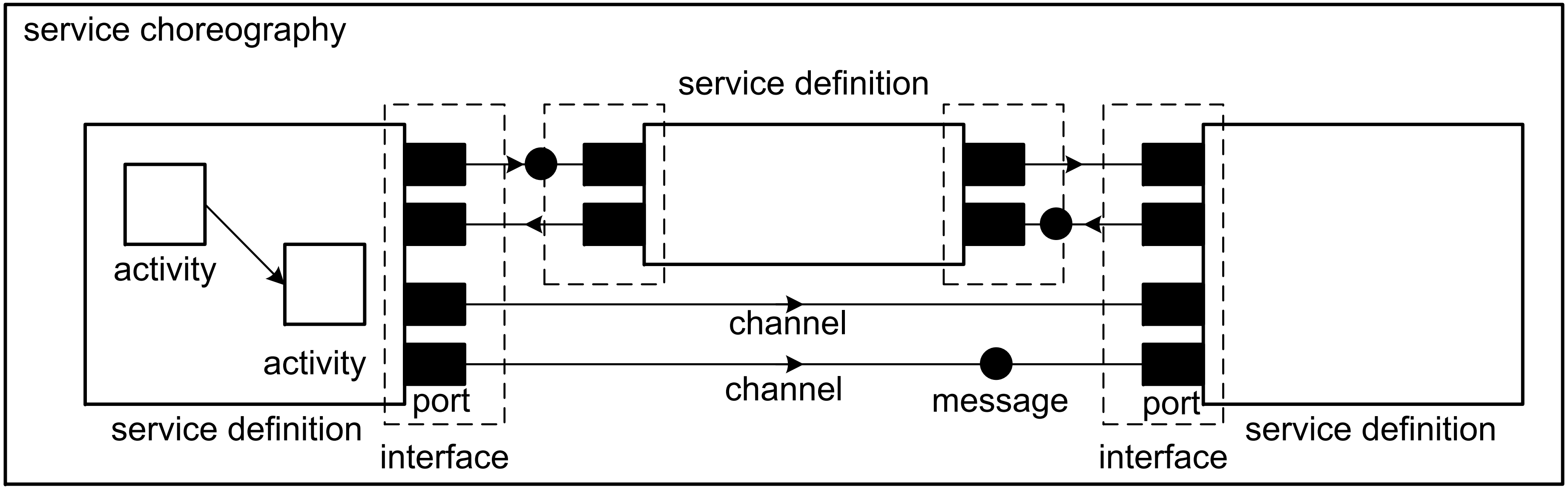


Workflow?

workflow in the classical sense

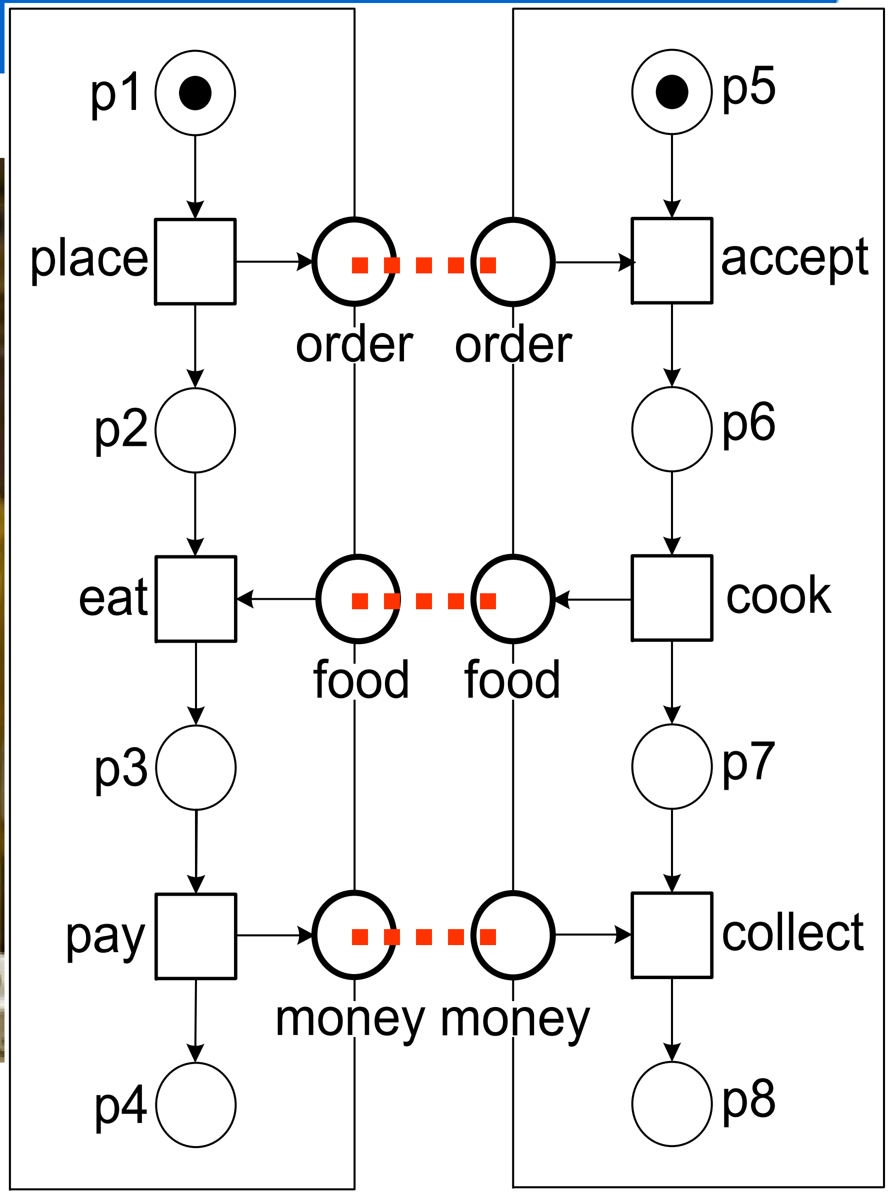


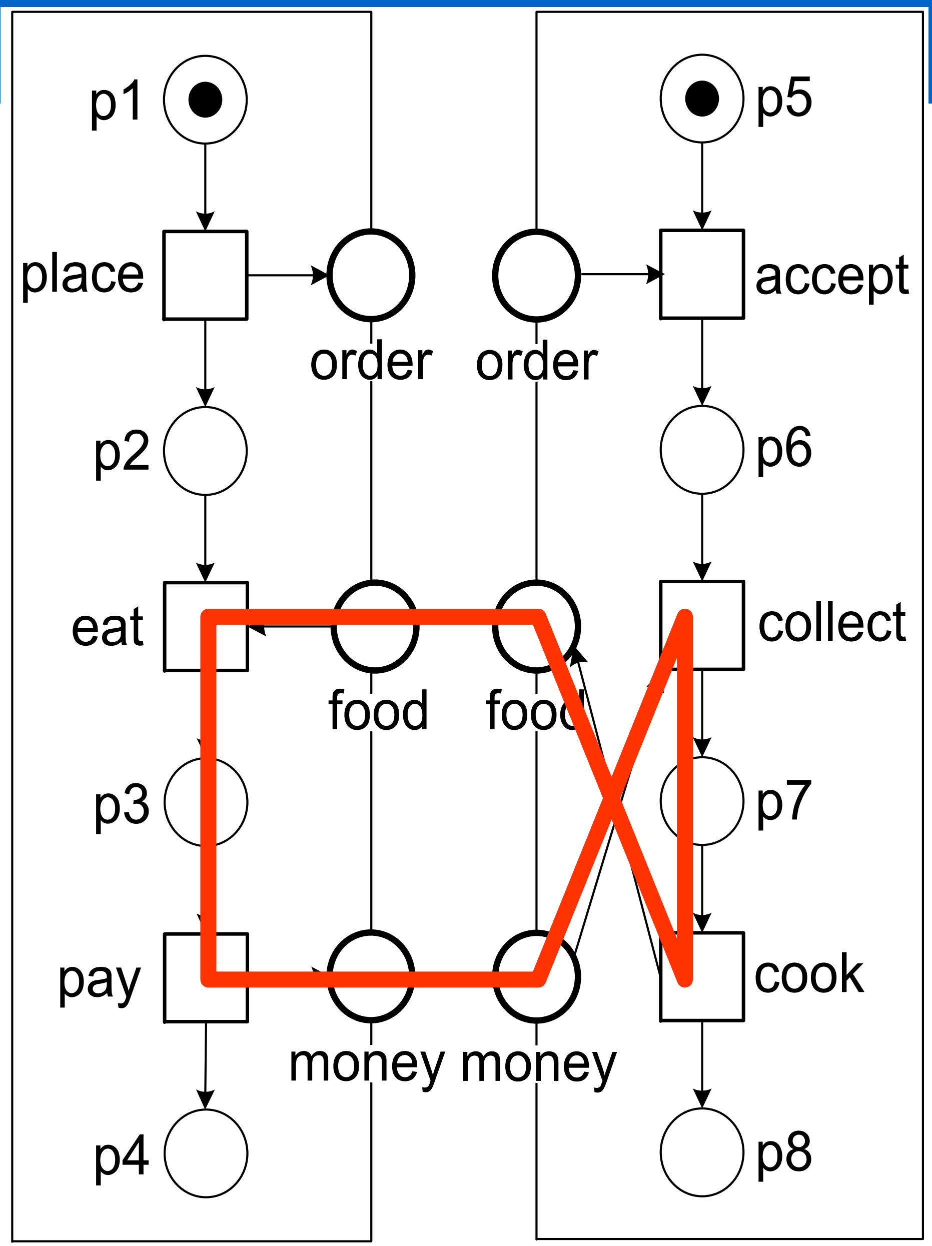
Some Terminology



Important assumption: asynchronous communication.

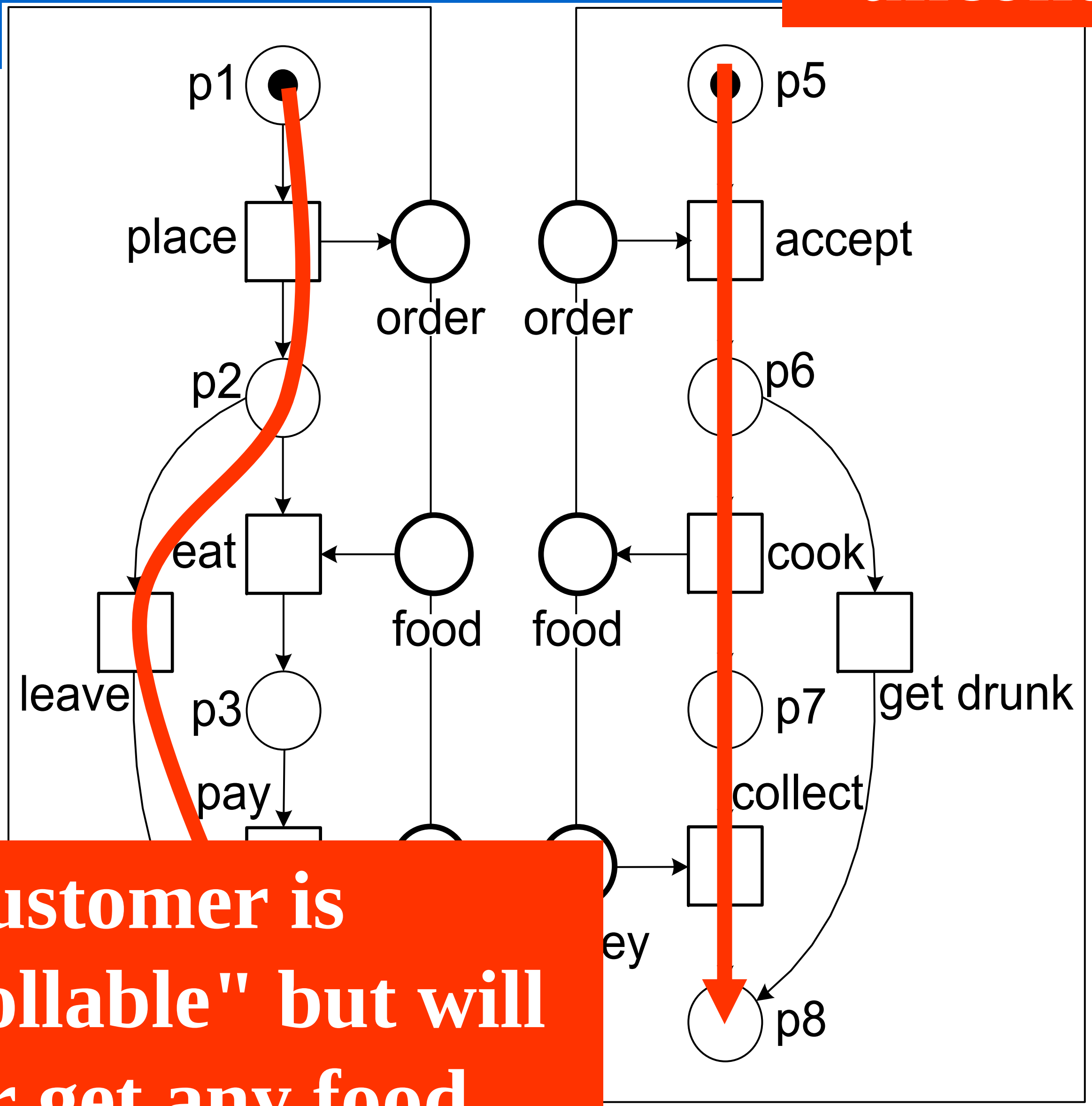
Interaction is a source of errors!


p1
p5
place
accept
order
order
p2
p6
eat
cook
food
food
p3
p7
pay
collect
money
money
p4
p8


p1
place
order
p2
eat
food
p3
pay
money
p4
p5
accept
order
p6
collect
food
p7
cook
money
p8

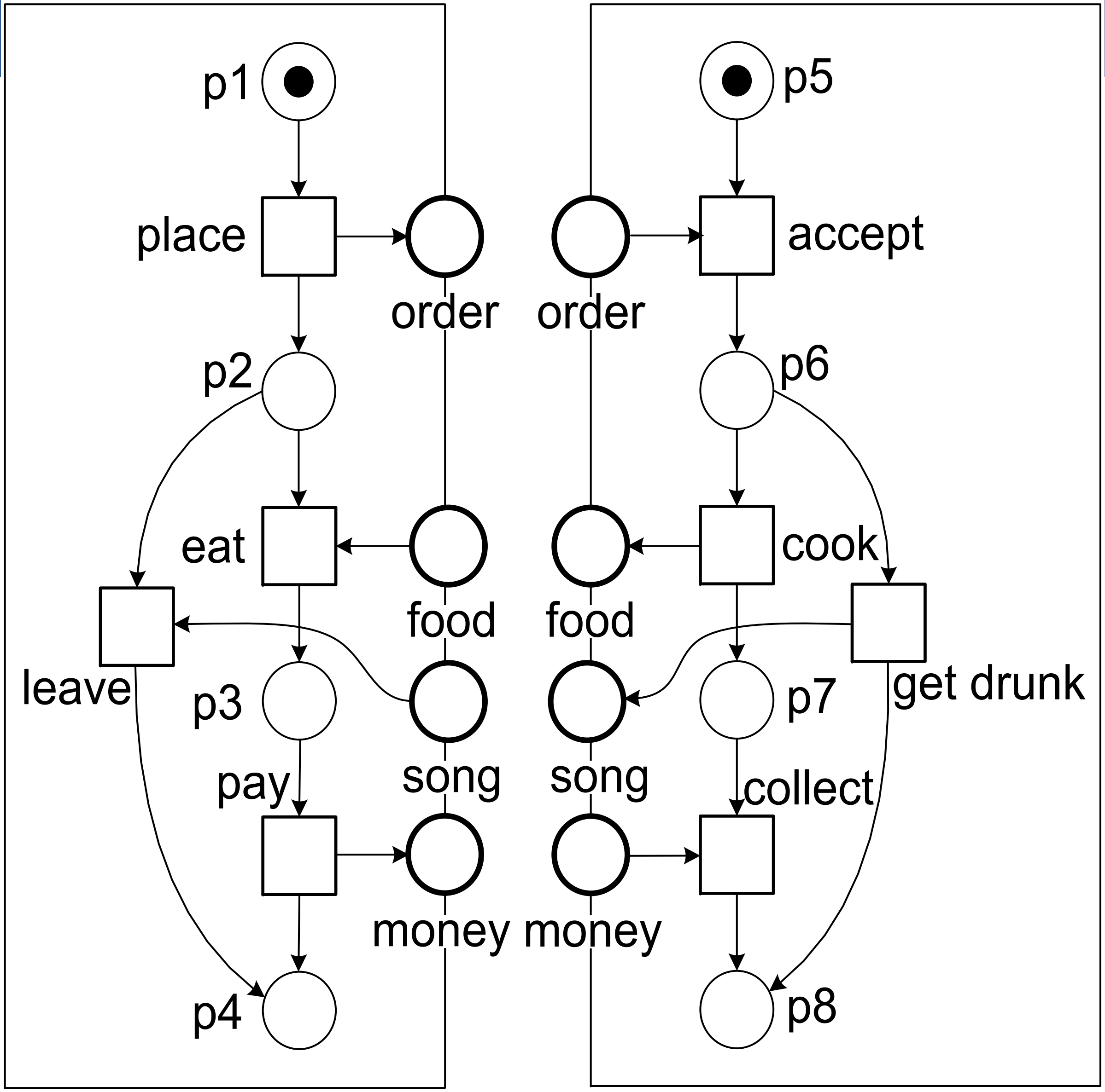
deadlock

restaurant is "uncontrollable"*



customer is "controllable" but will never get any food

*by any service with only dead final markings


p1
place
order
p2
eat
food
leave
p3
song
pay
money
p4
p5
accept
order
p6
cook
food
get drunk
p7
song
collect
money
p8

Workflow and Service Interaction Patterns

TU/e Technische Universiteit **Eindhoven** University of Technology

Where innovation starts


MS Workflow Foundation
Global 360 BPM Suite
YAWL
FileNet
InConcert
Fujitsu Interstage
Axxerion
BWise
Software AG/webMethods
XPDL
IBM WebSphere
casewise
COSA
BPEL
UML
Savvion BusinessManager
ADs
BPM|one
TIBCO iProcess Suite
jBPM
EPCs
FlowConnect
SAP Workflow
BPMN
Pegasystems SmartBPM Suite
Ensemble
Bizagi
TeamWARE
Oracle BPEL
Promatis
BiZZdesigner
Ultimus BPM Suite

Workflow Patterns Initiative

- Started in 1999, joint work TU/e and QUT
- Objectives:
  - Identification of workflow modeling scenarios and solutions
  - Benchmarking
    - Workflow products (MQ/Series Workflow, Staffware, etc)
    - Proposed standards for web service composition (BPML, BPEL)
    - Process modeling languages (UML, BPMN)
  - Foundation for selecting workflow solutions
- Home Page: www.workflowpatterns.com
- Primary publication:
  - W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, A.P. Barros, "Workflow Patterns", *Distributed and Parallel Databases* 14(3):5-51, 2003.
- Evaluations of commercial offerings, research prototypes, proposed standards for web service composition, etc

Workflow Patterns Framework

time	2000 / 2003	Jun 2005	Oct 2005	Sep 2006
	Control-flow P:s 20	Resource P:s - 43	Data P:s - 40	revised Control-flow P:s 43
	W. van der Aalst **A. ter Hofstede** **B. Kiepuszewski** **A. Barros**	**N. Russell** **W. van der Aalst** **A. ter Hofstede** **D. Edmond**	**N. Russell** **A. ter Hofstede** **D. Edmond** **W. van der Aalst**	**N. Russell** **A. ter Hofstede** **W. van der Aalst** **N. Mulyar**
	The ordering of activities in a process	Resource definition & work distribution in a process	Data representation and handling in a process	- 23 new patterns - Formalised in CPN notation
	CoopIS'2000 DAPD'2003	CAiSE'2005	ER'2005	TR

Slides adopted from Nick Russell, et al.

Workflow Patterns Framework

time

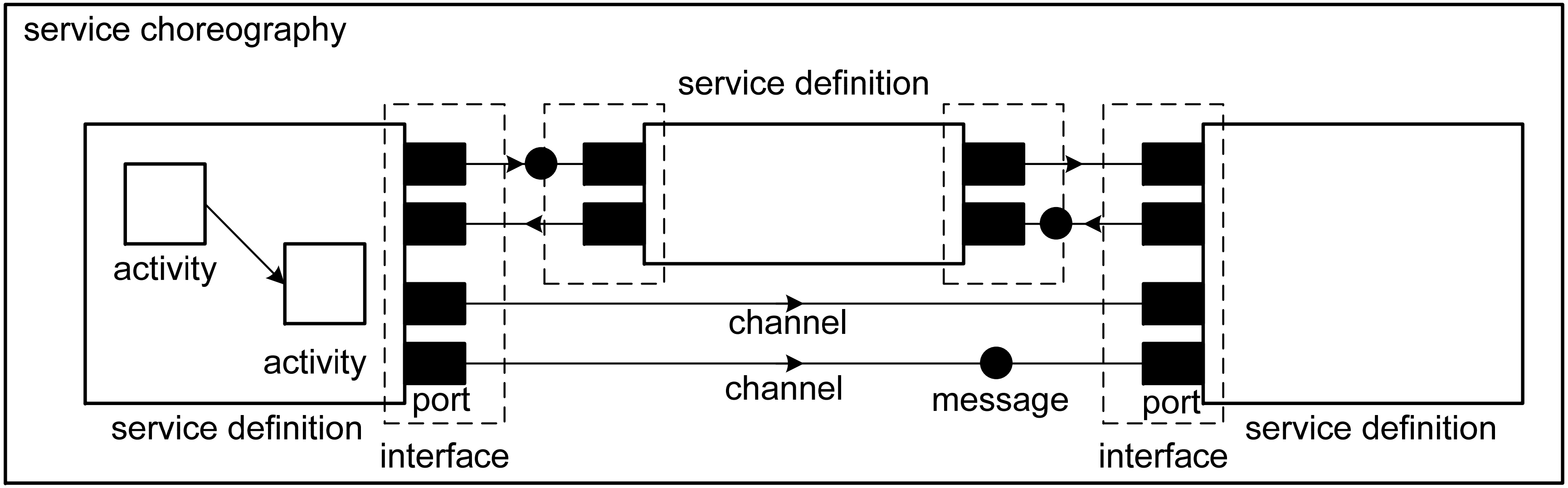
2000 — 2003 — Jun 2005 — Oct 2005

Control-flow P:s 20		Resource P:s - 43	Data P:s - 40	Ex
COSA FLOWer Eastman Meteor Mobile I-Flow Staffware InConcert	Domino Workflow Visual Workflow Forte Conductor MQSeries/Workflow SAR R/3 Workflow Verve Workflow Changengine	Staffware WebSphere MQ FLOWer COSA iPlanet	Staffware MQSeries FLOWer COSA	Sta We FLO CO iPla
XPDL, BPEL4WS, BPML, WSFL, XLANG, WSCI, UML AD 1.4 UML AD 2.0, BPMN		BPEL4WS UML AD 2.0 BPMN	XPDL, BPEL4WS UML AD 2.0, BPMN	XPD BPE

Evaluations

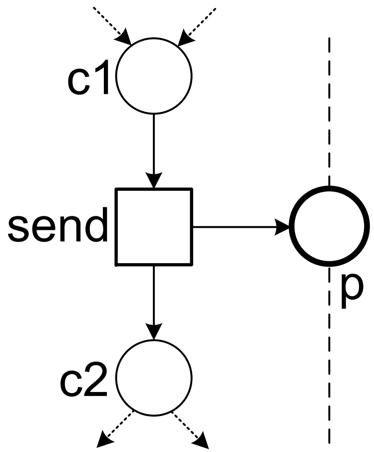
Language Development: YAWL/newYAWL

Service Interaction Patterns

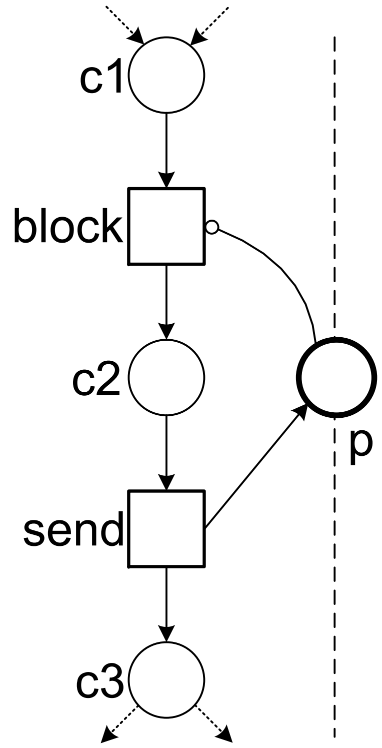


- **Basic Service Interaction Patterns (SIP-1, ... ,SIP-15)**
- **Correlation Patterns (SIP-16, ... ,SIP-23)**
- **Anti-Patterns (AP-1, AP-2, and AP-3)**

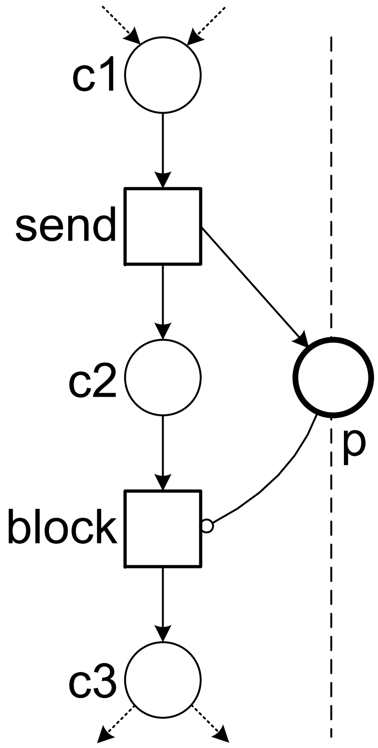
Send Patterns



SIP-1 Send pattern

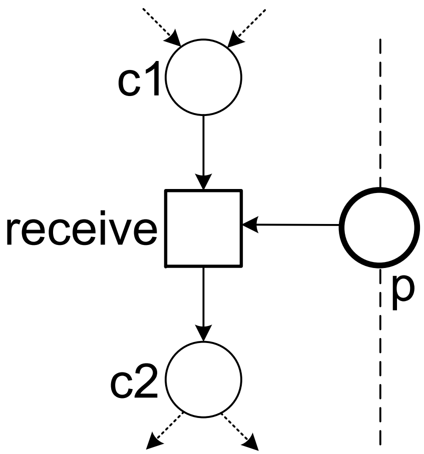


SIP-2 Pre-Blocking Send pattern

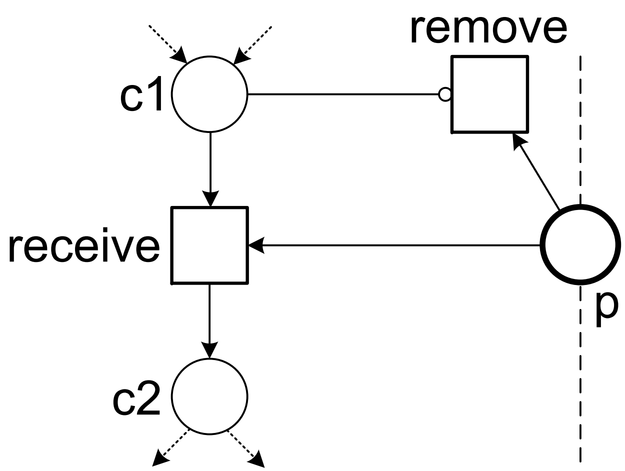


SIP-3 Post-Blocking Send pattern

Receive Patterns

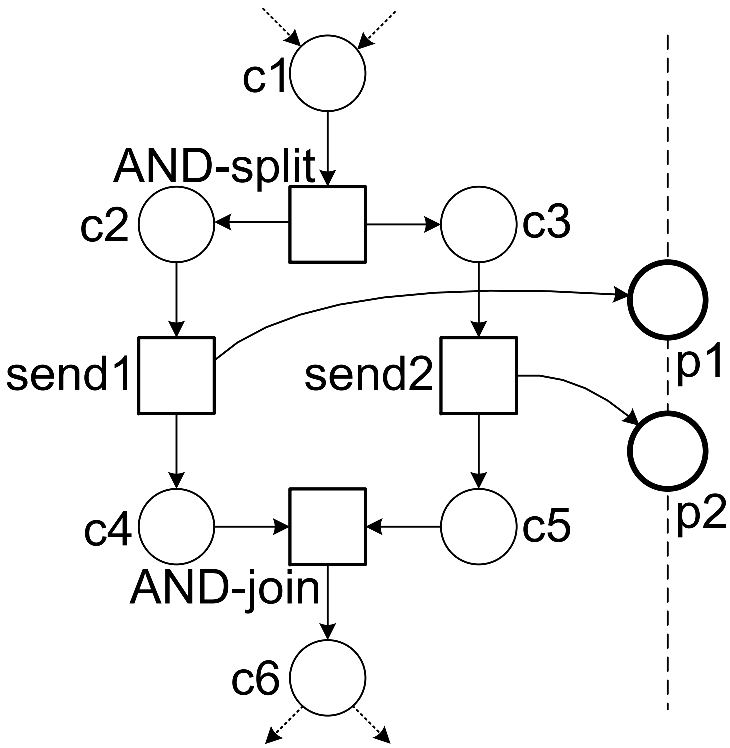


SIP-4 Receive pattern

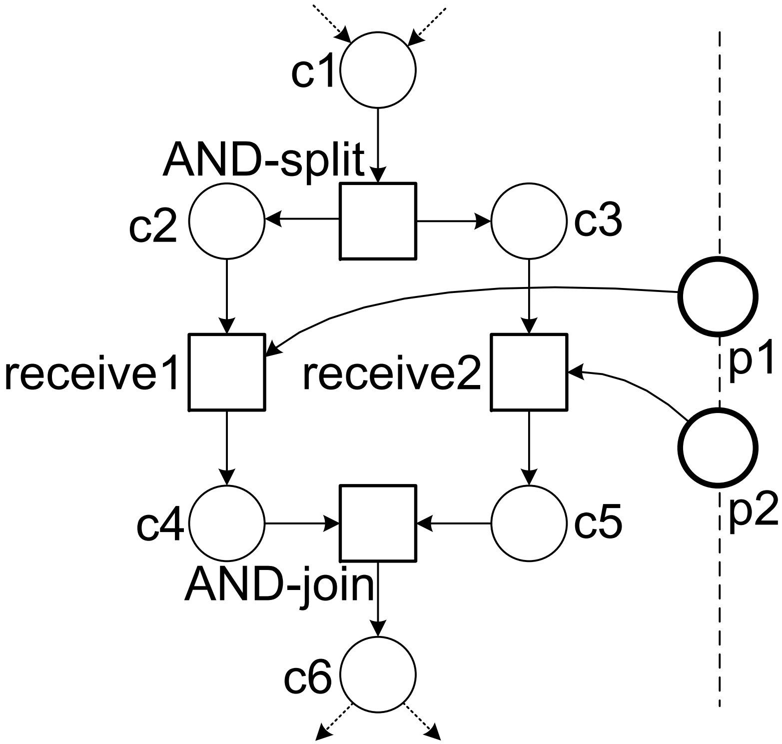


SIP-5 Lossy Receive pattern

Concurrent Send/Receive Patterns

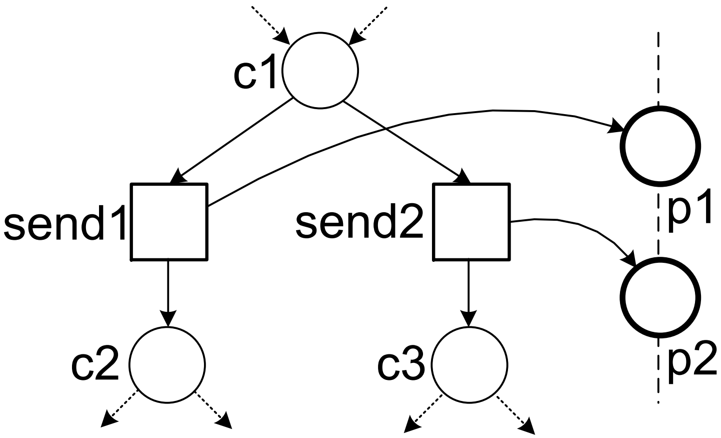


SIP-6 Concurrent Send pattern

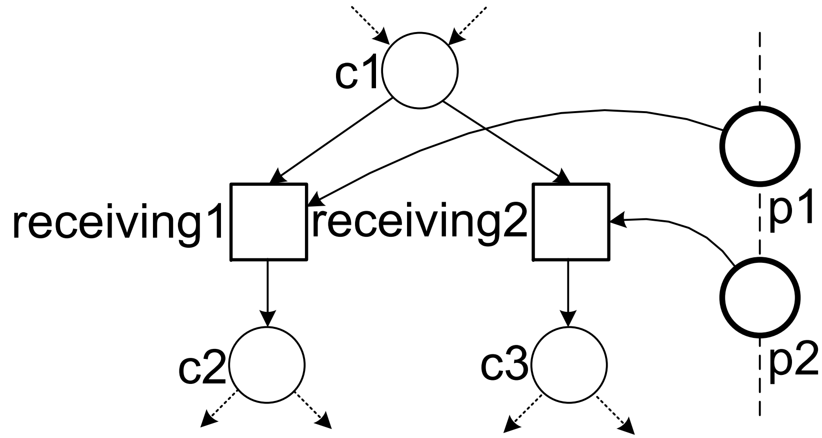


SIP-7 Concurrent Receive pattern

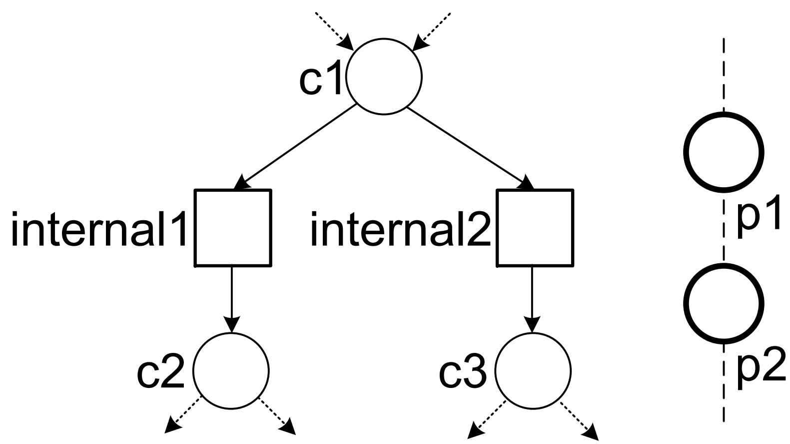
Choice Patterns



SIP-8 Sending Choice pattern

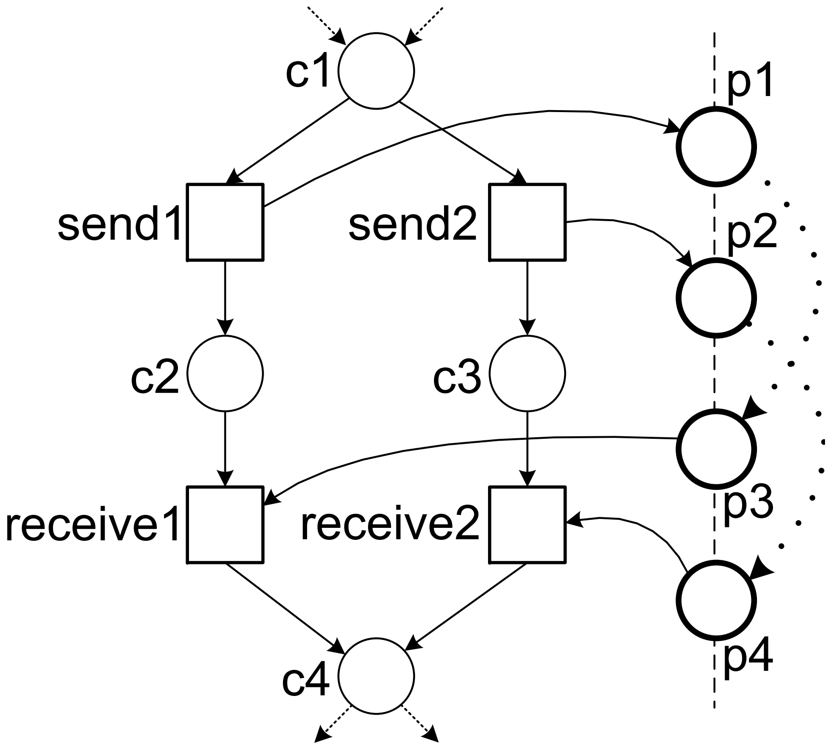


SIP-9 Receiving Choice pattern

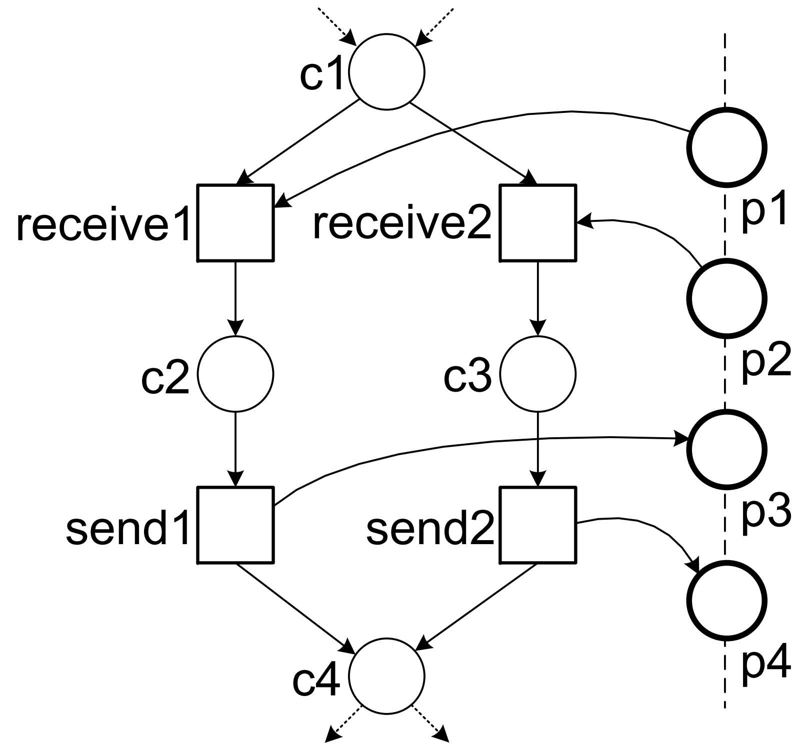


SIP-10 Internal Choice pattern

Choice With a Follow-Up Patterns (1/2)



SIP-11 Sending Choice Receiving Follow-Up pattern



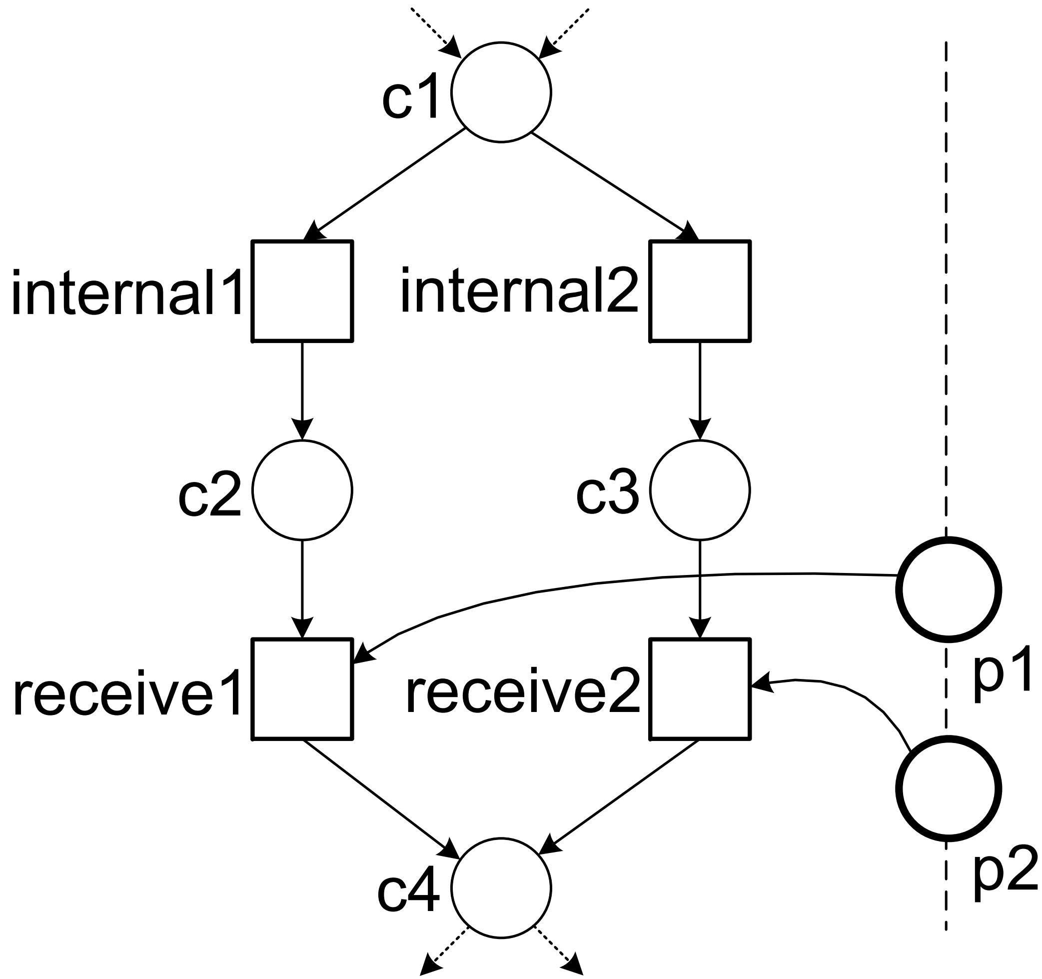
SIP-12 Receiving Choice Sending Follow-Up pattern

Choice With a Follow-Up Patterns (2/2)

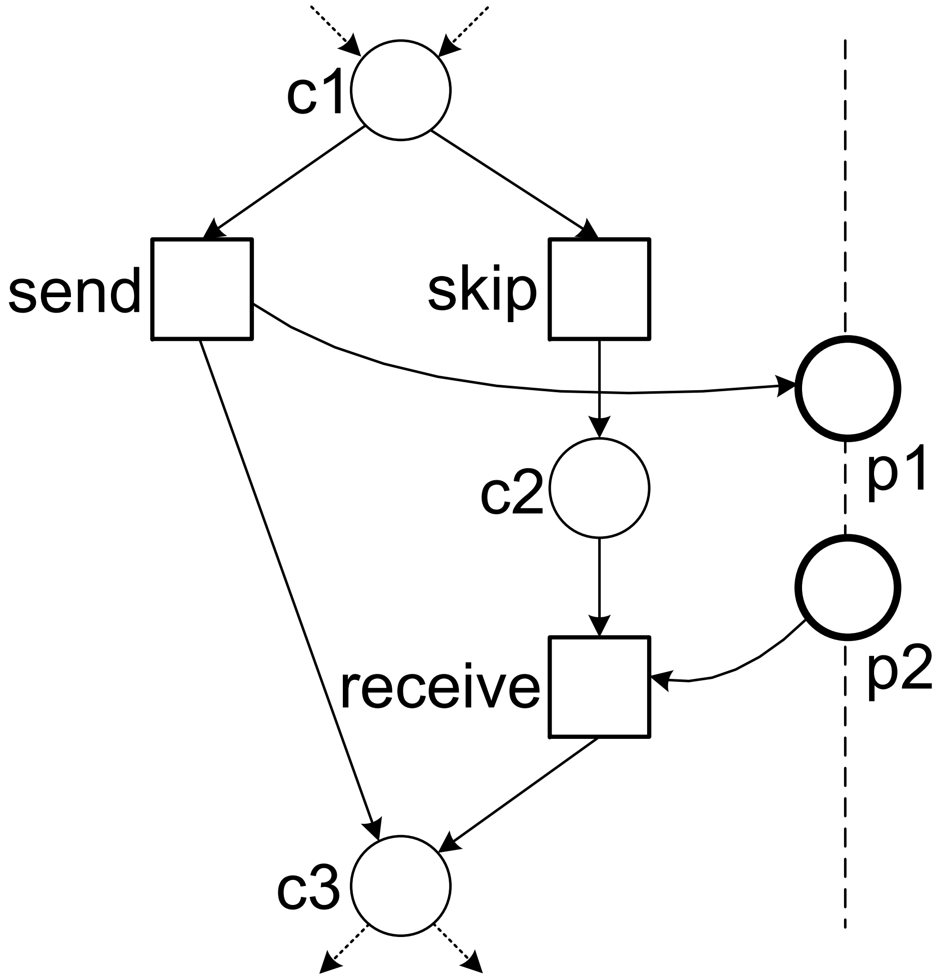
- SIP-13 Sending Choice Sending Follow-Up
- SIP-14 Receiving Choice Receiving Follow-Up, and
- SIP-15 Internal Choice Sending Follow-Up.

See problem?

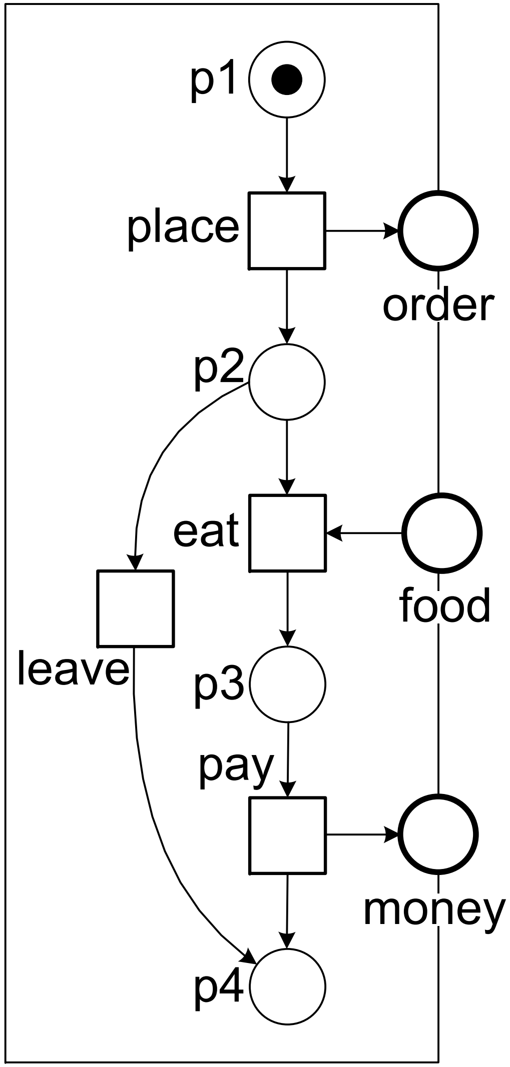
Anti-Pattern AP-1: Internal Choice Receiving Follow-Up Anti-Pattern

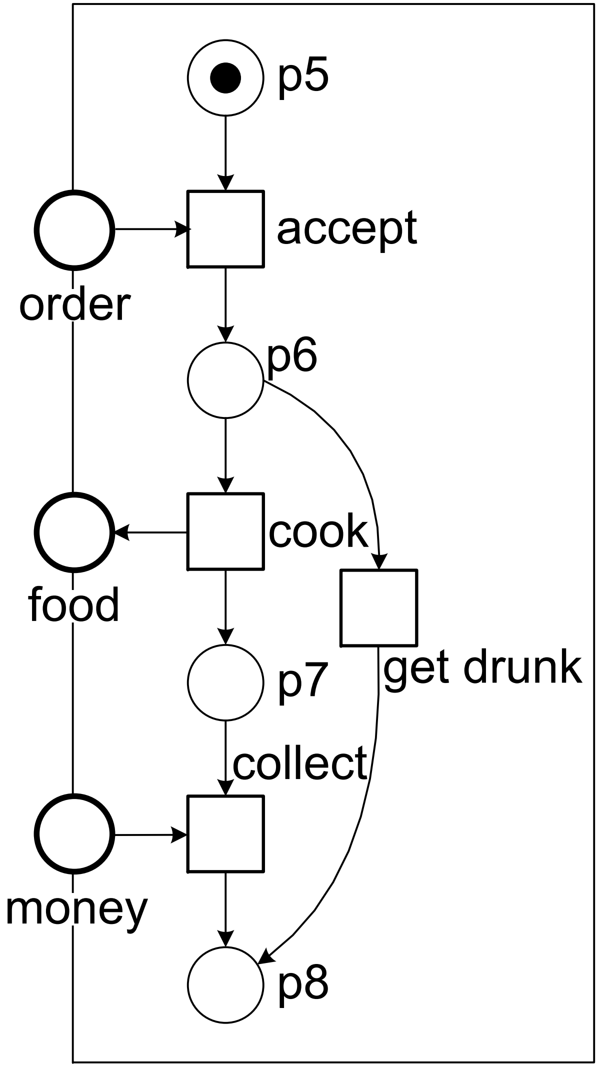


Another variant of AP-1

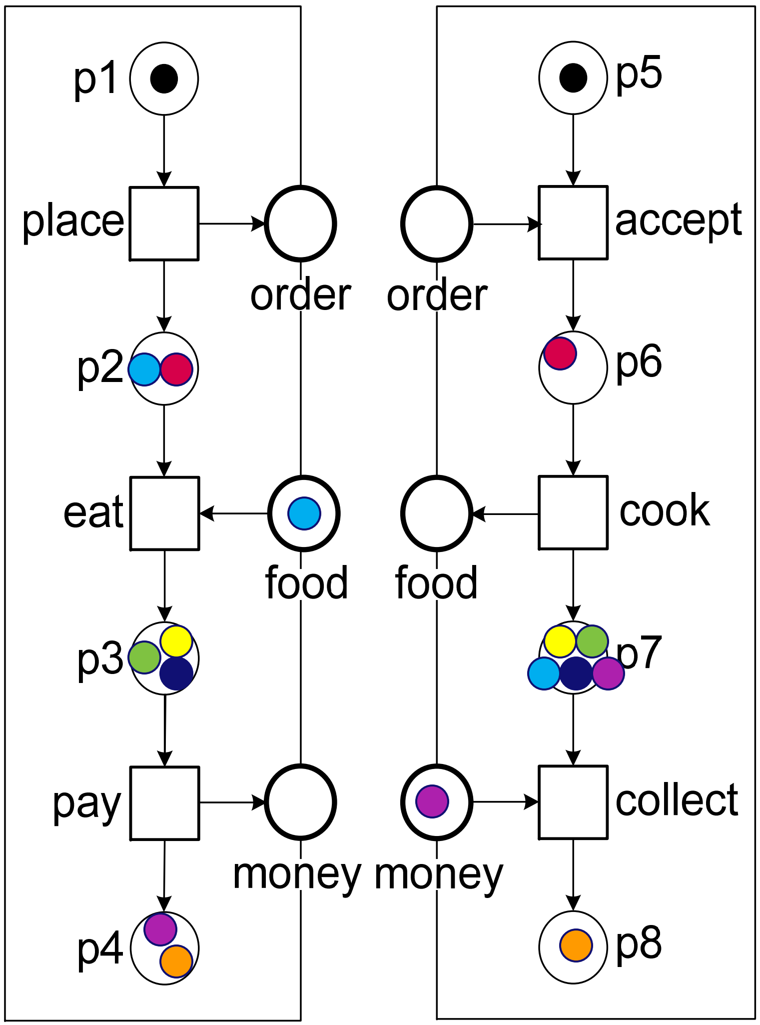


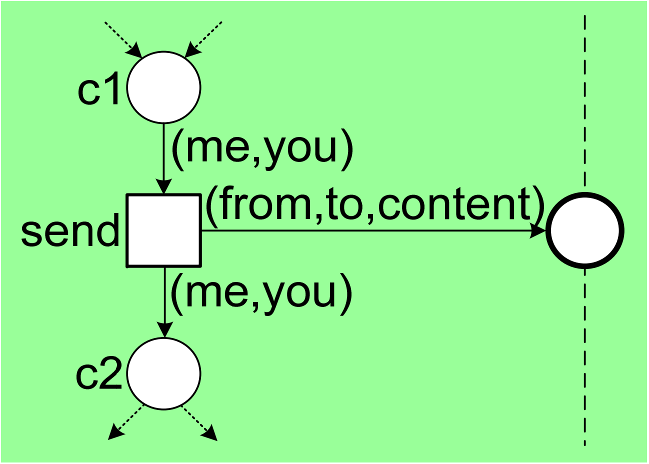
Two Additional Variants of AP-1

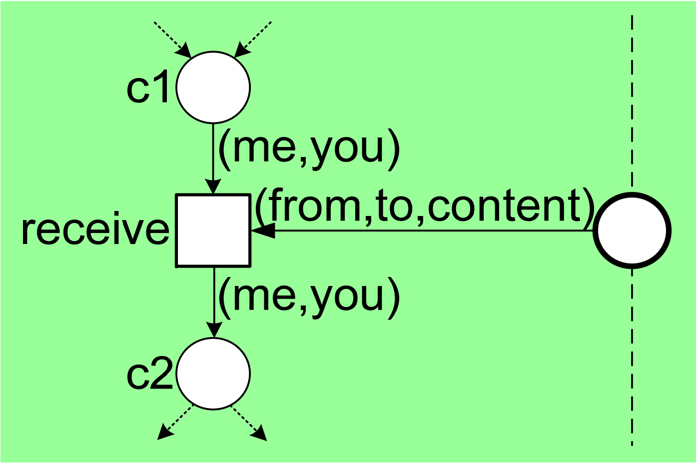




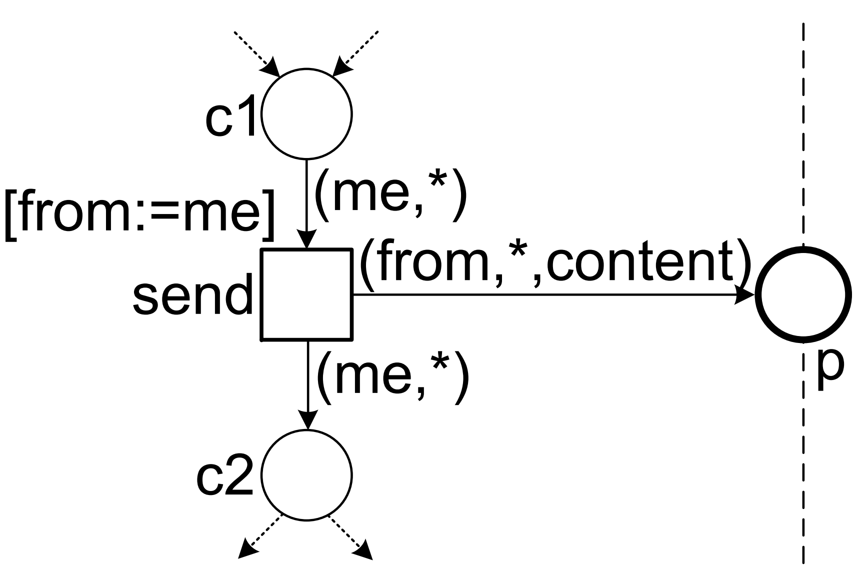
Correlation



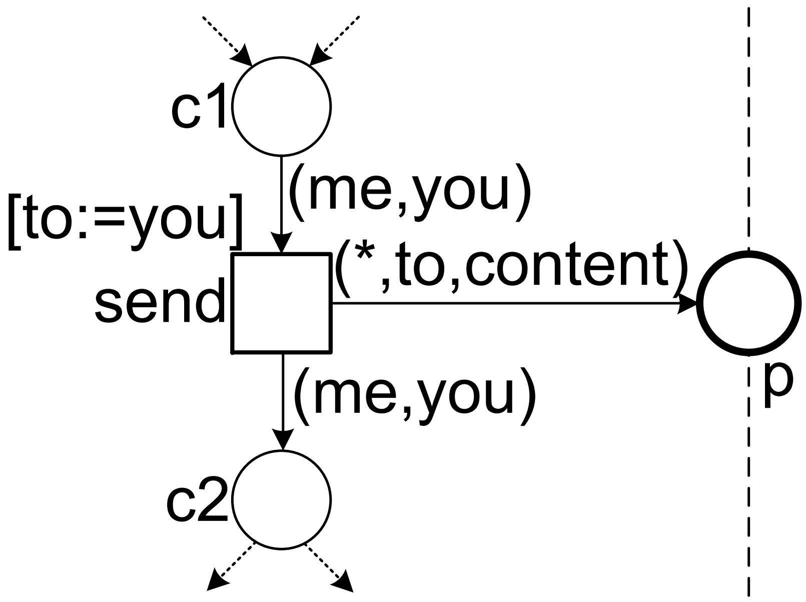




Correlation Send Patterns

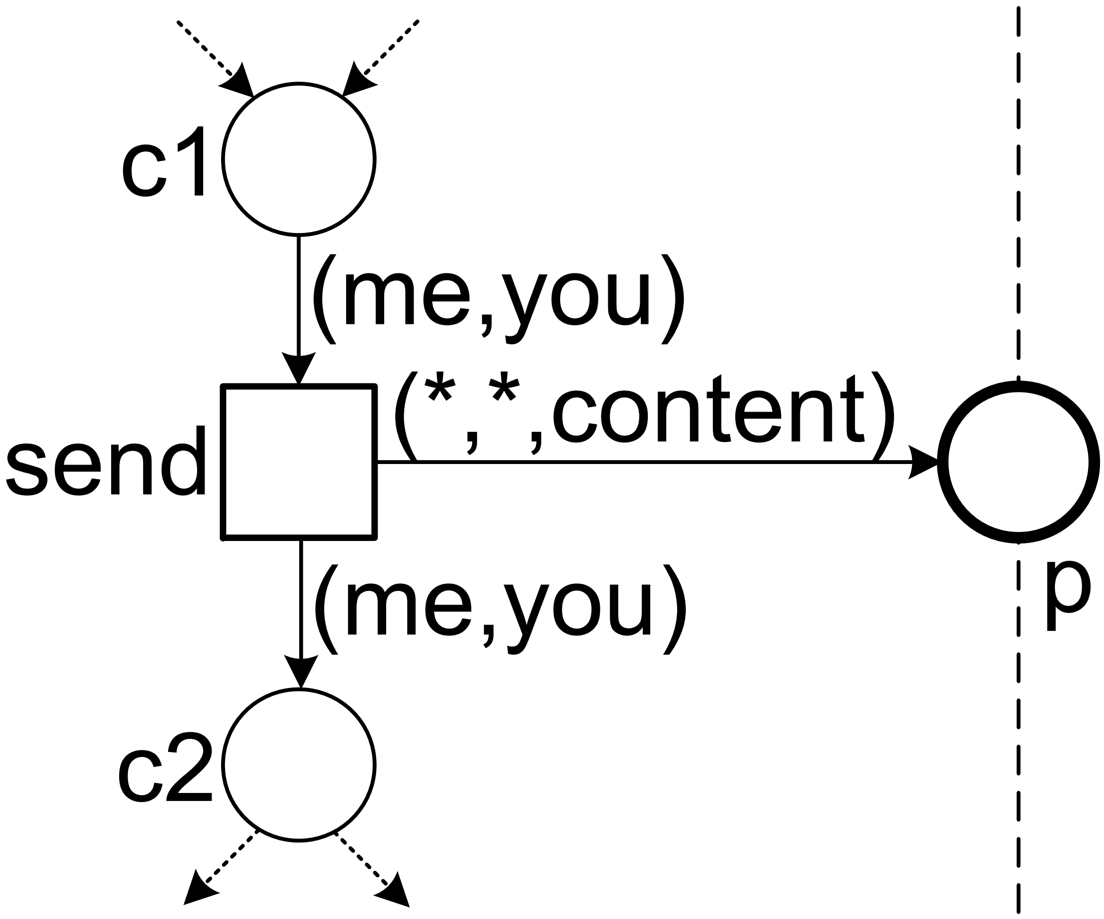


SIP-16 Leading Correlated Send pattern

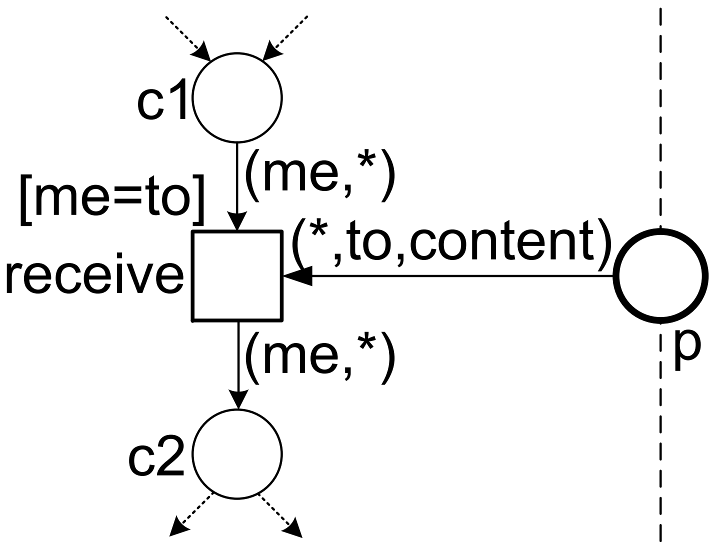


SIP-17 Following Correlated Send pattern

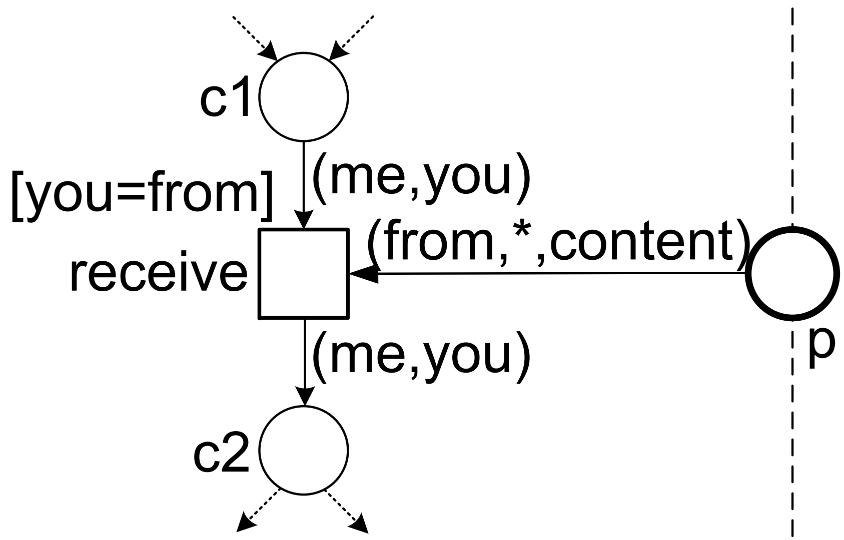
Anti-Pattern AP-2:
Uncorrelated Send Anti-Pattern



Correlation Receive Patterns (1/2)

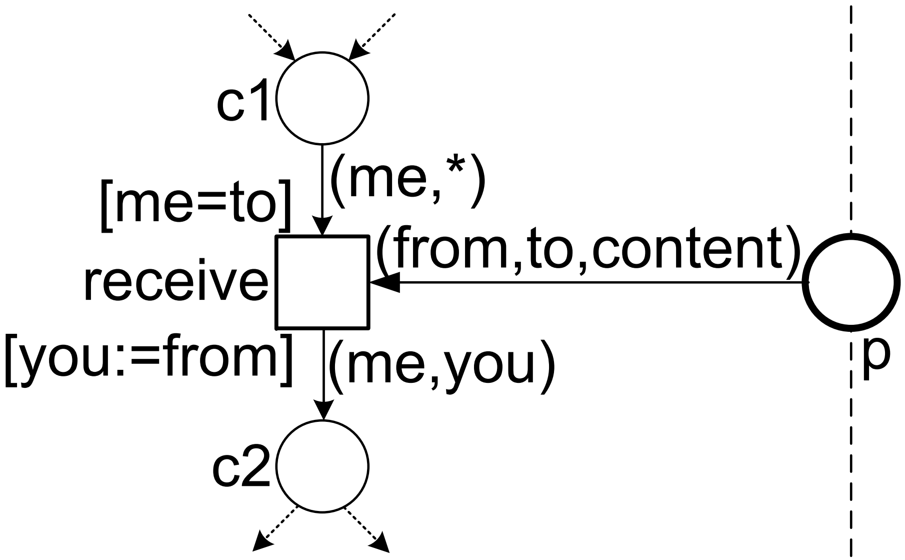


SIP-18 Leading Correlated Receive pattern

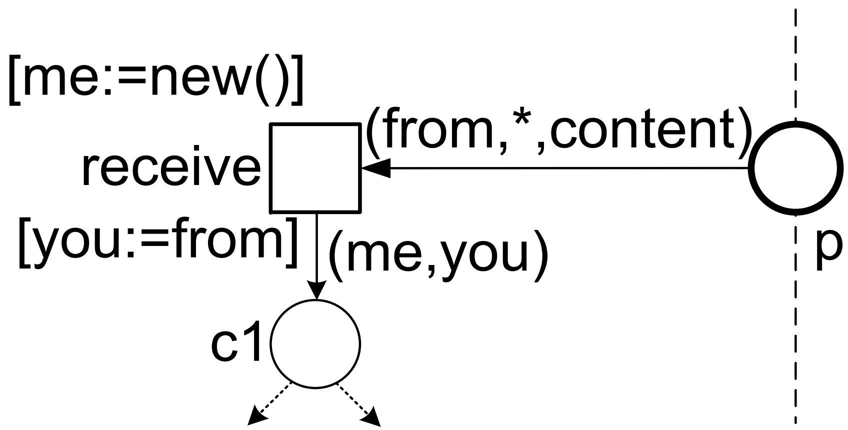


SIP-19 Following Correlated Receive pattern

Correlation Receive Patterns (2/2)

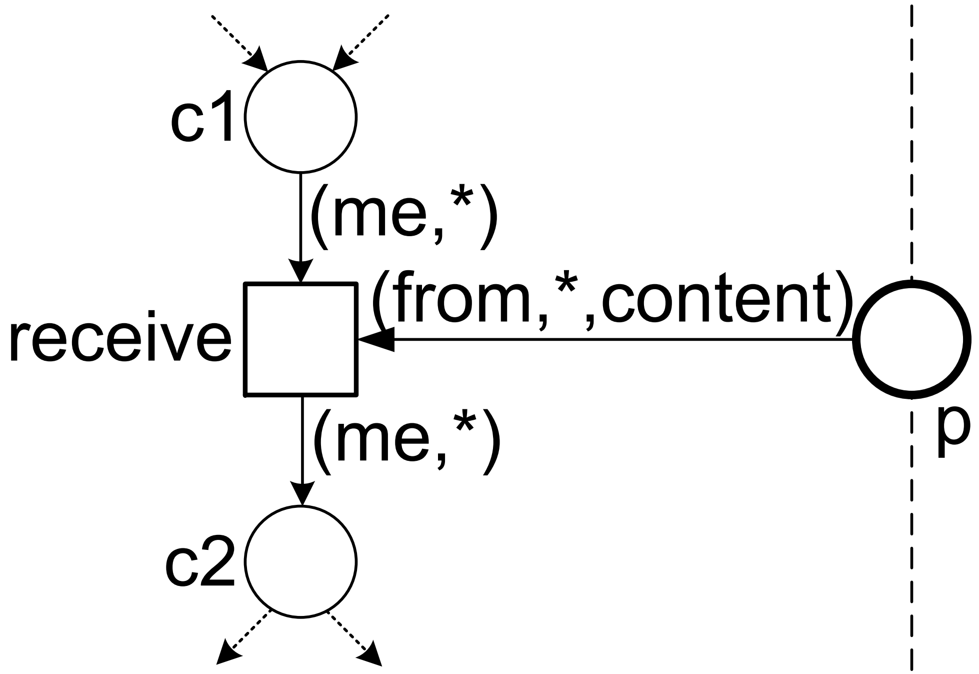


SIP-20 Learning Correlated Receive pattern



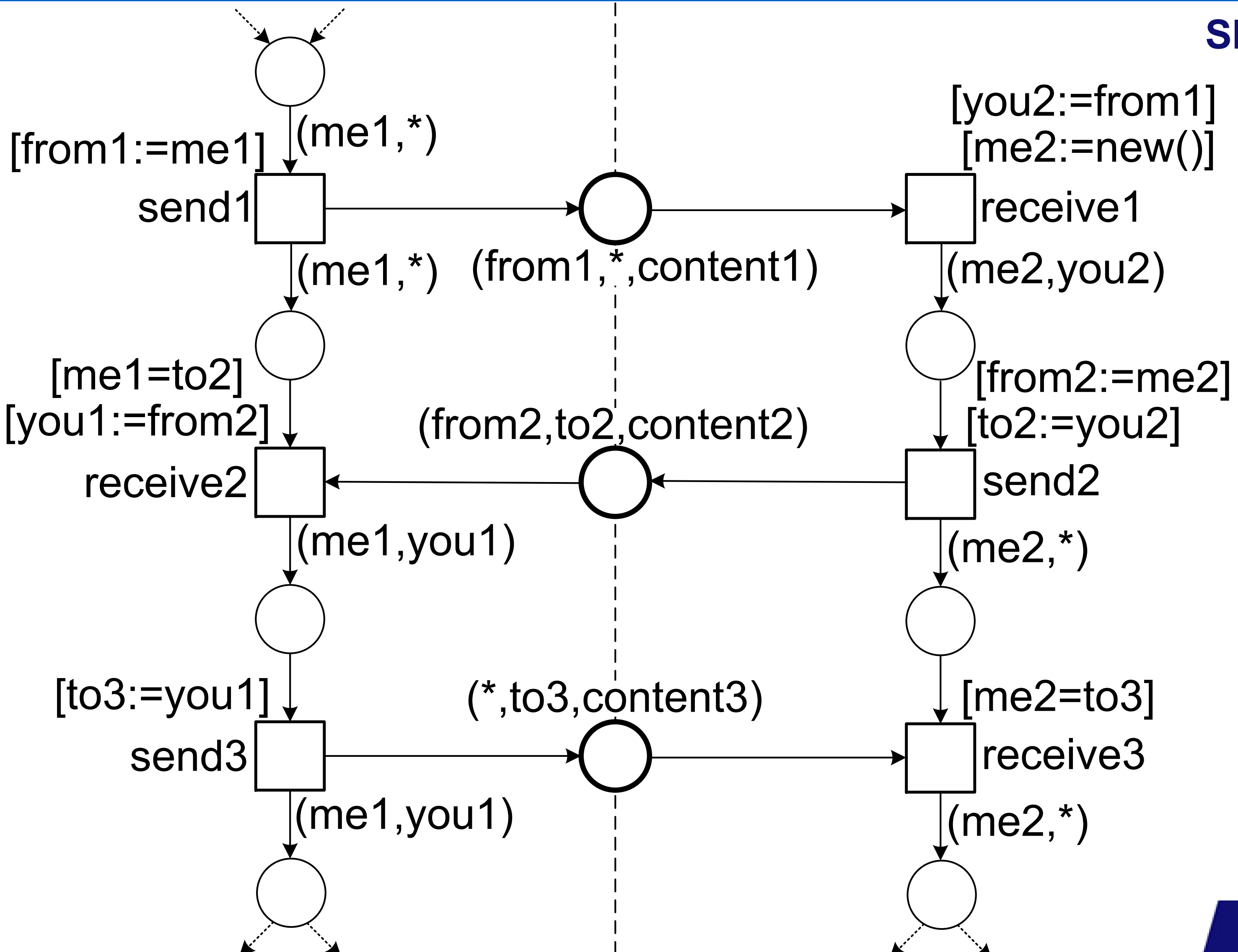
SIP-21 Creating Correlated Receive pattern

Anti-Pattern AP-3: Uncorrelated Receive Anti-Pattern

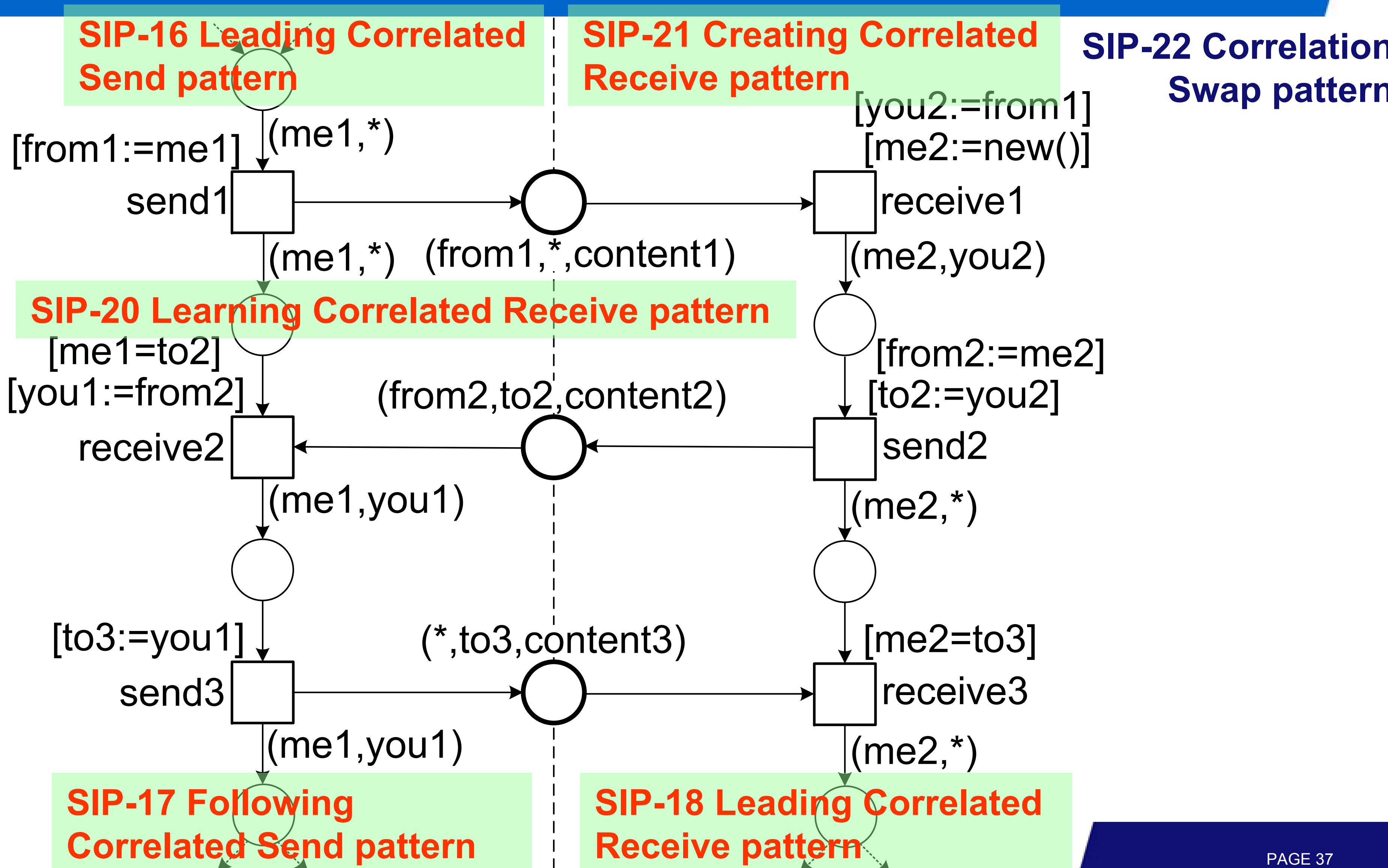


Composite Correlation Patterns (1/2)

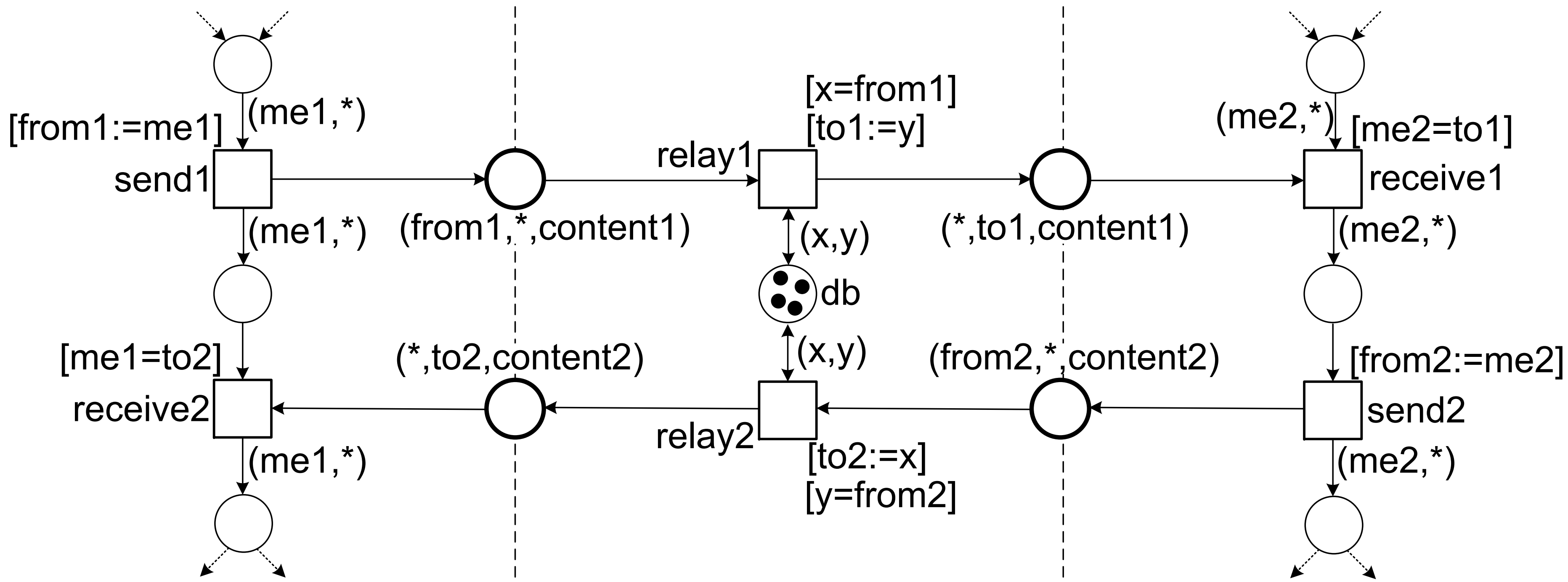
SIP-22 Correlation Swap pattern



Composite Correlation Patterns (1/2)

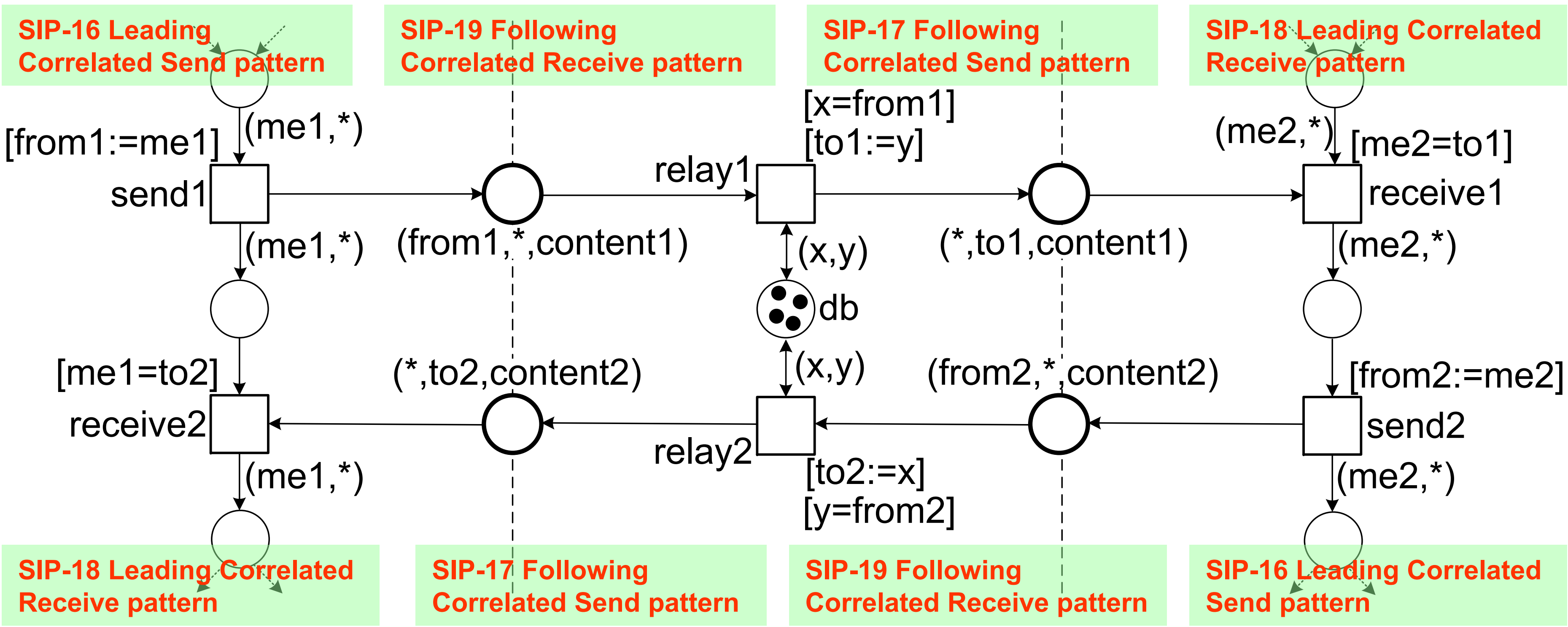


Composite Correlation Patterns (2/2)



SIP-23 Correlation Broker pattern

Composite Correlation Patterns (2/2)



SIP-23 Correlation Broker pattern

Recommended Reading (1/2)

- van der Aalst, W., Mooij, A.J., Stahl C., Wolf, K.. Service Interaction: Patterns, Formalization, and Analysis. In SFM 2009, volume 5569 of Lecture Notes in Computer Science, pages 42-88. Springer-Verlag, Berlin (2009)
- Barros, A., Dumas, M., ter Hofstede, A.: Service Interaction Patterns. In: Aalst, W., Benatallah, B., Casati, F. Curbera, F. (eds.) BPM 2005. LNCS, vol. 3649, pp. 302–318. Springer, Heidelberg (2005)
- Mulyar, N., Aldred, L., van der Aalst, W.: The Conceptualization of a Configurable Multiparty Multi-message Request-Reply Conversation. In: Meersman, R., Tari, Z. (eds.) OTM 2007, Part I. LNCS, vol. 4803, pp. 735–753. Springer, Heidelberg (2007)
- van der Aalst, W., ter Hofstede, A., Kiepuszewski, B., Barros, A.: Workflow Patterns. Distributed and Parallel Databases 14(1), 5–51 (2003)
- Russell, N., ter Hofstede, A., van der Aalst, W., Mulyar, N.: Workflow Control-Flow Patterns: A Revised View. BPM Center Report BPM-06-22, BPMcenter.org (2006)
- Mulyar, N.: Patterns for Process-Aware Information Systems: An Approach Based on Colored Petri Nets. Ph.D thesis, Eindhoven University of Technology, Eindhoven (2009)

Recommended Reading (2/2)

- Hohpe, G., Woolf, B.: Enterprise Integration Patterns. Addison-Wesley Professional, Reading (2003)
- Russell, N., van der Aalst, W., ter Hofstede, A., Edmond, D.: Workflow Resource Patterns: Identification, Representation and Tool Support. In: CAiSE 2005. LNCS, vol. 3520, pp. 216–232. Springer, Heidelberg (2005)
- Russell, N., ter Hofstede, A., Edmond, D., van der Aalst, W.: Workflow Data Patterns: Identification, Representation and Tool Support. In: ER 2005. LNCS, vol. 3716, pp. 353–368. Springer, Heidelberg (2005)
- Russell, N., van der Aalst, W., ter Hofstede, A.: Workflow Exception Patterns. In: Dubois, E., Pohl, K. (eds.) CAiSE 2006. LNCS, vol. 4001, pp. 288–302. Springer, Heidelberg (2006)
- Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Professional Computing Series. Addison Wesley, Reading (1995)
- Alexander, C.: A Pattern Language: Towns, Building and Construction. Oxford University Press, Oxford (1977)

Challenging Analysis Questions

Technische Universiteit
Eindhoven
University of Technology

Where innovation starts

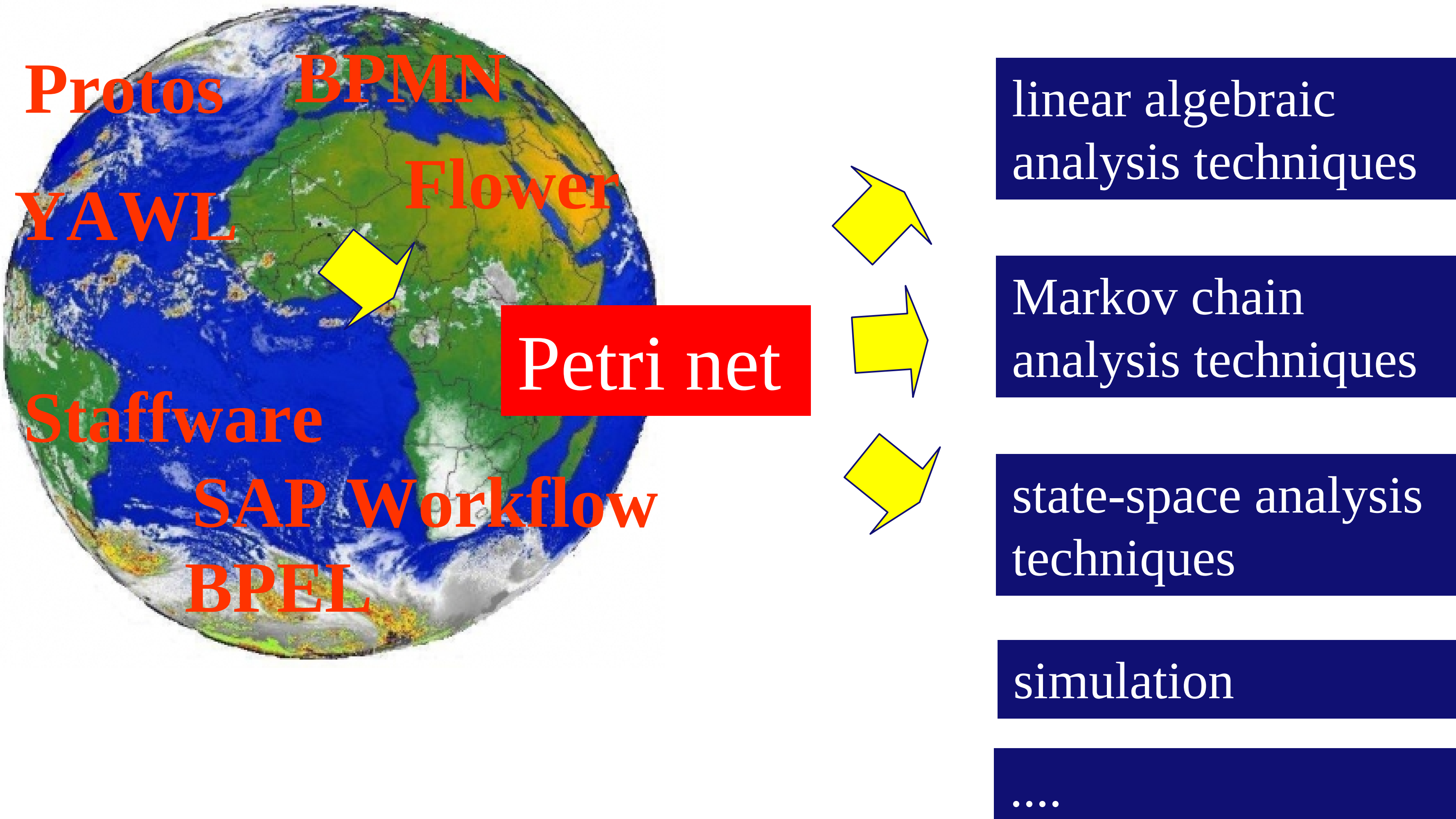
Questions Addressed in this Tutorial

1. Exposing Services
   - How to inform others about me such that cooperation is possible?
   - Two approaches: (a) expose own behavior and (b) provide operating guideline.
2. Replacing and Refining Services
   - How to replace or refine a service without introducing problems?
   - Inheritance, accordance, transformation rules, etc.
3. Integrating Services Using Adapters
   - How to resolve behavioral incompatibilities?
   - Adapter generation.
4. Service Mining
   - How to analyze the run-time behavior?

Additional Questions

- Verification (e.g., various types of soundness)
- Controllability (Is there a compatible partner?)
- Instance migration (Can I replace a service at run-time?)
- Querying software repositories (Is there a service that ... ?)
- Similarity of services (What is the least incompatible service? How many edit steps are needed to transform one into the other?)
- How to generate/compose services to meet specific requirements and goals?

Design-time analysis of processes



From BPEL to Petri Nets and Back

- Feature complete mappings from BPEL to Petri nets:
  - WofBPEL (TU/e & QUT)
  - BPEL2oWFN (Rostock & Humboldt)
- Mappings from Petri nets to BPEL:
  - WorkflowNet2BPEL4WS (TU/e & Aarhus)
  - oWFN2BPEL (Rostock & Humboldt)
- Similar results hold for the BPMN, EPCs, etc.!
- Be critical! Not all reported results exist :-)

Recommended Reading (1/3)

- van der Aalst, W., Mooij, A.J., Stahl C., Wolf, K.. Service Interaction: Patterns, Formalization, and Analysis. In SFM 2009, volume 5569 of Lecture Notes in Computer Science, pages 42-88. Springer-Verlag, Berlin (2009)
- van der Aalst, W., Lohmann, N., Massuthe, P., Stahl, C., Wolf, K.: From Public Views to Private Views: Correctness-by-Design for Services. In: Dumas, M., Heckel, H. (eds.) WS-FM 2007. LNCS, vol. 4937, pp. 139–153. Springer, Heidelberg (2008)
- Massuthe, P., Reisig, W., Schmidt, K.: An Operating Guideline Approach to the SOA. Annals of Mathematics, Computing & Teleinformatics 1(3), 35–43 (2005)
- Wolf, K.: Does my service have partners? In: ToPNoC II 2008. LNCS, vol. 5460, pp. 152–171. Springer, Heidelberg (2008)
- Massuthe, P., Reisig, W., Schmidt, K.: An Operating Guideline Approach to the SOA. In: Proceedings of the 2nd South-East European Workshop on Formal Methods 2005 (SEEFM 2005), Ohrid, Republic of Macedonia (2005)
- van der Aalst, W.M.P. et al.: Life After BPEL? In Formal Techniques for Computer Systems and Business Processes, LNCS, vol. 3670, pp. 35–50. Springer, Heidelberg (2005)
- www.service-technology.org

Recommended Reading (2/3)

- Lohmann, N., Massuthe, P., Wolf, K.: Operating guidelines for finite-state services. In: Kleijn, J., Yakovlev, A. (eds.) ICATPN 2007. LNCS, vol. 4546, pp. 321–341. Springer, Heidelberg (2007)
- Stahl, C., Massuthe, P., Bretschneider, J.: Deciding substitutability of services with operating guidelines. In: ToPNoC II 2008. LNCS, vol. 5460, pp. 172–191. Springer, Heidelberg (2008)
- Massuthe, P., Serebrenik, A., Sidorova, N., Wolf, K.: Can I find a partner? Undecidablity of partner existence for open nets. Information Processing Letters 108(6), 374–378 (2008)
- van der Aalst, W.M.P., et al.: Soundness of Workflow Nets with Reset Arcs is Undecidable! In J. Kleijn and M. Koutny, editors, Proceedings of the International Workshop on Concurrency Methods Issues and Applications (CHINA'08), pages 57-72 (2008)
- Trcka, N., van der Aalst, W.M.P., Sidorova, N.: Data-Flow Anti-Patterns: Discovering Data-Flow Errors in Workflows, CAiSE 2009, LNCS, 2009.
- Rozinat, A., van der Aalst, W.M.P.: Conformance Checking of Processes Based on Monitoring Real Behavior. Information Systems, 33(1):64-95, (2008)
- van der Aalst, W.M.P., Dumas, M., Ouyang, C., Rozinat, A., Verbeek, H.M.W.: Conformance Checking of Service Behavior. ACM Transactions on Internet Technology, 8(3):29-59 (2008)

Recommended Reading (3/3)

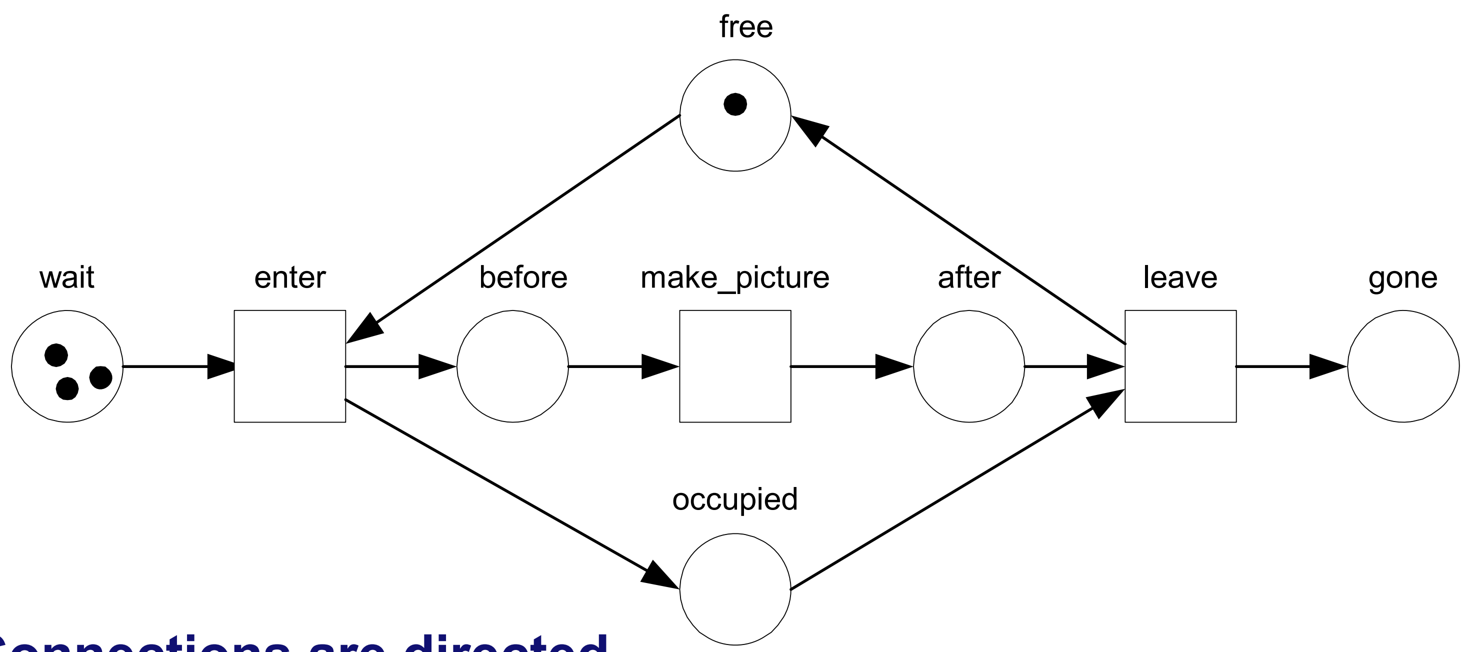
- Ouyang, C., van der Aalst, W.M.P., Breutel, S., Dumas, M., ter Hofstede, A.H.M., Verbeek, H.M.W.: Formal Semantics and Analysis of Control Flow in WS-BPEL. *Science of Computer Programming*, 67(2-3):162-198, 2007.
- C. Ouyang, E. Verbeek, W.M.P. van der Aalst, S. Breutel, M. Dumas, and A.H.M. ter Hofstede. WofBPEL: A Tool for Automated Analysis of BPEL Processes. In B. Benatallah, F. Casati, and P. Traverso, editors, *Proceedings of Service-Oriented Computing (ICSOC 2005)*, volume 3826 of *Lecture Notes in Computer Science*, pages 484-489. Springer-Verlag, Berlin, 2005.
- H.M.W. Verbeek and W.M.P. van der Aalst. Analyzing BPEL Processes using Petri Nets. In D. Marinescu, editor, *Proceedings of the Second International Workshop on Applications of Petri Nets to Coordination, Workflow and Business Process Management*, pages 59-78. Florida International University, Miami, Florida, USA, 2005.
- Lohmann, N., Massuthe, P., Stahl, C., Weinberg, D.: Analyzing interacting WSBPEL processes using flexible model generation. Data & Knowledge Engineering, 64(1), 38–54 (2008)
- van der Aalst, W.M.P., Lassen, K.B.: Translating Unstructured Workflow Processes to Readable BPEL: Theory and Implementation. *Information and Software Technology*, 50(3):131-159 (2008)
- C. Ouyang, M. Dumas, A.H.M. ter Hofstede, and W.M.P. van der Aalst. Pattern-Based Translation of BPMN Process Models to BPEL Web Services. *International Journal of Web Services Research*, 5(1):42-62 (2007)

A "Crash Course" in Petri Nets

TU/e Technische Universiteit Eindhoven University of Technology

Where innovation starts

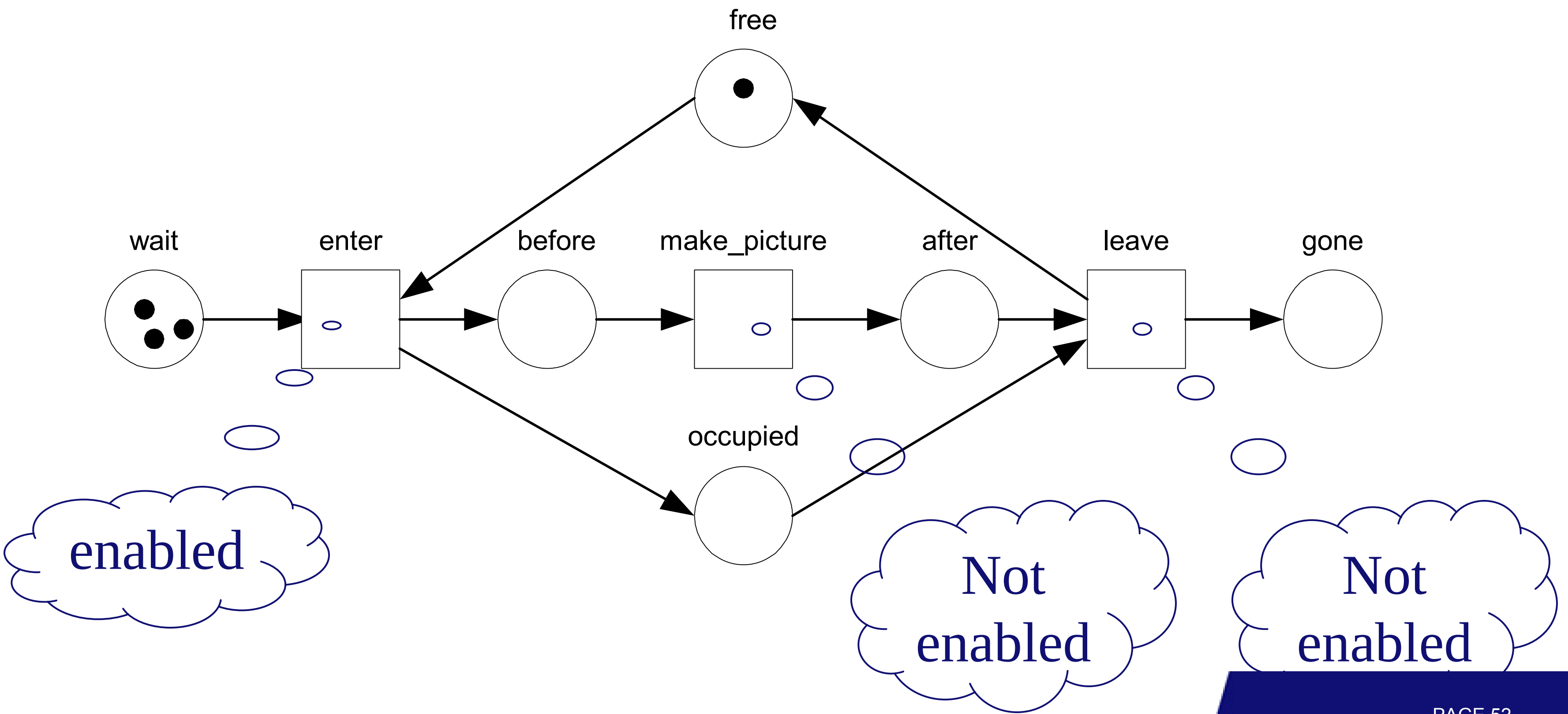
Rules



- **Connections are directed.**
- **No connections between two places or two transitions.**
- **Places may hold zero or more tokens.**
- **First, we consider the case of at most one arc between two nodes.**

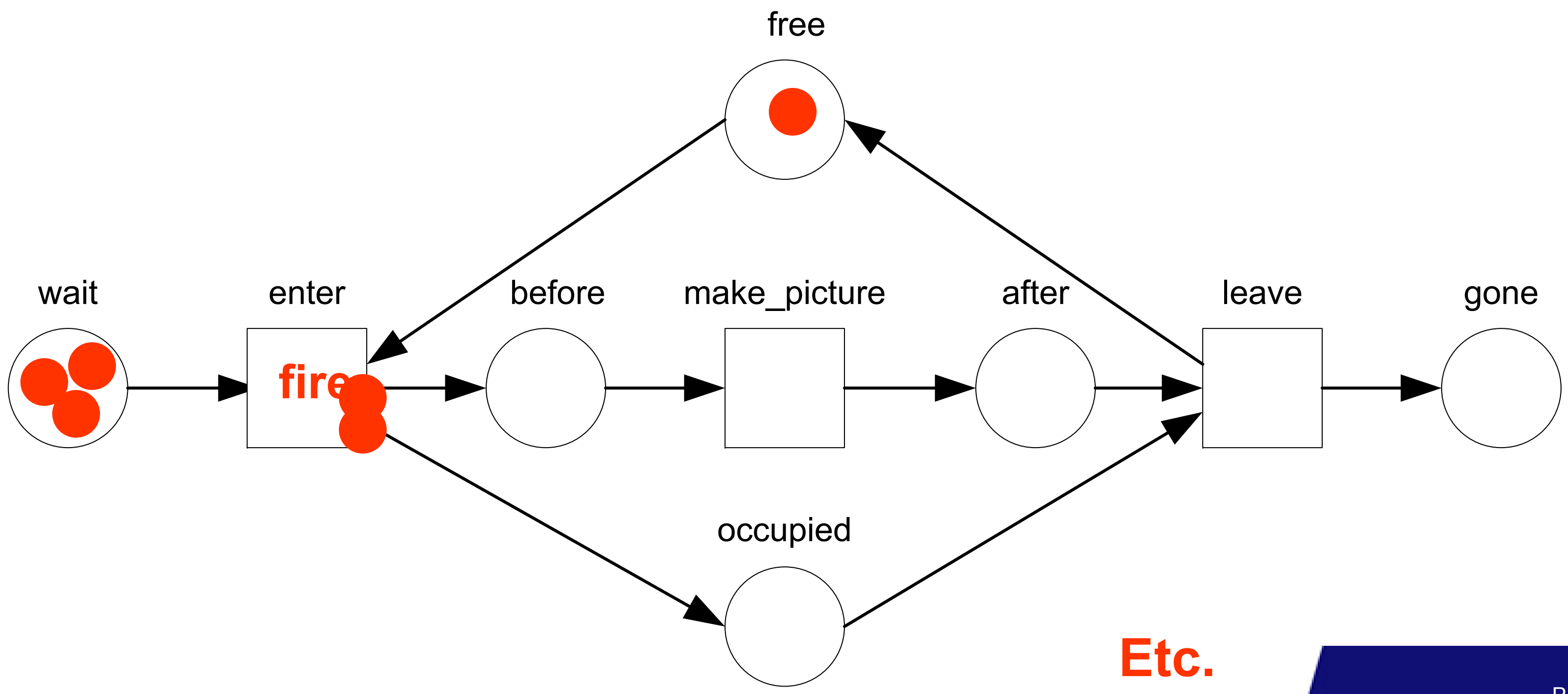
Enabled

- **A transition is enabled if each of its input places contains at least one token.**



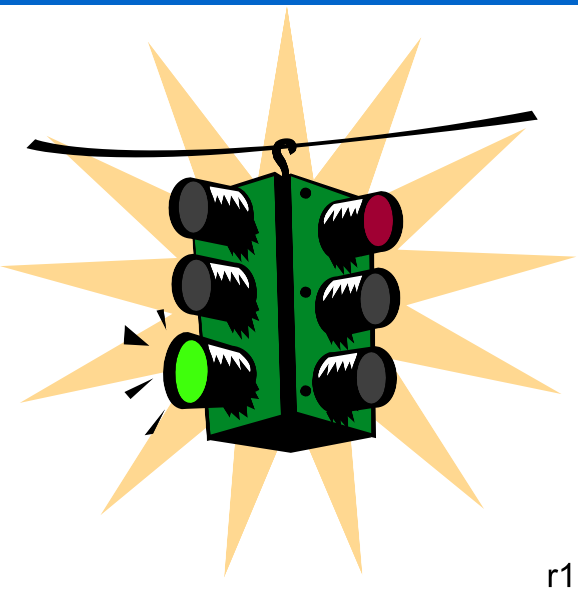
Firing

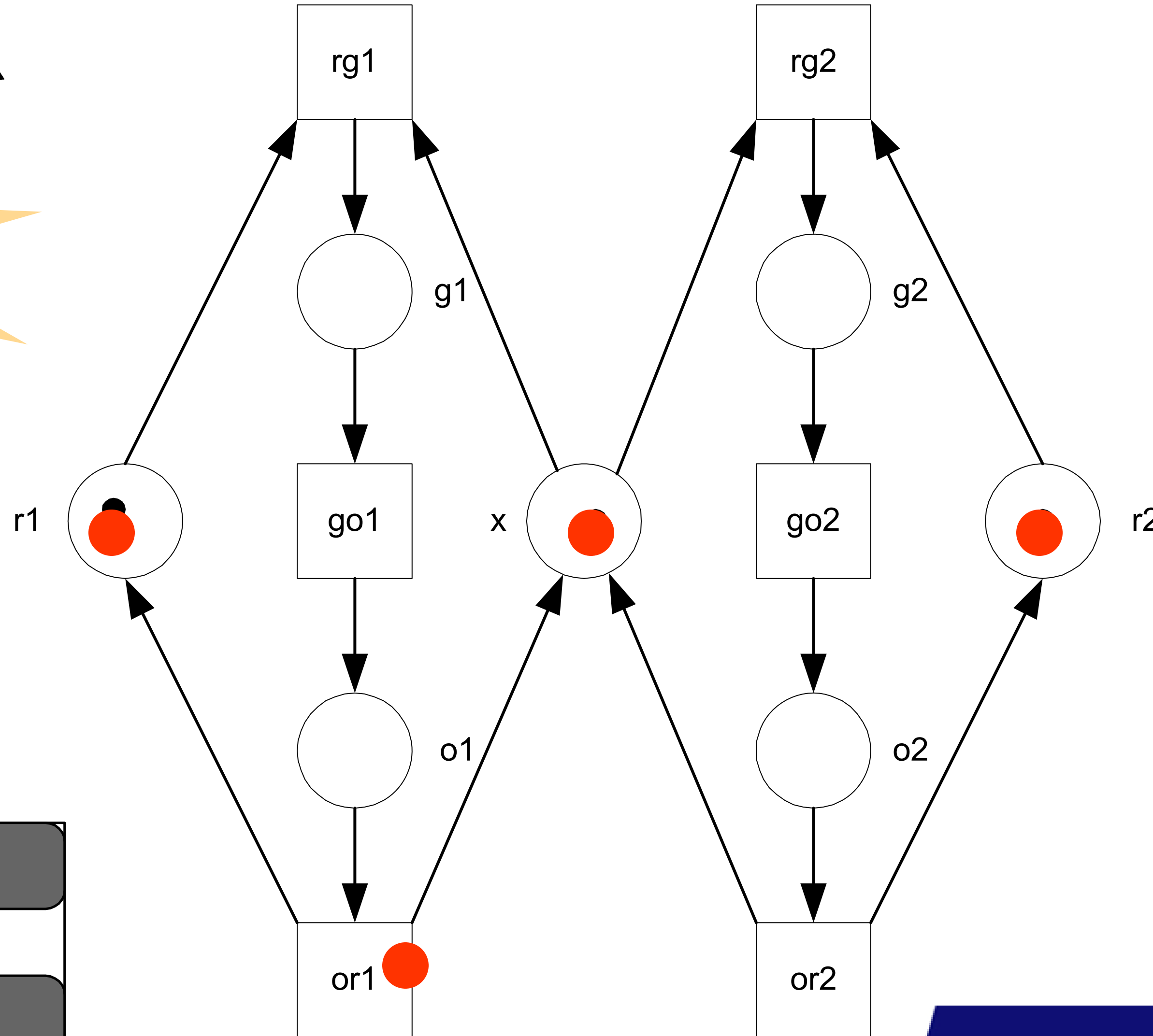
- **An enabled transition can fire (i.e., it occurs).**
- **When it fires it consumes a token from each input place and produces a token for each output place.**

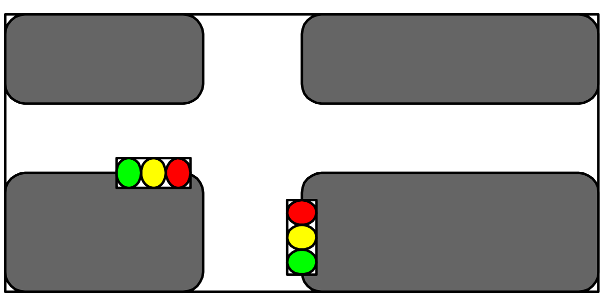


Etc.

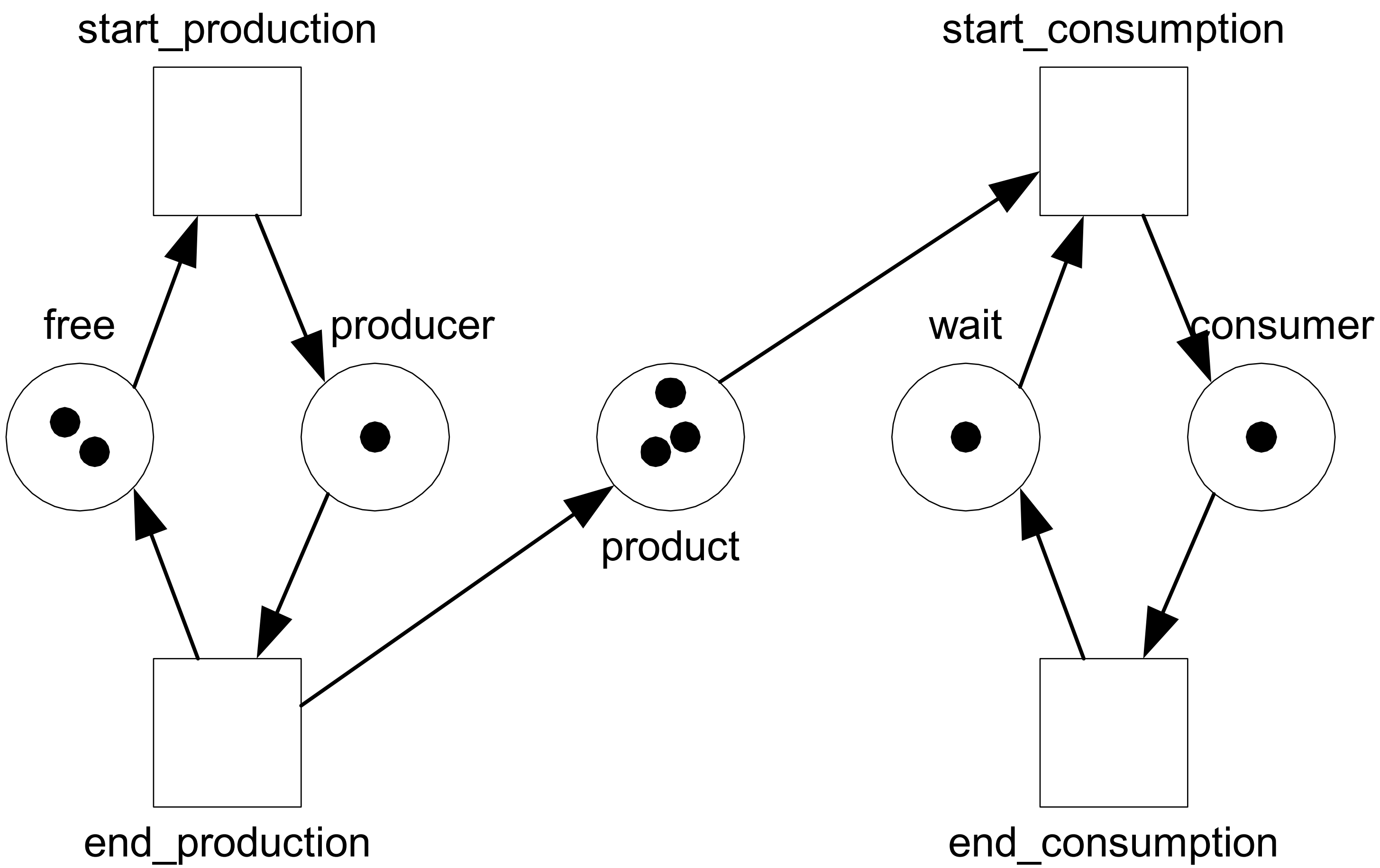
Example: Traffic Lights



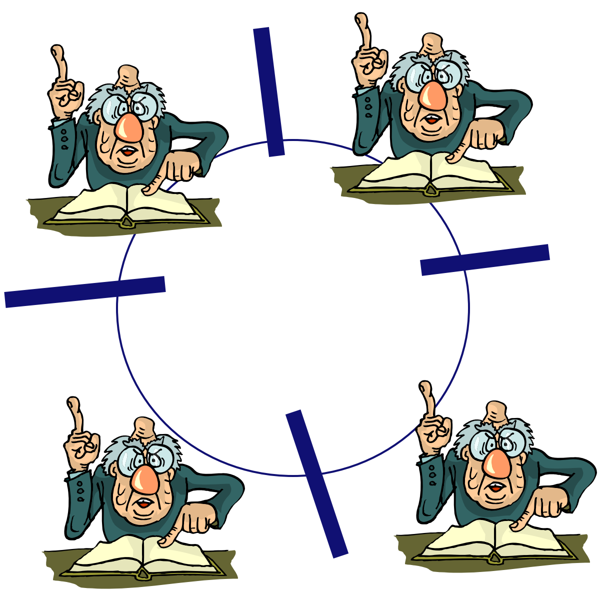


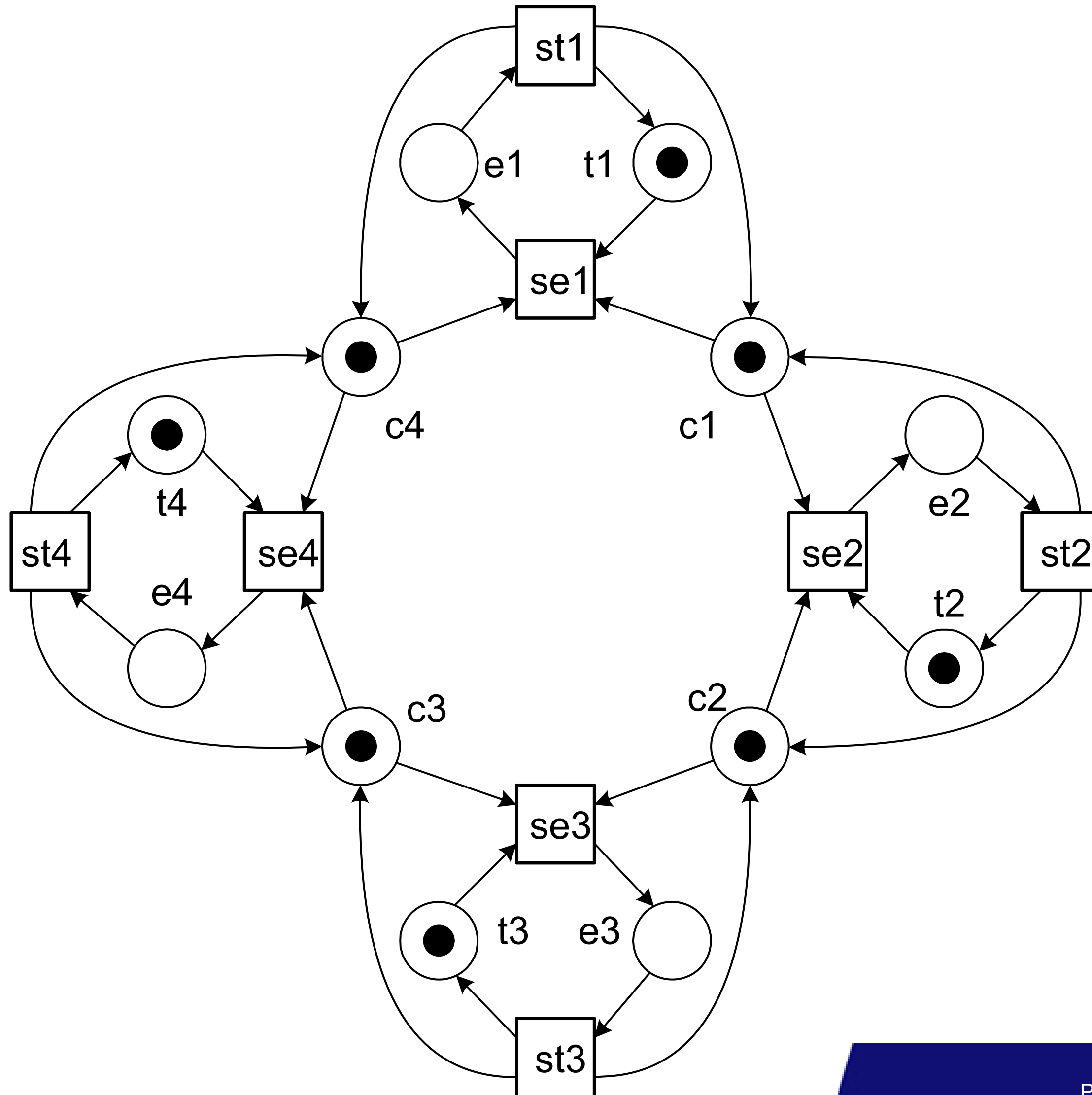


Example: Producers and Consumers



Example: Four Philosophers

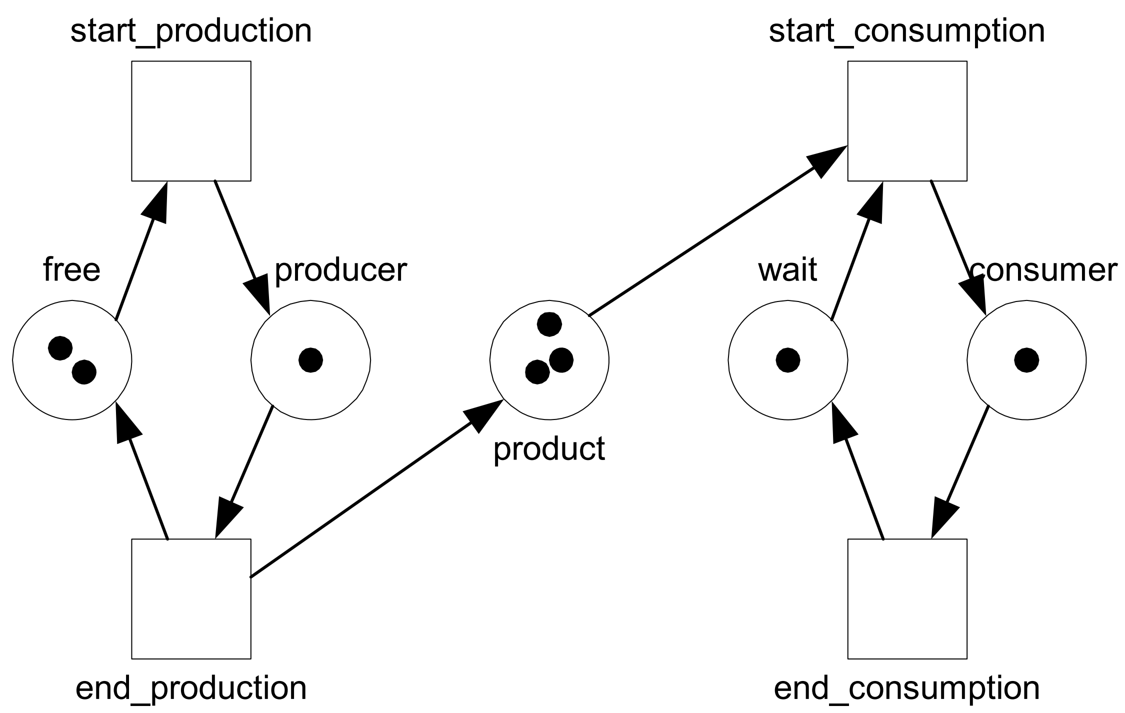




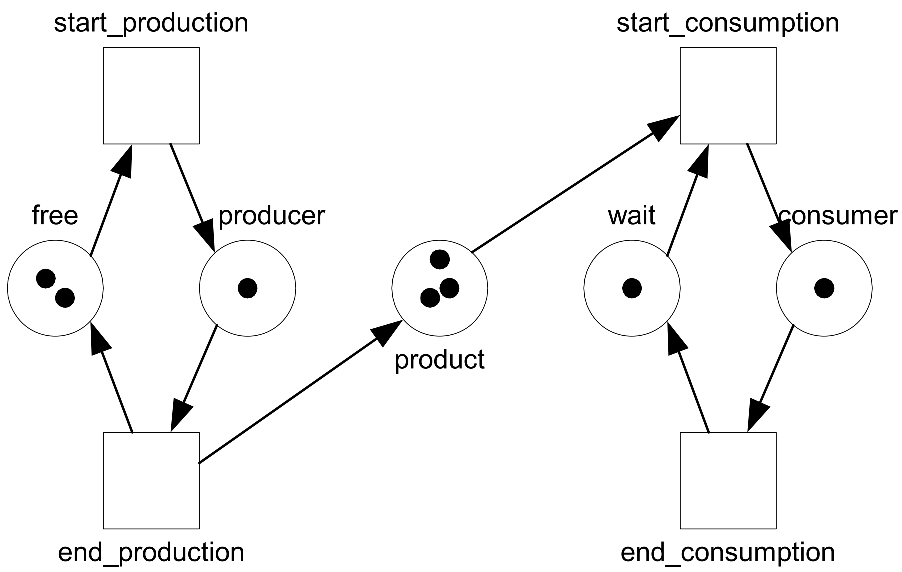
Definition: Petri Net

Definition 1 (Petri net). *A* Petri net $N = [P, T, F, m_0]$ *consists of*

- *two finite and disjoint sets P and T of* places *and* transitions,
- *a* flow relation $F \subseteq (P \times T) \cup (T \times P)$, *and*
- *an* initial marking m_0, *where a marking is a mapping* $m : P \rightarrow \mathbb{N}$.



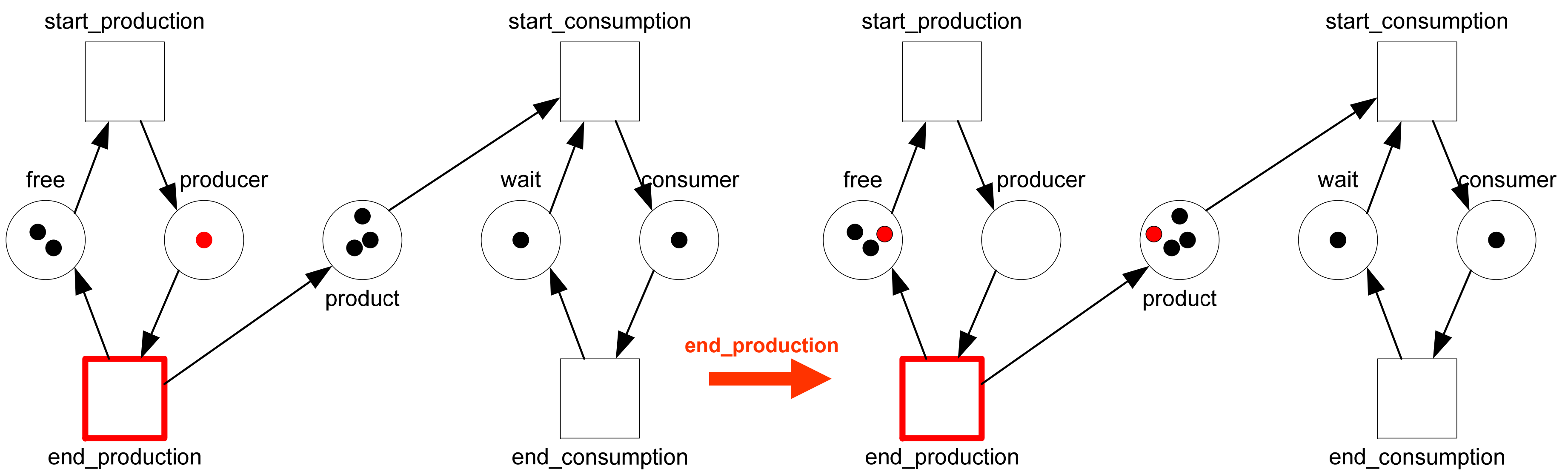
Pre- and Post-Set



For the flow relation of a Petri net N we introduce the following notation to denote the pre-set and the post-set of places and transitions. Let $x \in P \cup T$ be a node of N. Then, ${}^\bullet x = \{y \mid [y, x] \in F\}$ denotes the *pre-set* of x (i.e. all nodes y that have an arc to x) and $x^\bullet = \{y \mid [x, y] \in F\}$ denotes the *post-set* of x (i.e. all nodes y with an arc from x to y).

Firing Rule

The dynamics of a Petri net N is defined by the *firing rule*. The firing rule defines *enabledness* of Petri net transitions and their effects. A transition t is enabled at a marking m if there is a token on every place in its pre-set. The firing of an enabled transition t yields a new marking m', which is derived from its predecessor marking m by consuming (i.e. removing) a token from each place of t's pre-set and producing (i.e. adding) a token on each place of t's post-set. The described firing relation is denoted $m \xrightarrow{t} m'$. Thereby $m \xrightarrow{t} m'$ is a *step of* N.

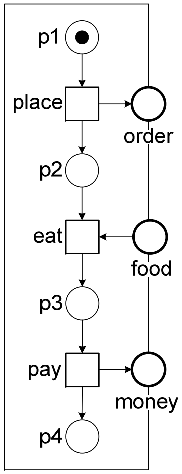


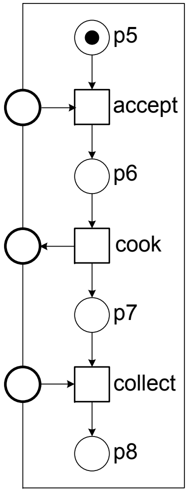
Open Nets

Definition 2 (Open net). *An* open net $N = [P, T, F, I, O, m_0, \Omega]$ *consists of a Petri net* $[P, T, F, m_0]$ *together with*

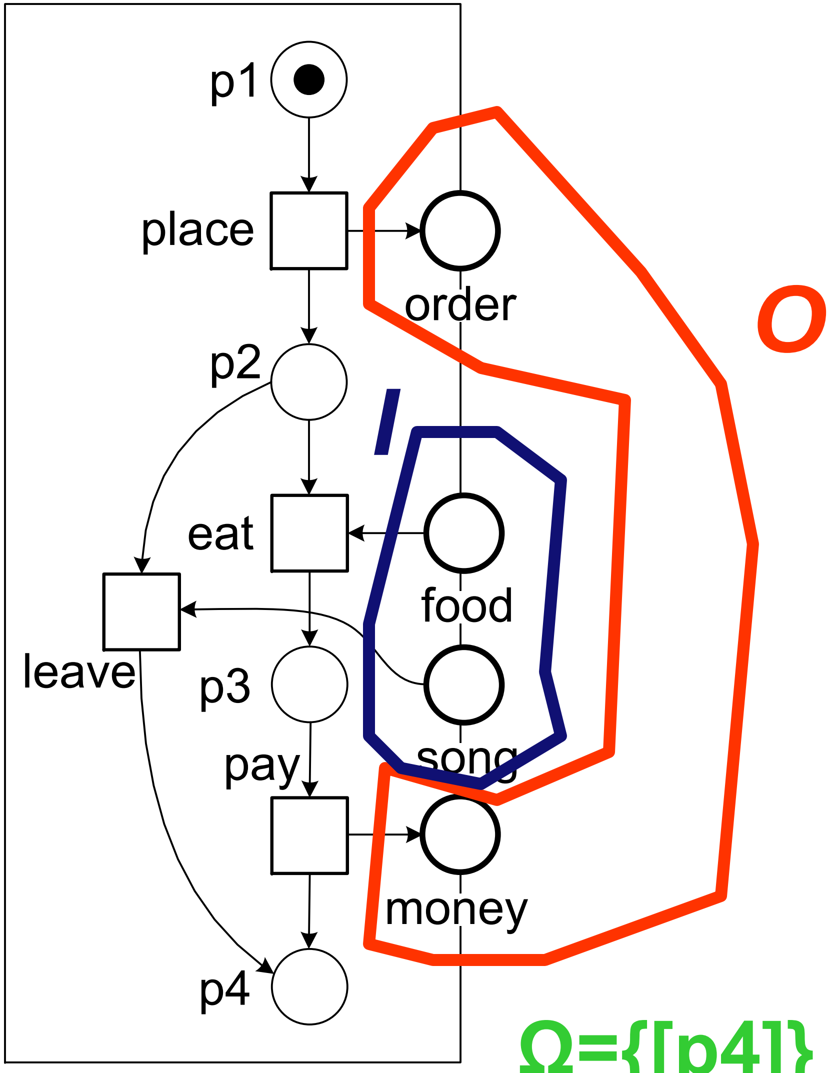
- *an* interface $(I \cup O) \subseteq P$ *defined as two disjoint sets* I *of* input places *and* O *of* output places *such that* ${}^\bullet p = \emptyset$ *for any* $p \in I$ *and* $p^\bullet = \emptyset$ *for any* $p \in O$*, and*
- *a set* Ω *of* final markings.

We further require that in the initial and the final markings the interface places are not marked, i.e., for all $m \in \Omega \cup \{m_0\}$ *we have* $m(p) = 0$*, for all* $p \in I \cup O$.



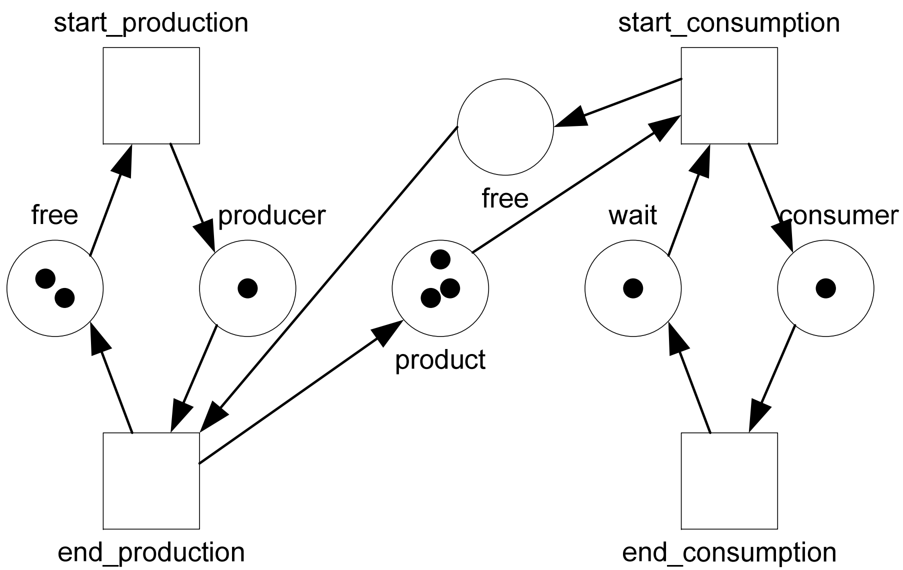


Example



Some More Definitions

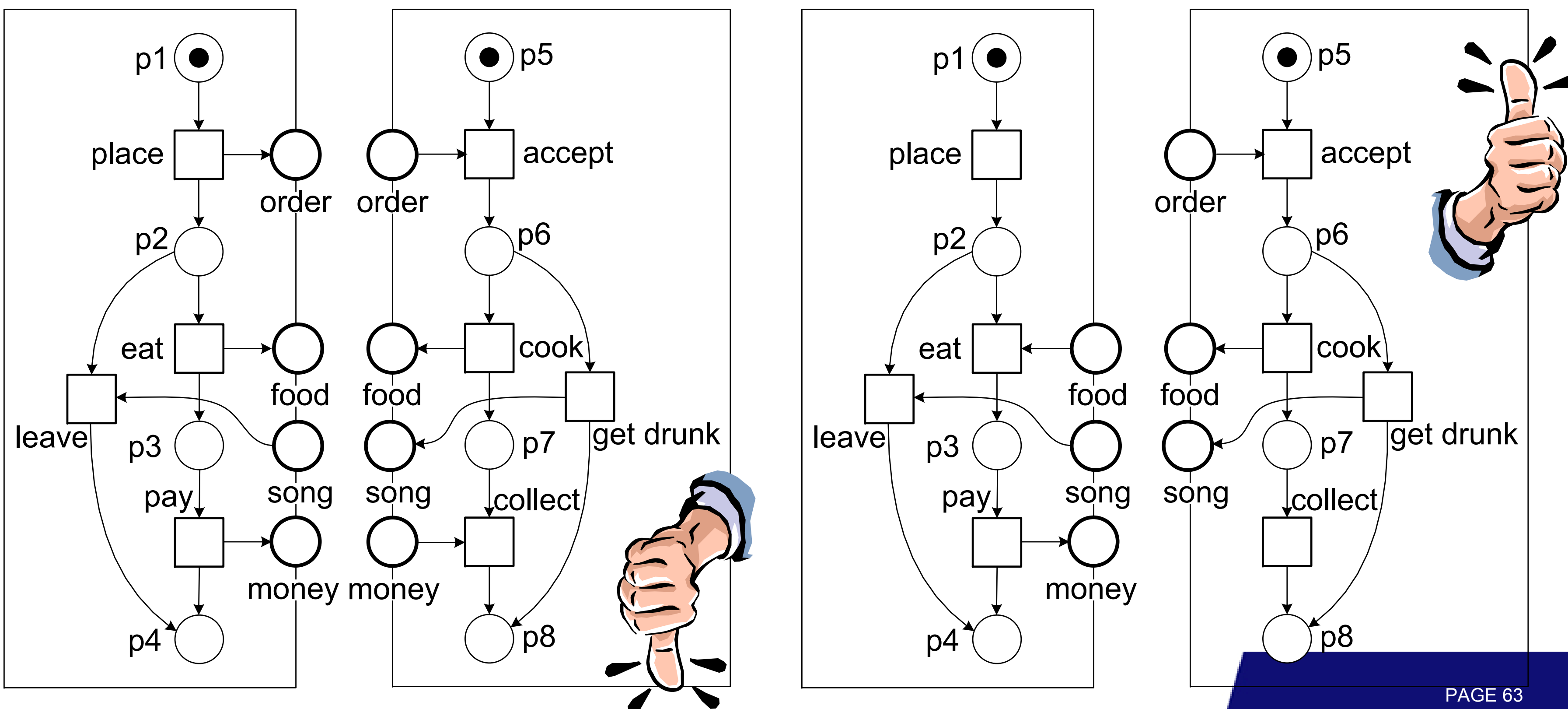
Definition 3 (Closed net). *An open net N with an empty interface, i.e., $I_N = \emptyset$ and $O_N = \emptyset$, is a* closed net.



Definition 4 (Boundedness). *A closed net N is* k-bounded *if there exists a $k \in \mathbb{N}$ such that for each reachable marking $m \in R_{(N)}(m_0)$, $m(p) \leq k$, for all $p \in P_N$.*

Interface Compatible

Definition 5 (Interface compatible open nets). *Let N_1, N_2 be two open nets with pairwise disjoint constituents except for the interfaces. If only input places of one open net overlap with output places of the other open net, i.e.,* $I_1 \cap I_2 = \emptyset$ *and* $O_1 \cap O_2 = \emptyset$, *then* N_1 *and* N_2 *are* interface compatible.



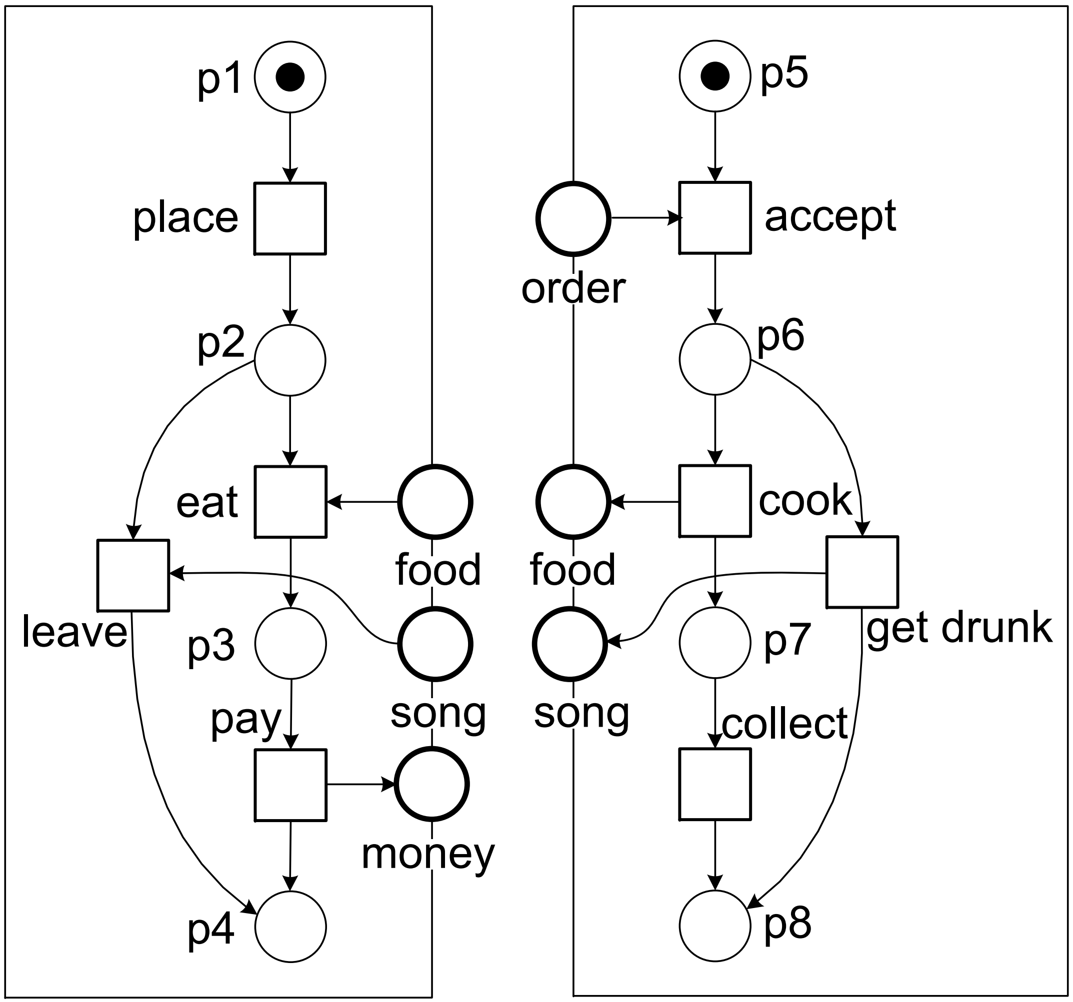
Composition

Definition 6 (Composition of open nets). *Let N_1 and N_2 be two interface compatible open nets. The* composition $N = N_1 \oplus N_2$ *is the open net with the following constituents:*

- $P = P_1 \cup P_2$,
- $T = T_1 \cup T_2$,
- $F = F_1 \cup F_2$,
- $I = (I_1 \cup I_2) \setminus (O_1 \cup O_2)$,
- $O = (O_1 \cup O_2) \setminus (I_1 \cup I_2)$,
- $m_0 = m_{01} \oplus m_{02}$, *and*
- $\Omega = \{m_1 \oplus m_2 \mid m_1 \in \Omega_1, m_2 \in \Omega_2\}$.

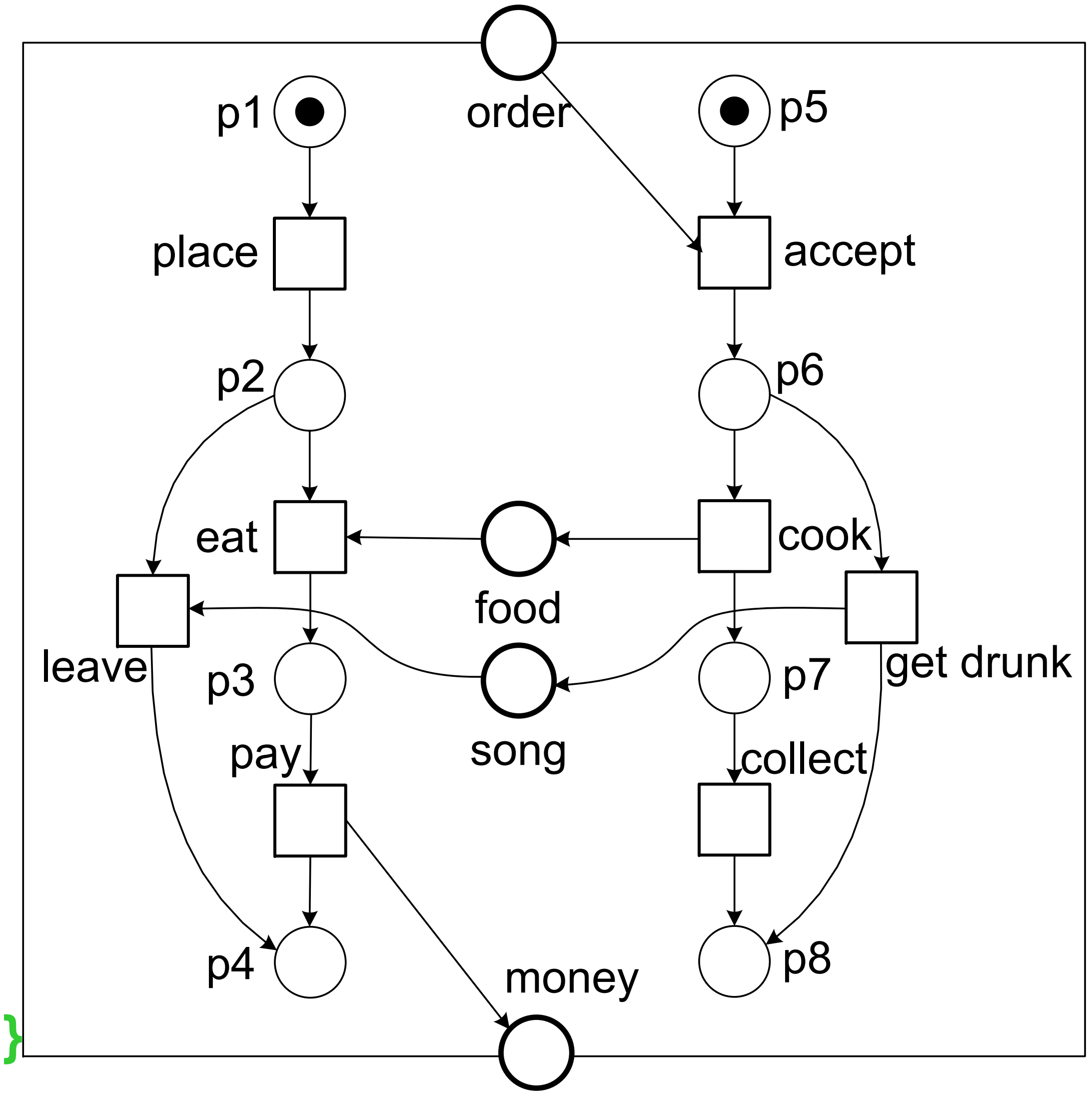
For markings m_1 of N_1 and m_2 of N_2 which do not mark the interface places, their composition $m = m_1 \oplus m_2$ is defined by $m(p) = m_i(p)$ if $p \in P_i$, for $i = 1, 2$.

Compose



$\Omega=\{[p4]\}$ $\Omega=\{[p8]\}$

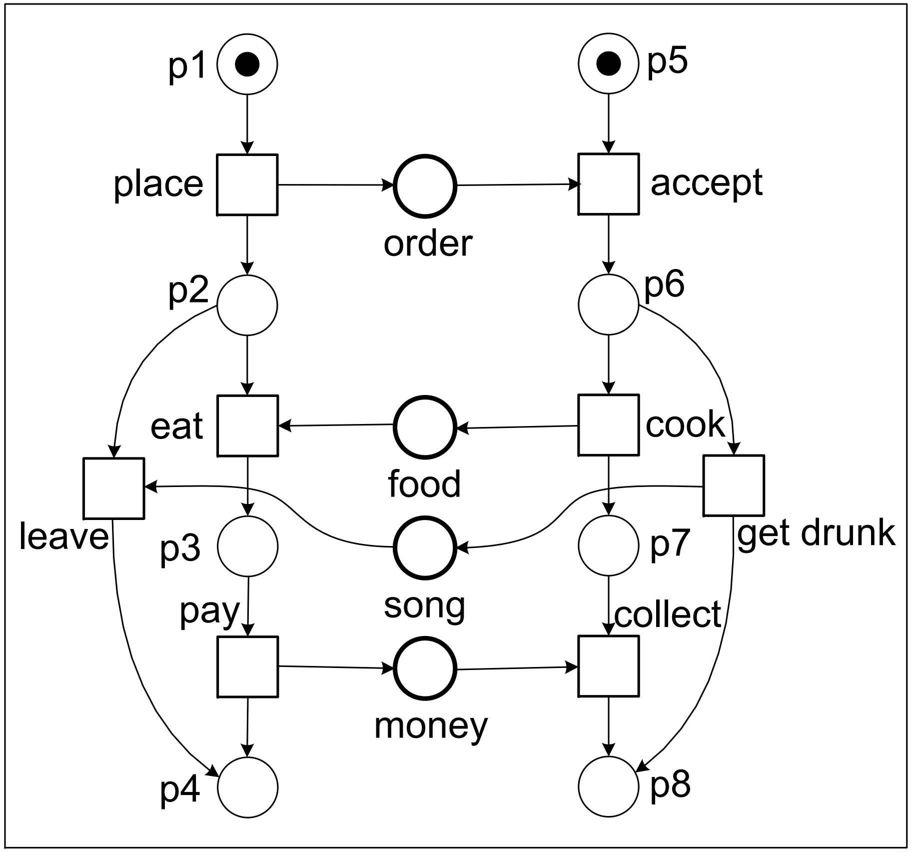
Composed Net



Ω={[p4,p8]}

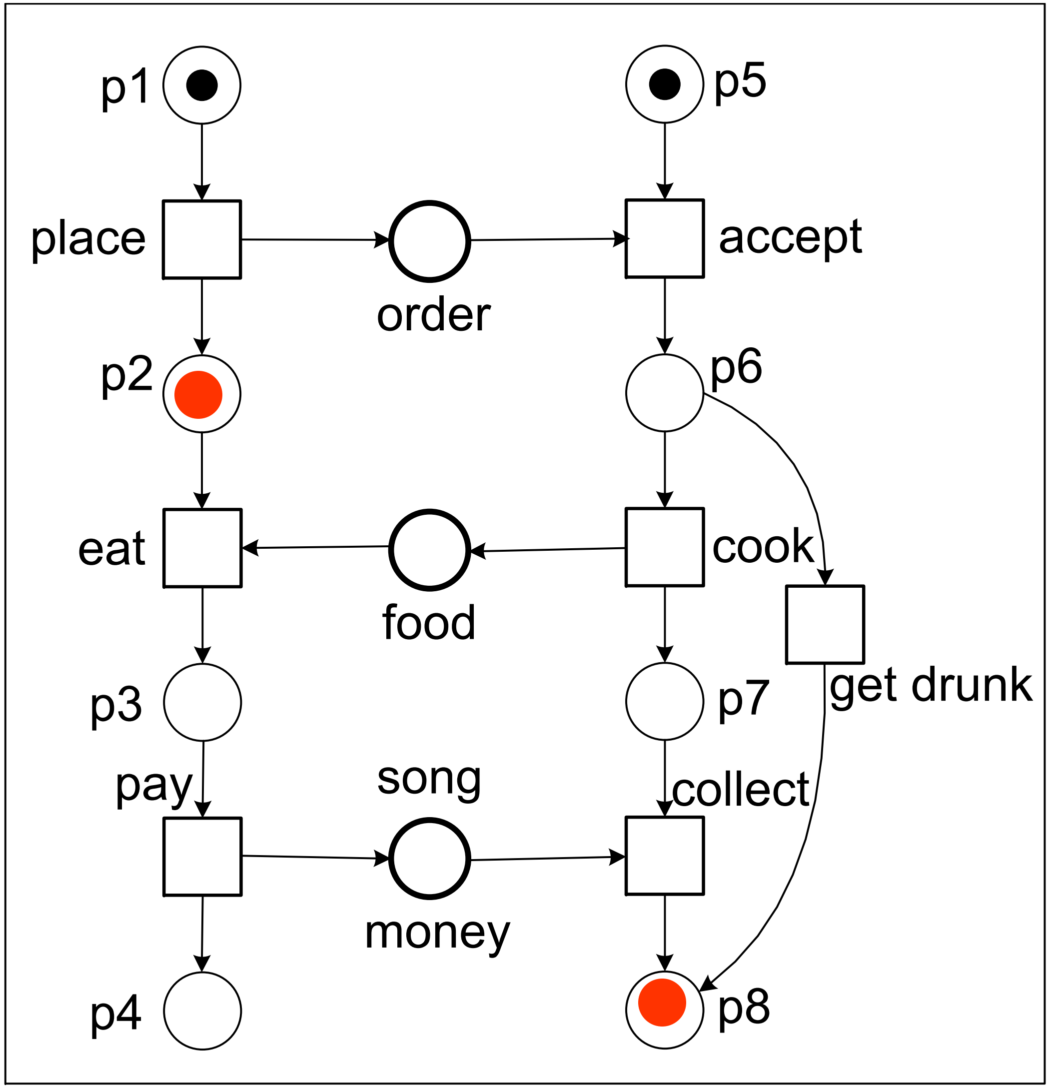
Deadlock Free

Definition 7 (Deadlock). *Let* $N = [P, T, F, I, O, m_0, \Omega]$ *be a closed net. A reachable marking* $m \in R_N(m_0)$ *is a* deadlock *in* N *iff* $m \notin \Omega$ *and no transition* $t \in T$ *is enabled in* m. *If no such* m *exists in* N, *then* N *is* deadlock-free.



$\Omega=\{[p4,p8]\}$

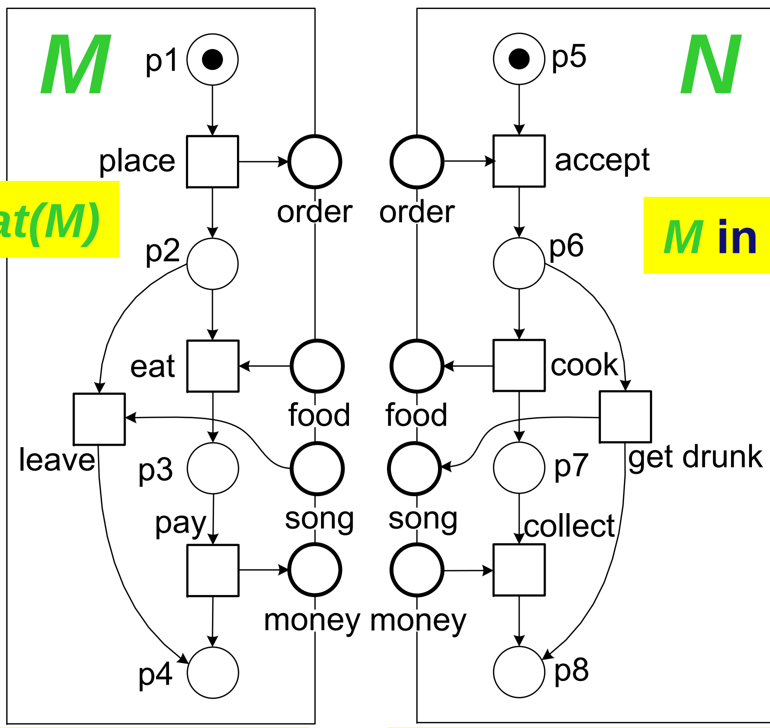
Deadlock Free ?



Ω={[p4,p8]}

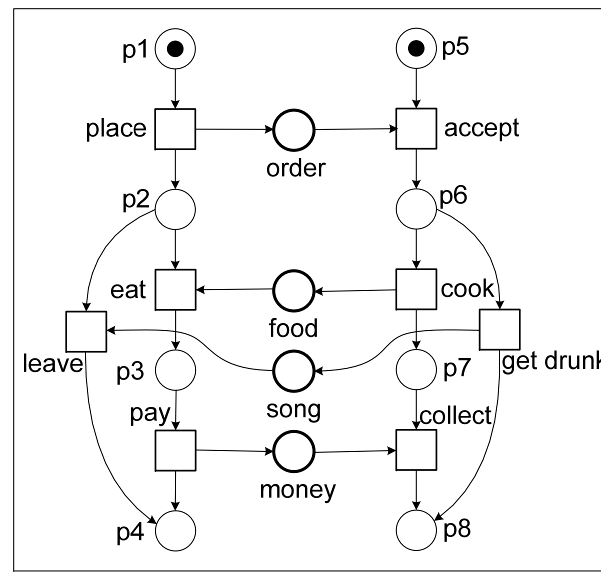
Strategy & Controllability

Definition 8 (Strategy, controllability). *Let M, N be two open nets such that $I_M = O_N$ and $O_M = I_N$. Then, M is a* strategy *for N iff $M \oplus N$ is deadlock-free. With Strat(N) we denote the set of all strategies for N. N is* controllable *iff its set of strategies is nonempty.*



N* in *Strat(M)

M* in *Strat(N)



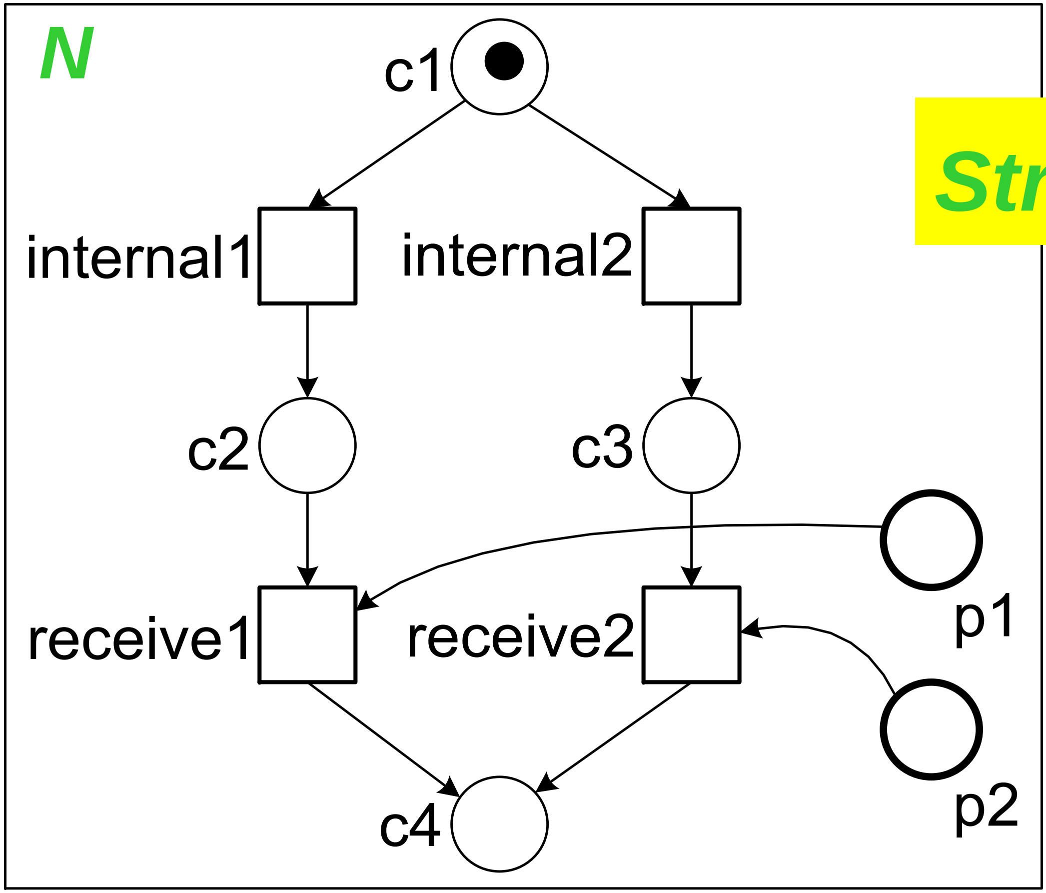
M* ⊕ *N

***M* is controllable**

***N* is controllable**

Controllable ?

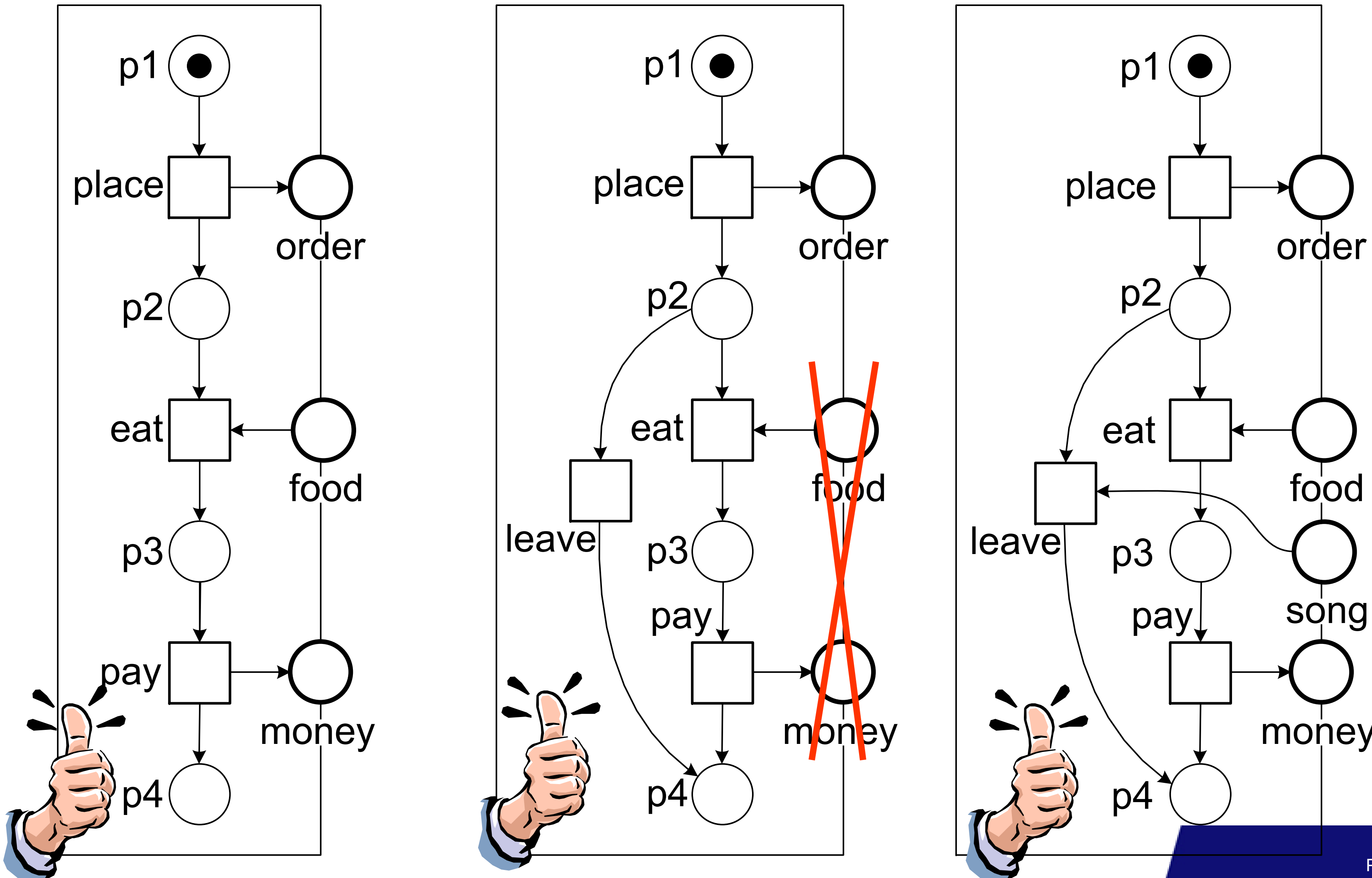
$\Omega=\{[c4]\}$



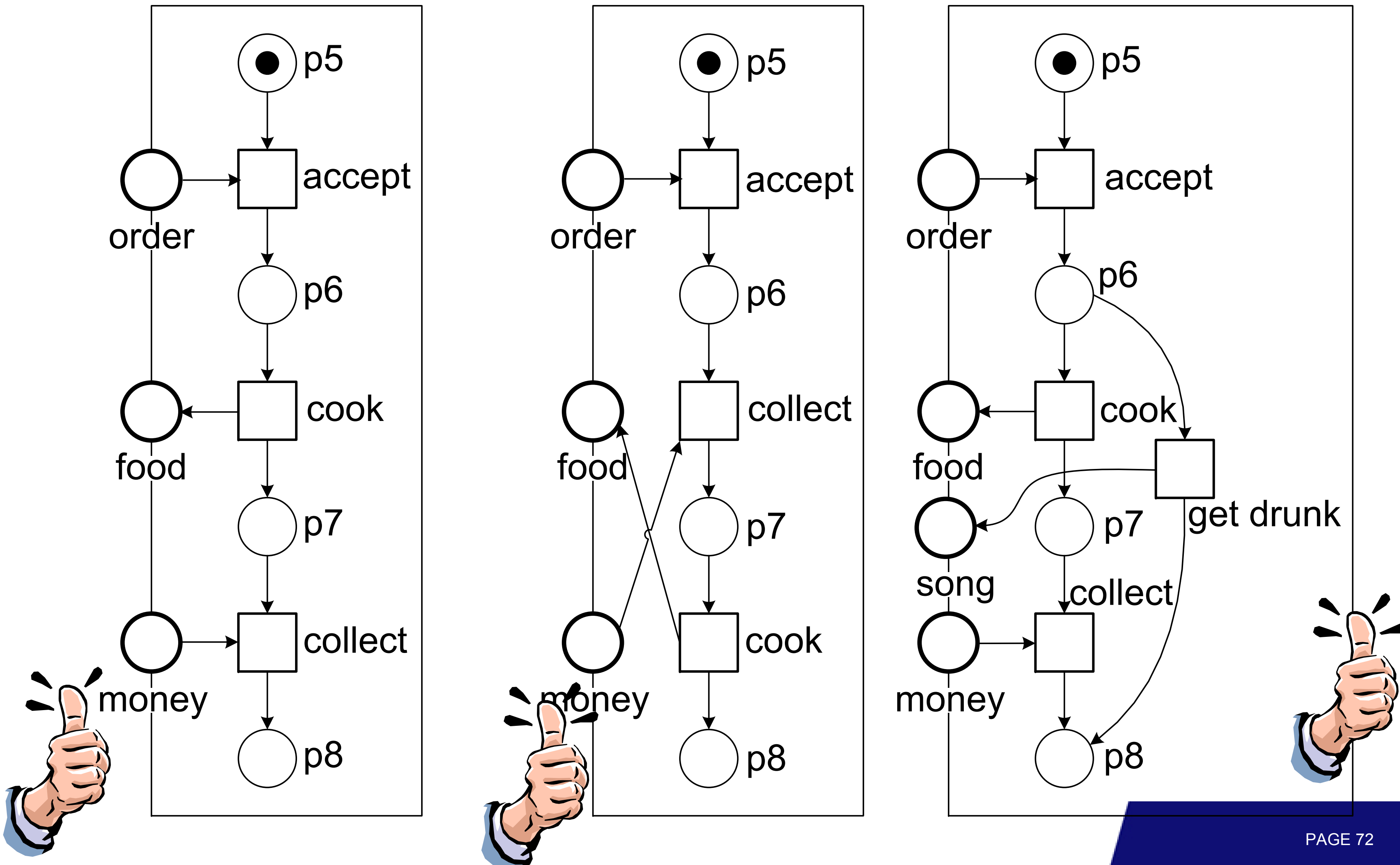
Strat(N) = Ø

AP-1: Internal Choice Receiving Follow-Up Anti-Pattern

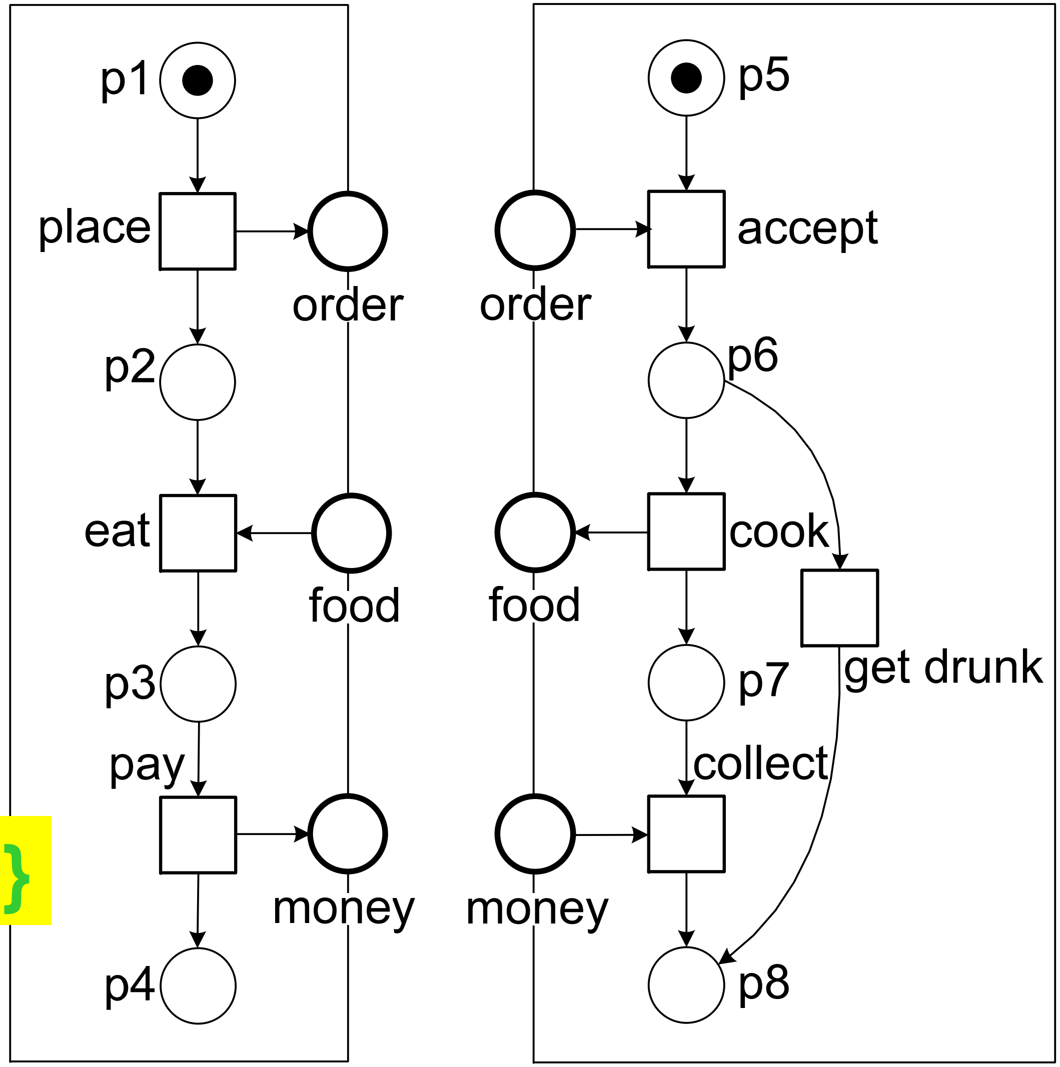
Controllable?



Controllable?

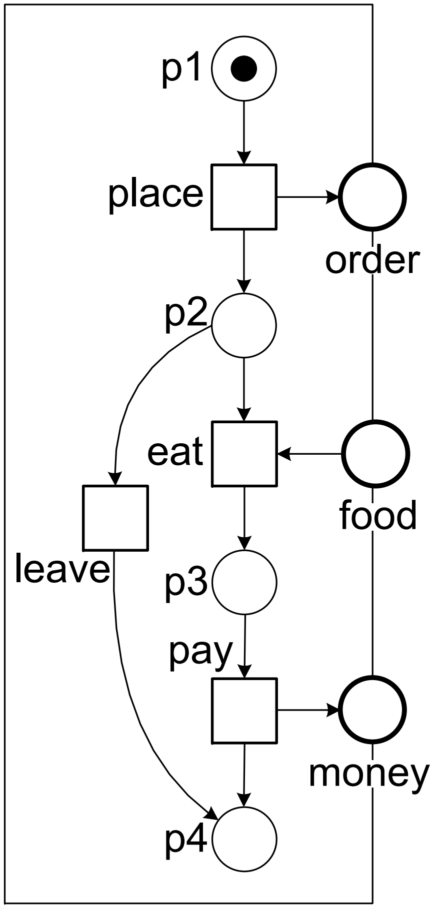


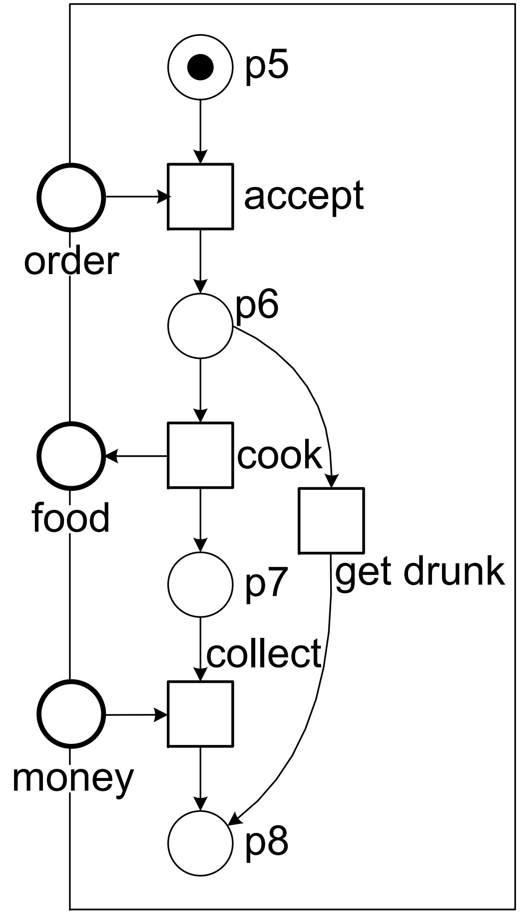
Controllable?



$\Omega=\{[p2,p4]\}$

Possible Additional Requirements to Rule Out Undesirable Strategies





No dead transitions / interface places

Ω states need to be dead

Recommended Reading

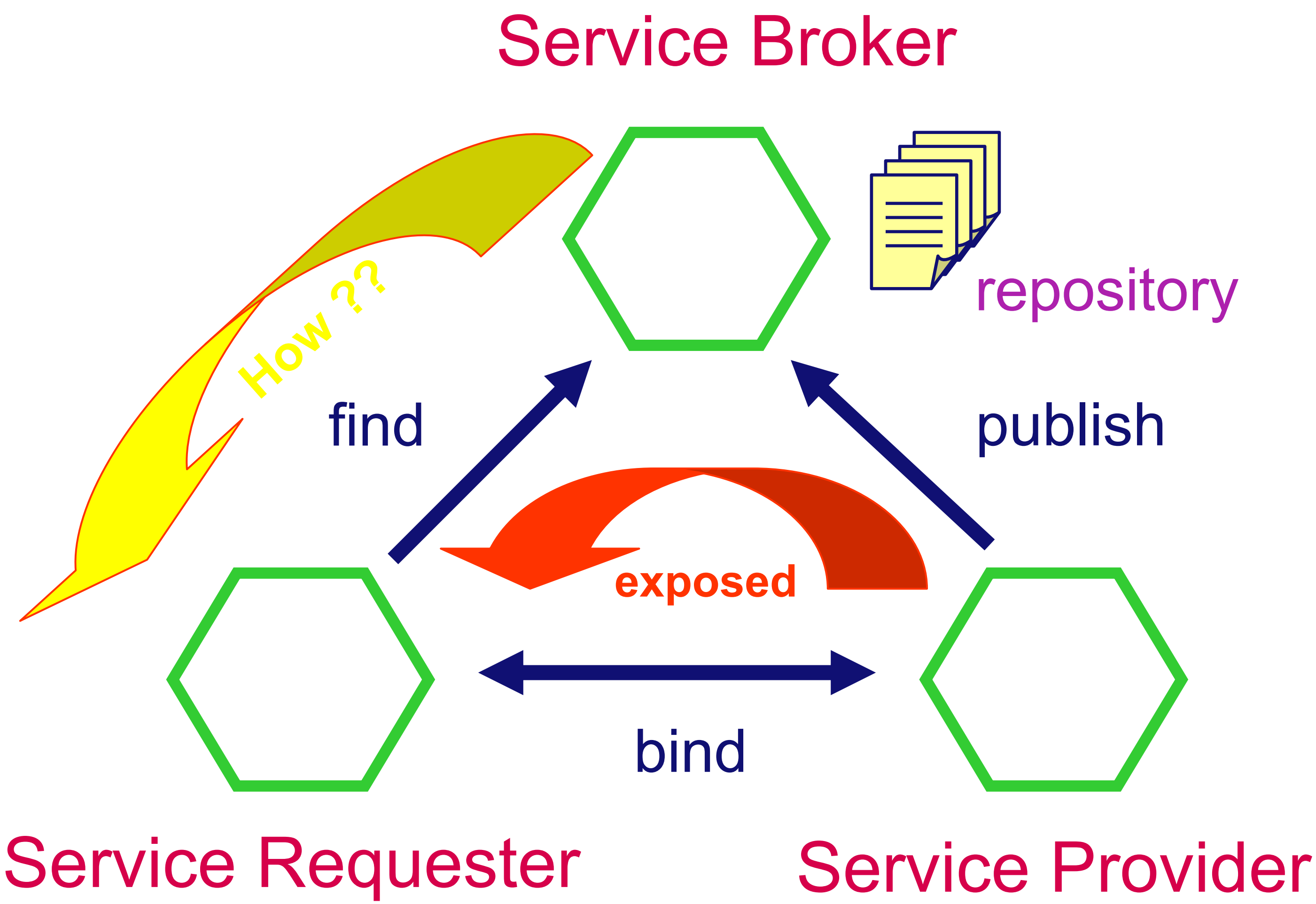
- van der Aalst, W.: The Application of Petri Nets to Workflow Management. The Journal of Circuits, Systems and Computers 8(1), 21–66 (1998)
- van der Aalst, W.M.P.: Business Process Management Demystified: A Tutorial on Models, Systems and Standards for Workflow Management. In J. Desel, W. Reisig, and G. Rozenberg, editors, *Lectures on Concurrency and Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*, pages 1-65. Springer-Verlag, Berlin, 2004.
- Desel, J., Esparza, J.: Free Choice Petri Nets. Cambridge Tracts in Theoretical Computer Science, vol. 40. Cambridge University Press, Cambridge (1995)
- Murata, T.: Petri Nets: Properties, Analysis and Applications. Proceedings of the IEEE 77(4), 541–580 (1989)
- Reisig, W.: Petri Nets: An Introduction. EATCS Monographs in Theoretical Computer Science, vol. 4. Springer, Berlin (1985)

Exposing Services

TU/e Technische Universiteit Eindhoven University of Technology

Where innovation starts

Exposing Services

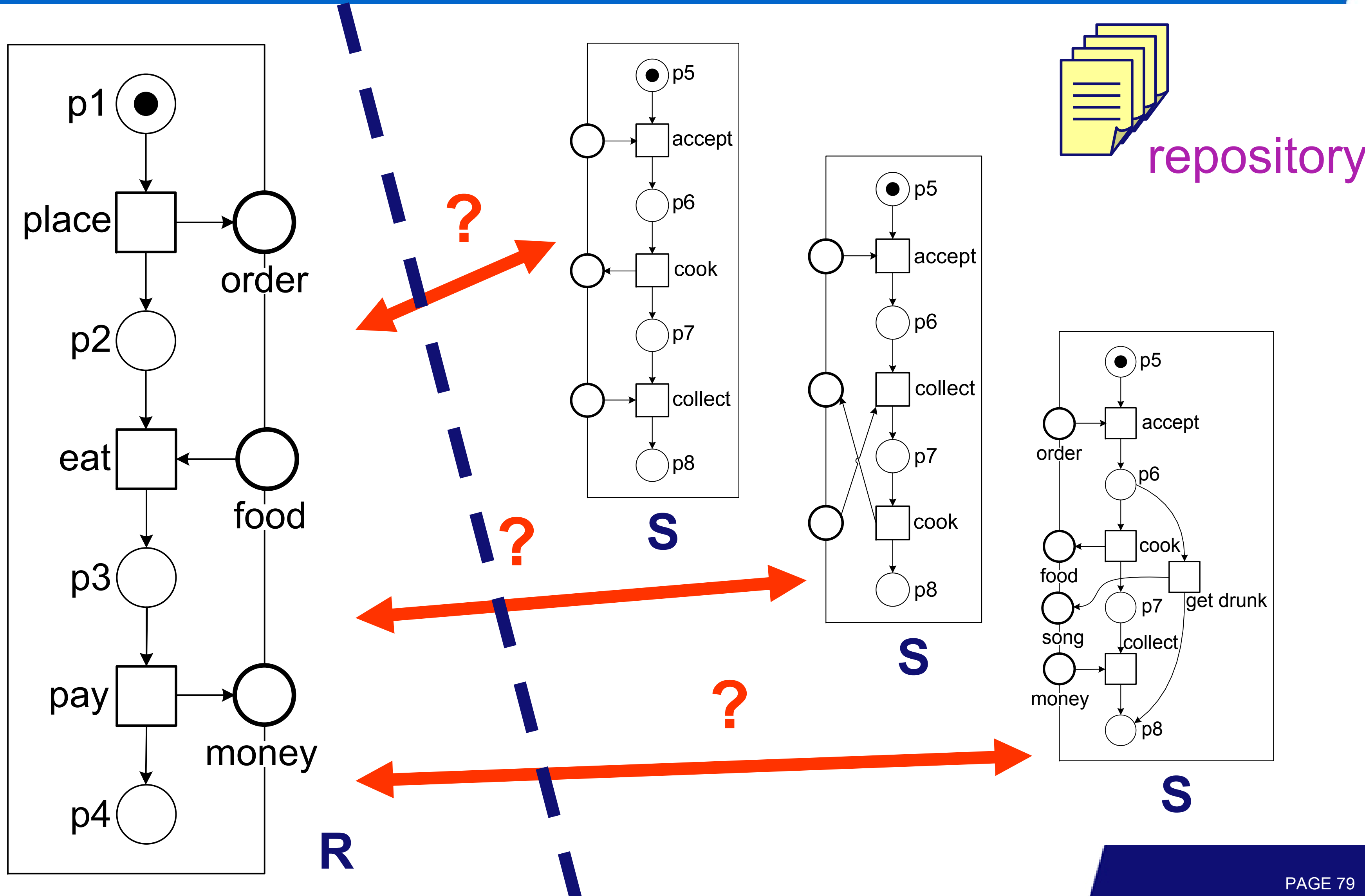


Services also need to be exposed in the bilateral case!

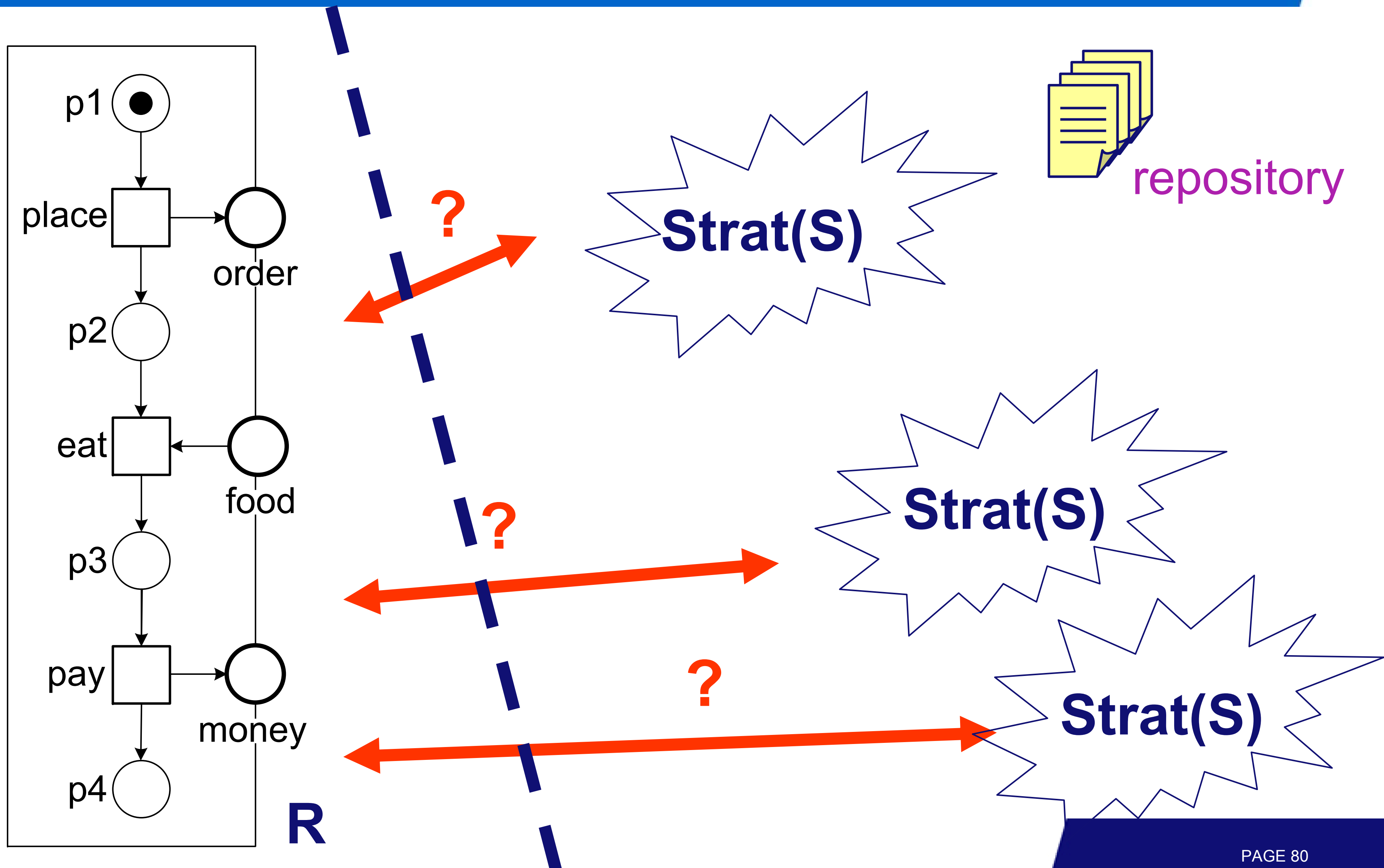
Two main approaches

- Selecting a service means to find for a given service *R* (whose behavior is given) a compatible service *S* in the repository.
- One approach is to expose the *behavior of S* (this needs to be done for all services in the repository).
- Well-behavior of the composition of *R* and *S* can be verified using standard state space verification techniques.
- However, organizations usually want to hide the trade secrets of their services and thus need to find a proper abstraction of *S* which is published.
- Another approach is to not expose the behavior of *S*, but a *class of services R* that is compatible with *S*, e.g., the set *Strat*(*S*).
- Then the composition of *R* and *S* is compatible if *Strat*(*S*) contains *R*. From the set of strategies it is in general not possible to derive the original service.

First Approach



Second Approach

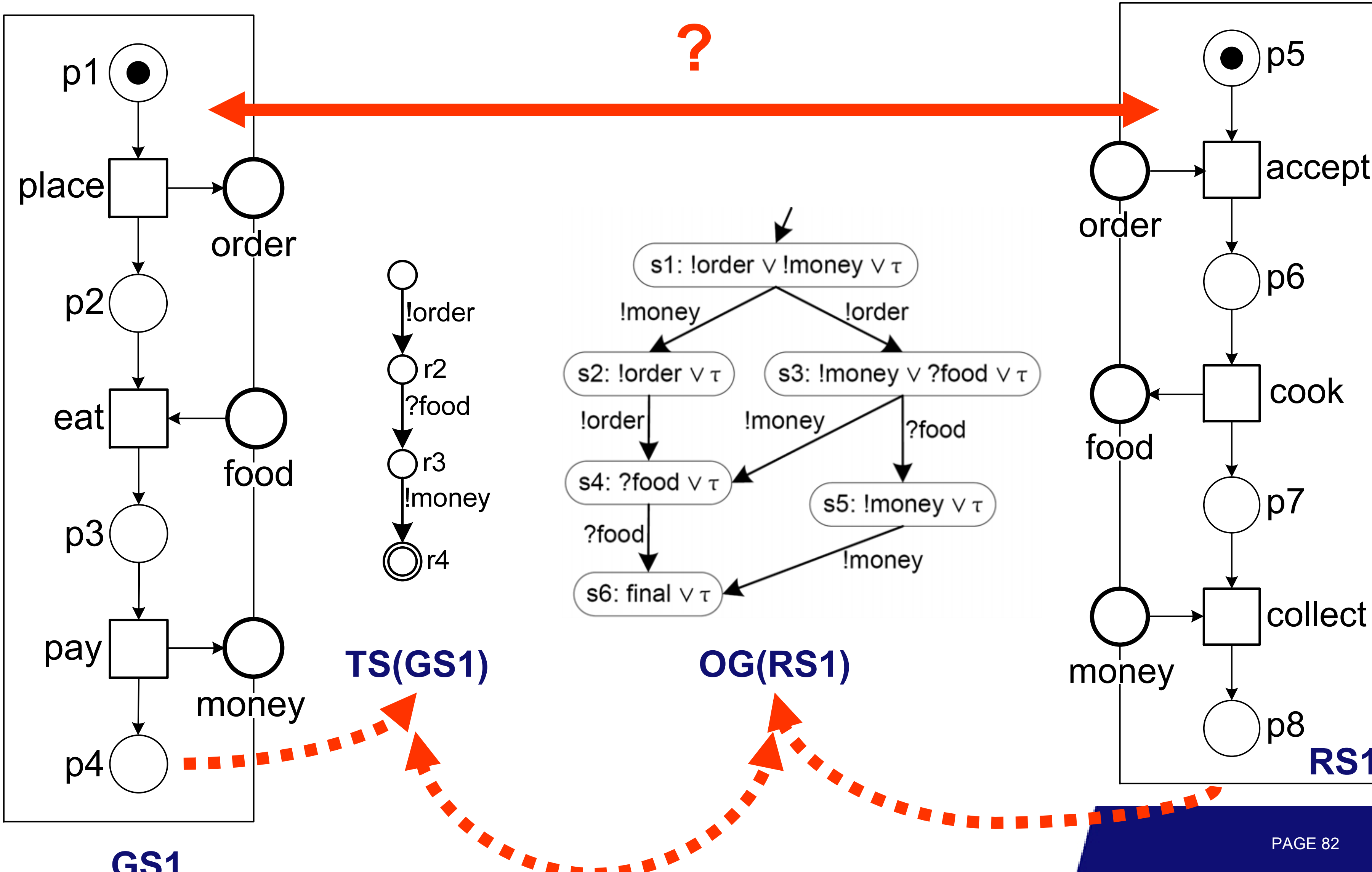


Operating Guidelines

- We advocate the second approach for reasons of efficiency and hiding trade secrets.
- Problem: Strat(S) is typically infinite!
- Operating guidelines provide a finite representation of a possibly infinite set of compatible services.

- Here we do not explain how the operating guideline is computed (see recommended reading) and focus on its application.

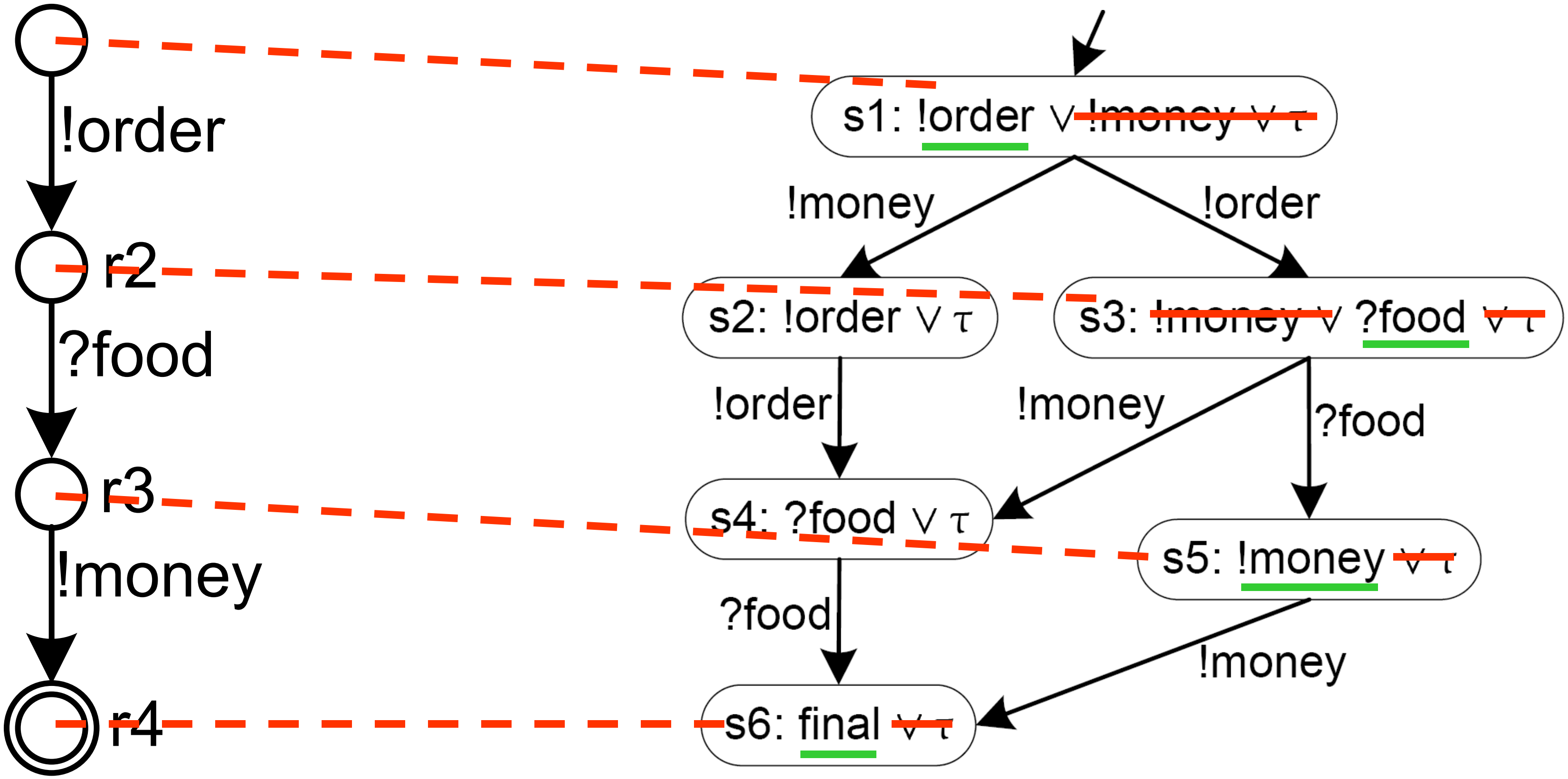
Basic Idea


?
p1
place
order
p2
eat
food
p3
pay
money
p4
GS1
!order
r2
?food
r3
!money
r4
TS(GS1)
s1: !order ∨ !money ∨ τ
!money
!order
s2: !order ∨ τ
s3: !money ∨ ?food ∨ τ
!order
!money
?food
s4: ?food ∨ τ
s5: !money ∨ τ
?food
!money
s6: final ∨ τ
OG(RS1)
p5
accept
order
p6
cook
food
p7
collect
money
p8
RS1

Matching

(2) constraints of corresponding states are satisfied

(1) simulation relation (weak simulation)

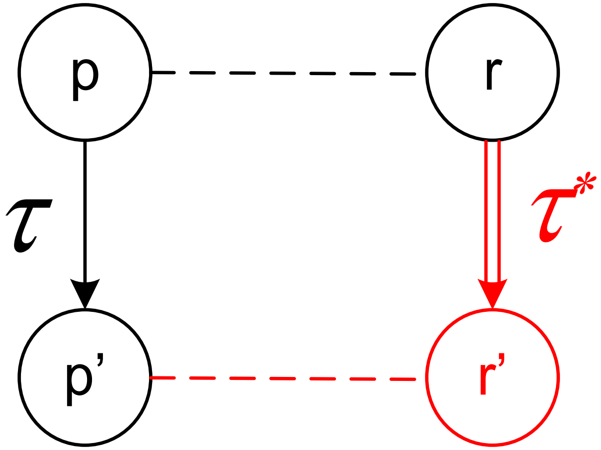


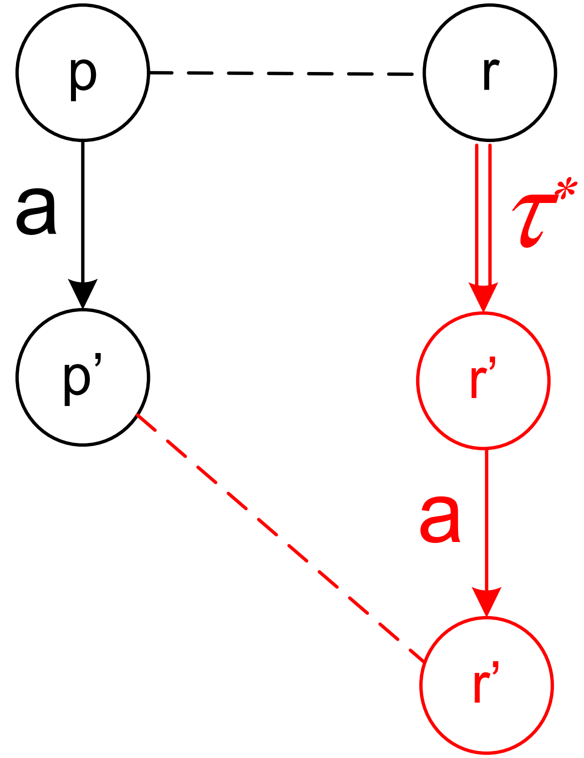
TS(GS1)

OG(RS1)

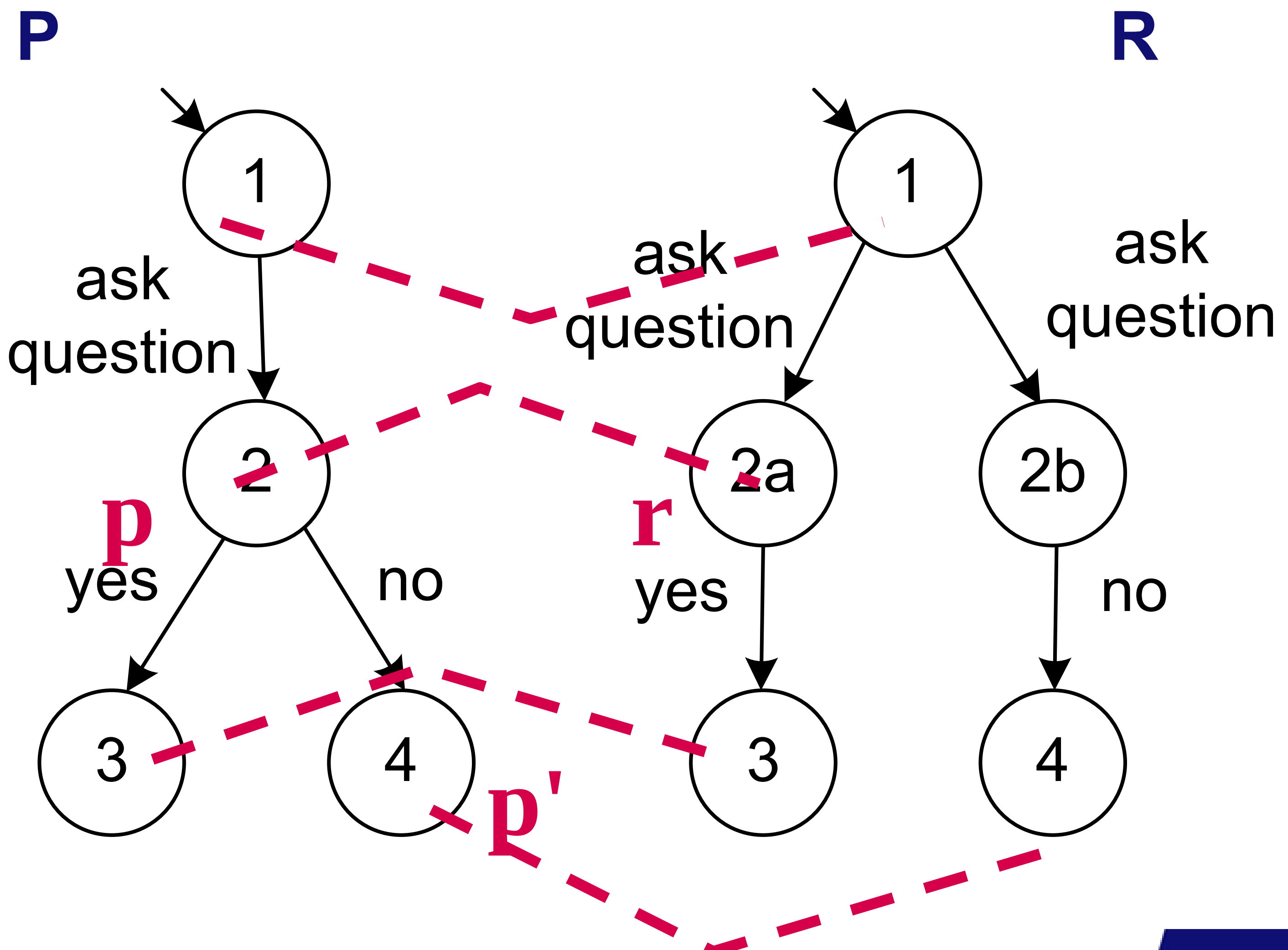
Weak simulation

- R **weakly simulates** P iff R can mimic any behavior of P.
- Formally, there exists a weak simulation relation such that:

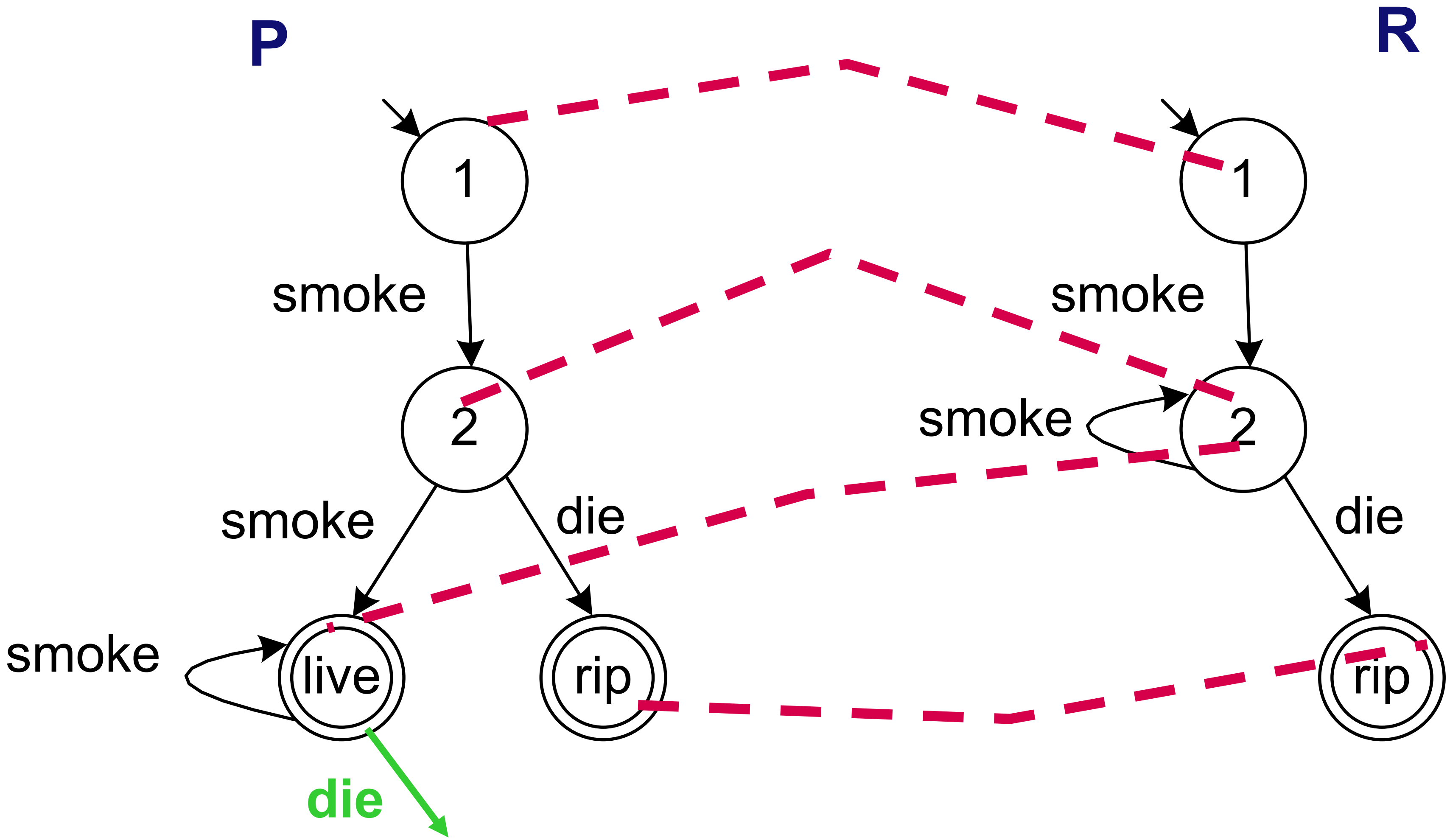




R does not weakly simulate P



R weakly simulates P, but ...
P does not weakly simulate R

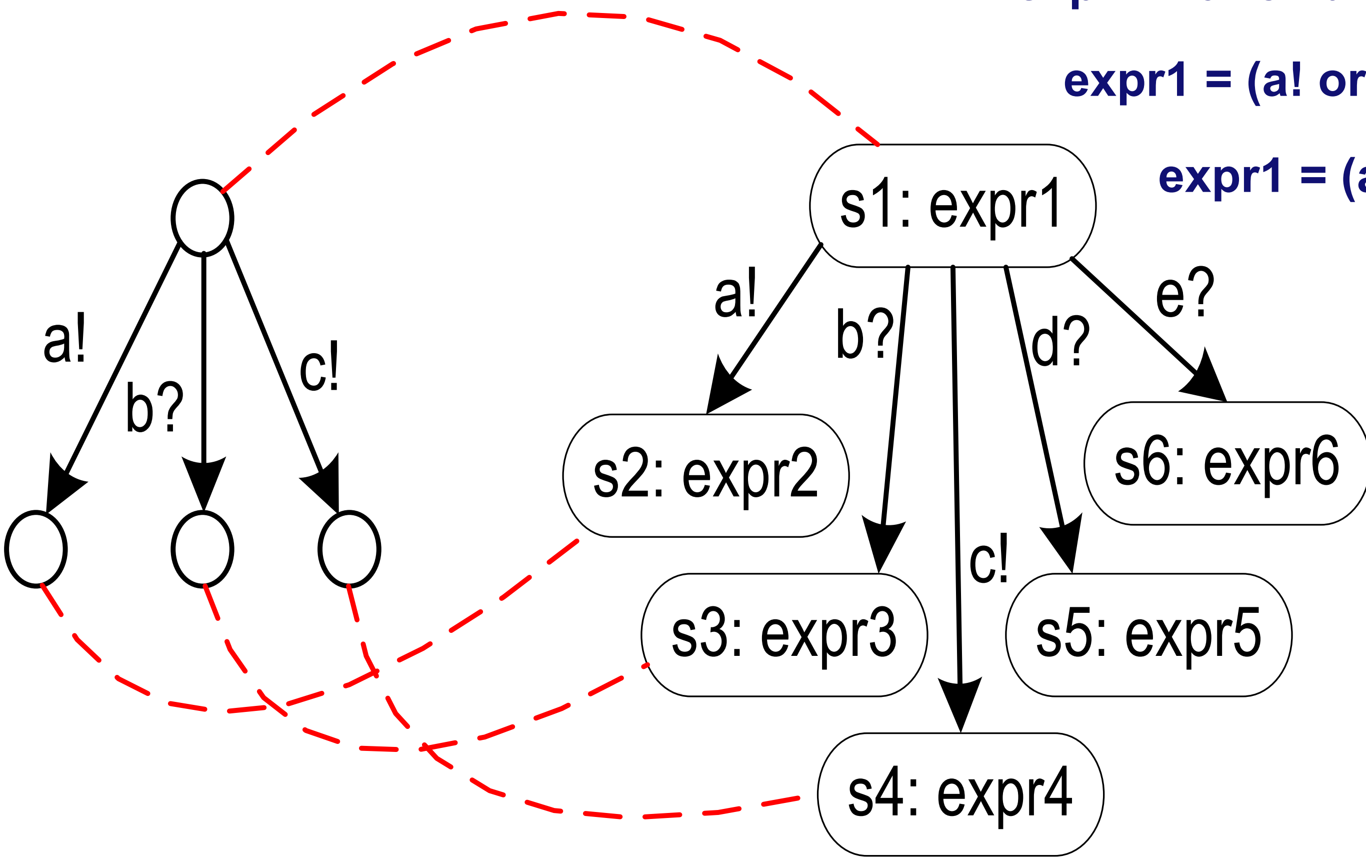


Evaluating Expressions

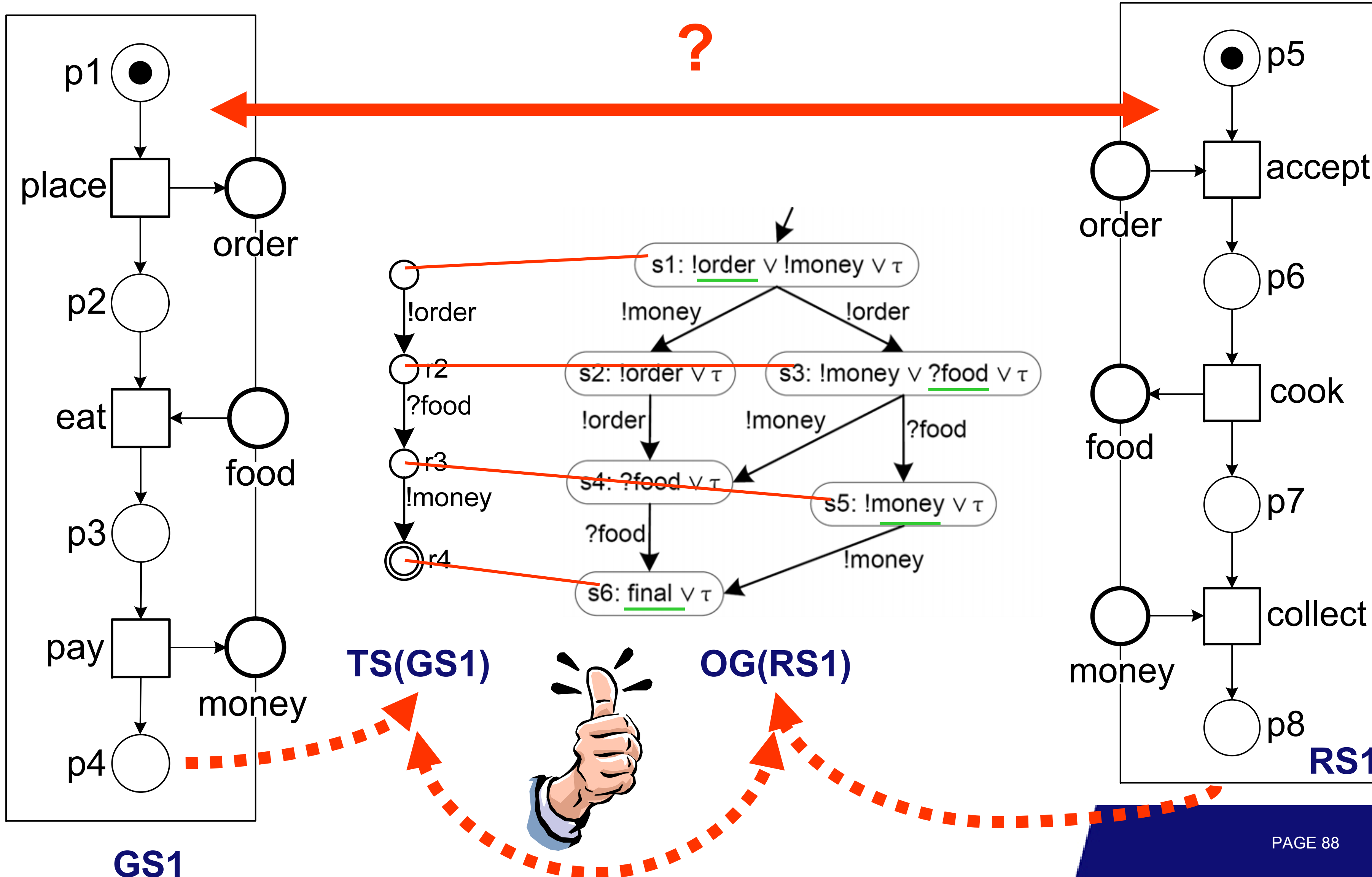
expr1 = a! or b? or c! or d? or f?

expr1 = (a! or b? or c!) and d?

expr1 = (a! and final) or d!



GS1 is a Strategy for RS1

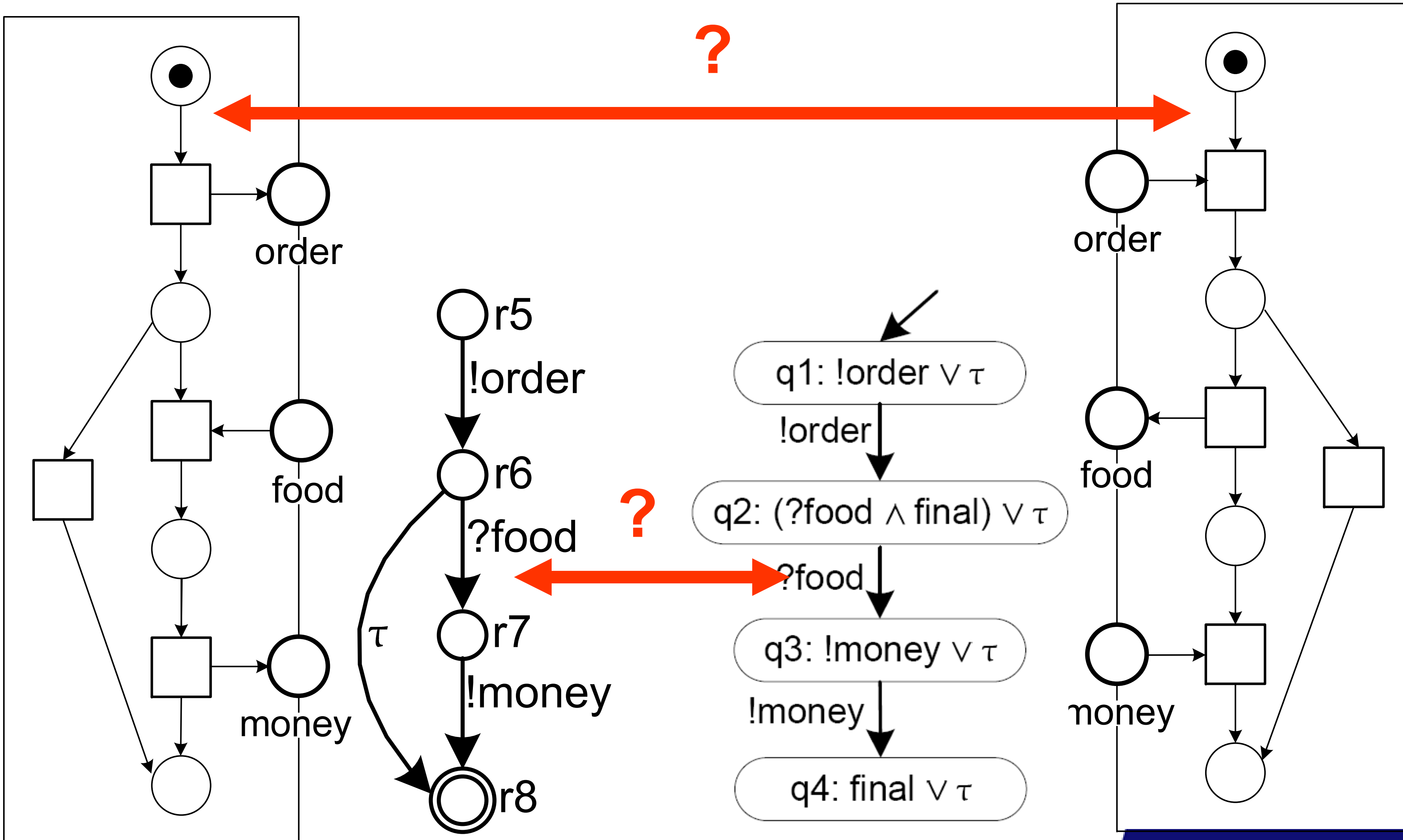


Operating Guideline

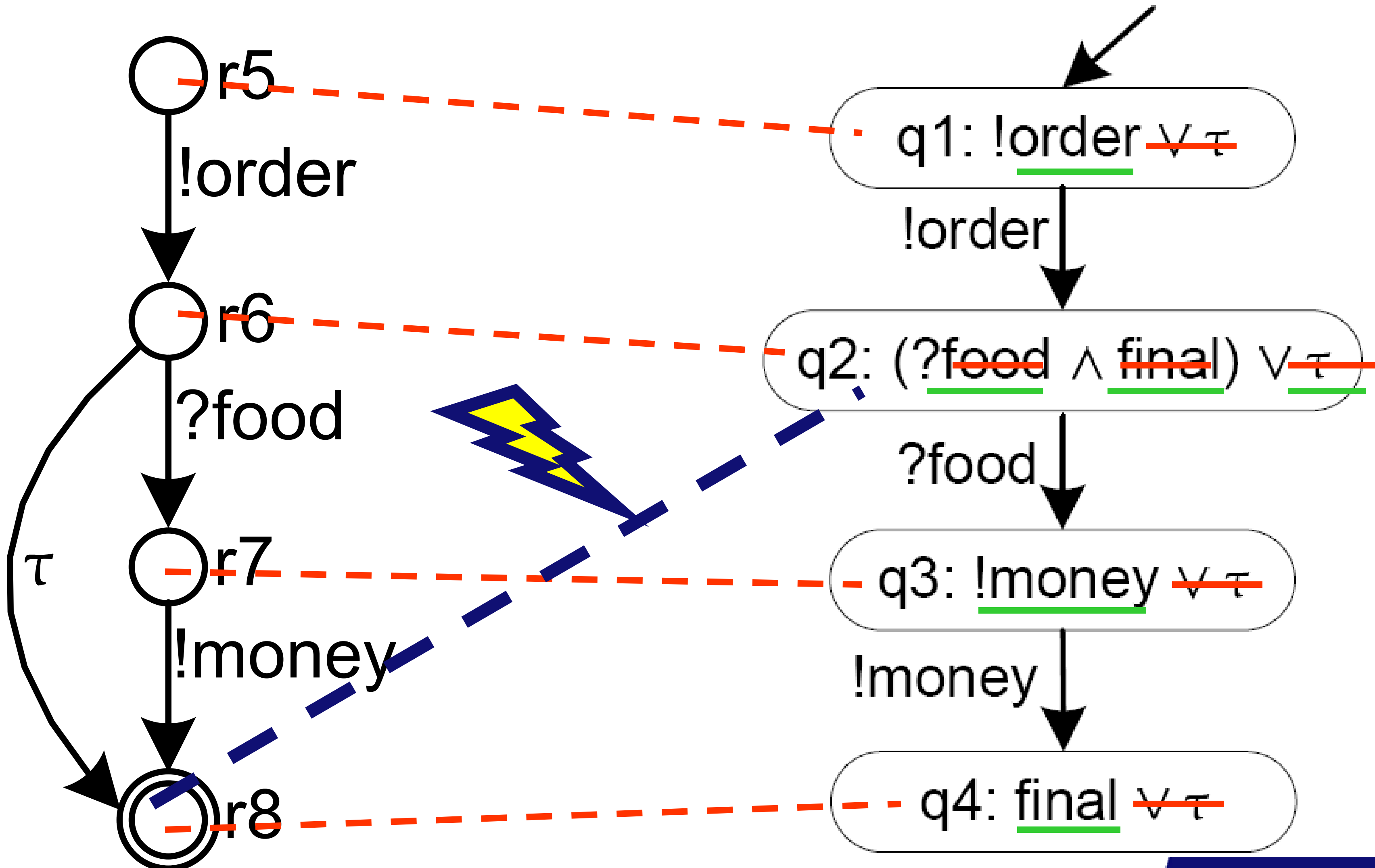
- N is an open net
- B^{Φ} is a Boolean Annotated Service Automaton (BSA), i.e., an automaton with annotated states that aims to describe (possible infinite) sets of open nets
- Match(B^{Φ}) is the set of all open nets that match with B^{Φ} (all nets that are weakly simulated by B^{Φ} such that constraints of corresponding states are satisfied)
- OG(N) is the operating guideline for N, i.e., a BSA B^{Φ} such that Match(B^{Φ}) = Strat(N)

Definition 16 (Operating guidelines, OG). *The operating guidelines $OG(N)$ of an open net N is a BSA such that $Match(OG(N)) = Strat(N)$.*

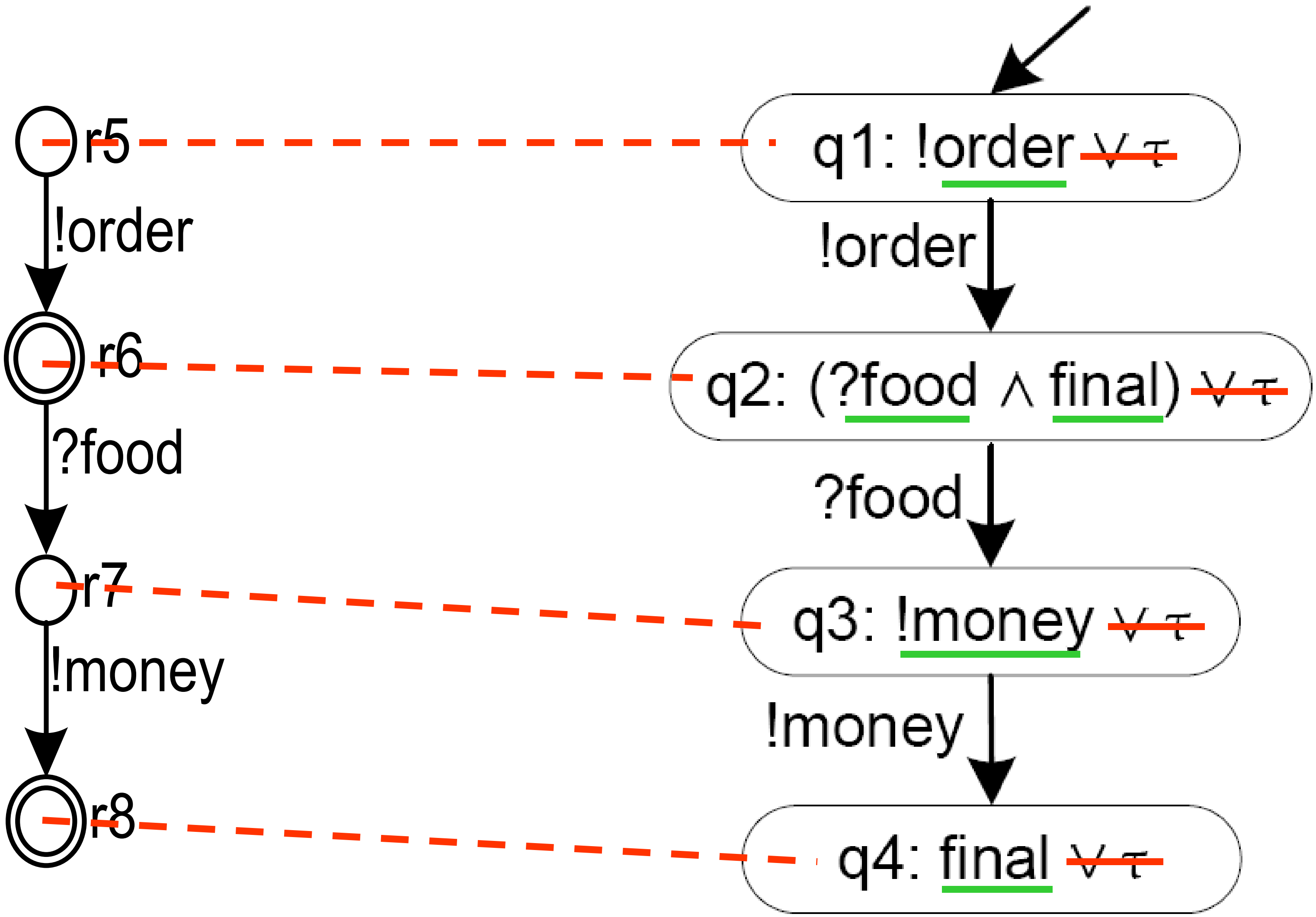
Another Example



Not Matching



Repaired Service Behavior



Most Permissive Strategy

Definition 17 (Most permissive strategy). *Let* $OG(N) = [Q, MP, \delta, q_0, \Phi]$ *be the operating guidelines for a controllable open net* N. *Then, an open net* M *is the most permissive strategy for* N *iff* $TS(M) = [Q, MP, \delta, q_0, \Omega]$, *where* $\Omega = \{q \mid \mathit{final} \text{ occurs in } \Phi(q)\}$.

Recommended Reading

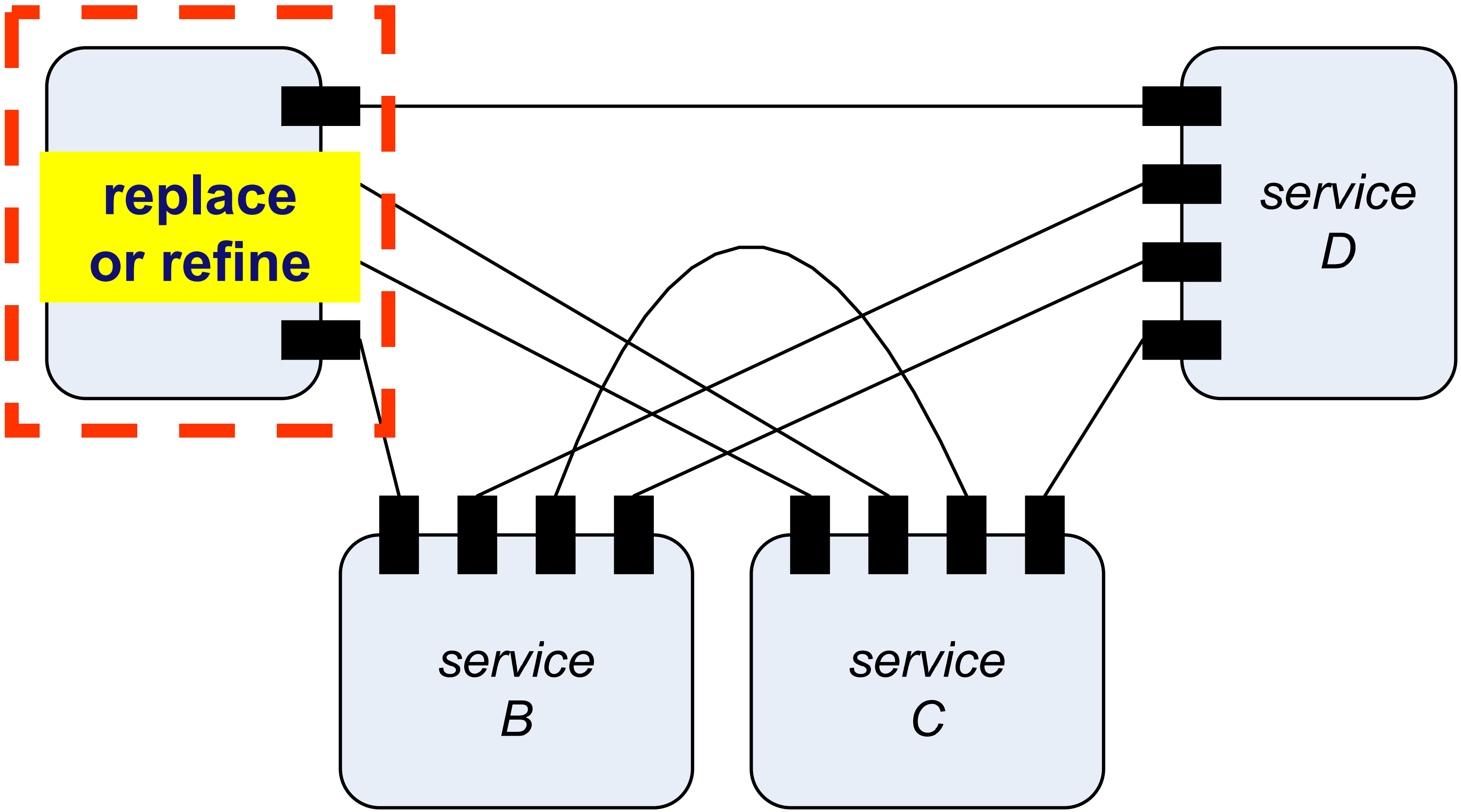
- van der Aalst, W., Mooij, A.J., Stahl C., Wolf, K.. Service Interaction: Patterns, Formalization, and Analysis. In SFM 2009, volume 5569 of Lecture Notes in Computer Science, pages 42-88. Springer-Verlag, Berlin (2009)
- van der Aalst, W., Lohmann, N., Massuthe, P., Stahl, C., Wolf, K.: From Public Views to Private Views: Correctness-by-Design for Services. In: Dumas, M., Heckel, H. (eds.) WS-FM 2007. LNCS, vol. 4937, pp. 139–153. Springer, Heidelberg (2008)
- Massuthe, P., Reisig, W., Schmidt, K.: An Operating Guideline Approach to the SOA. Annals of Mathematics, Computing & Teleinformatics 1(3), 35–43 (2005)
- Wolf, K.: Does my service have partners? In: ToPNoC II 2008. LNCS, vol. 5460, pp. 152–171. Springer, Heidelberg (2008)
- Massuthe, P., Reisig, W., Schmidt, K.: An Operating Guideline Approach to the SOA. In: Proceedings of the 2nd South-East European Workshop on Formal Methods 2005 (SEEFM 2005), Ohrid, Republic of Macedonia (2005)

Replacing and Refining Services

TU/e Technische Universiteit Eindhoven University of Technology

Where innovation starts

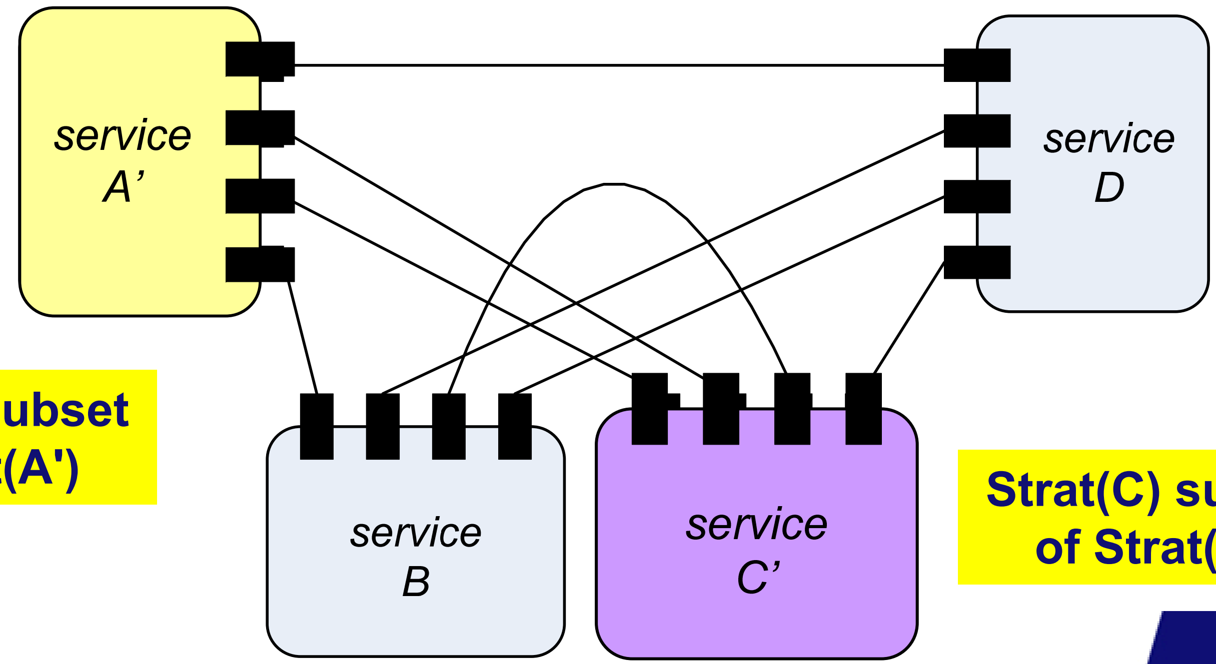
Replacing or Refining Services



Accordance

Definition 18 (Interface equivalent open nets). *Two open nets M and N are* interface equivalent *iff $I_M = I_N$ and $O_M = O_N$.*

Definition 19 (Accordance). *Let N and N' be two interface equivalent open nets. N'* can replace N under accordance *(N' accords with N, for short) iff $Strat(N) \subseteq Strat(N')$.*

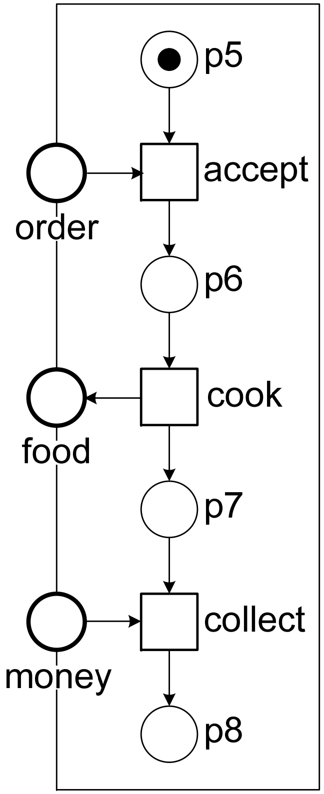


Strat(A) subset of Strat(A')

Strat(C) subset of Strat(C')

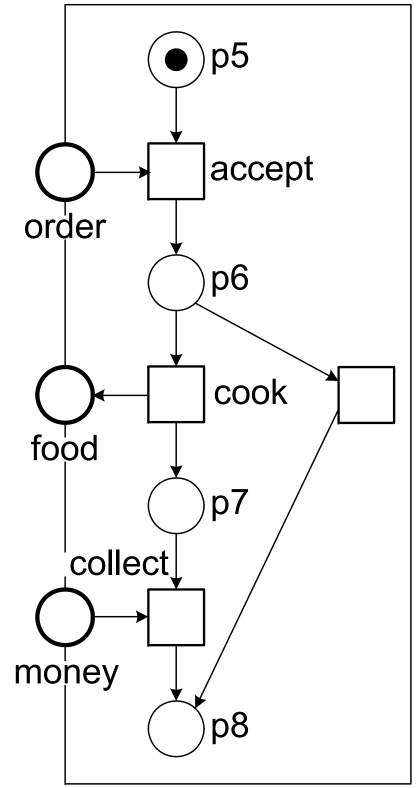
Projection Inheritance is Too Strict

Theorem 1 **(Projection inheritance implies accordance [37]).** *Let N and N' be two open nets. If N and N' are related by projection inheritance, then N' accords with N and N accords with N'.*



accords with

(while there is no inheritance relation)



Refinement

$$OG(N) \sqsubseteq OG(N')$$

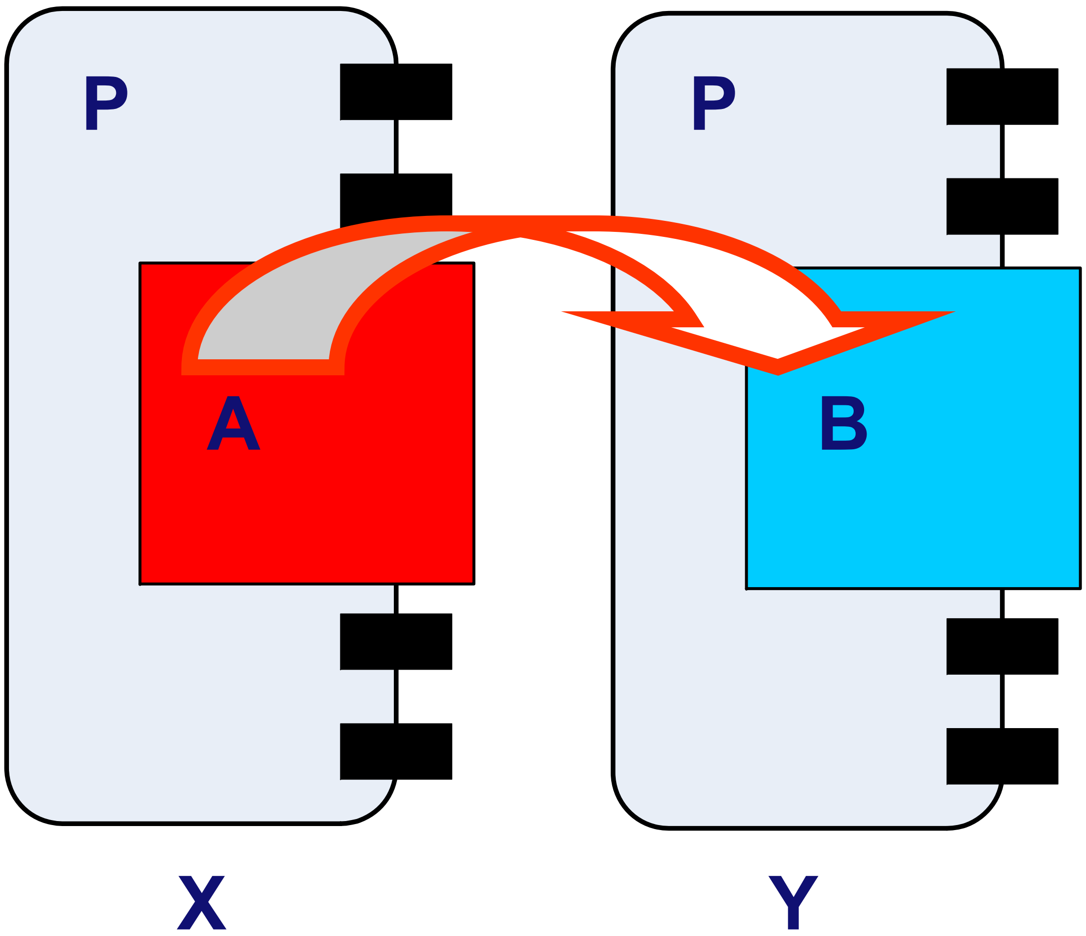
Definition 22 (Refinement of OGs). *Let N and N' be interface equivalent open nets and let $OG(N) = [Q, MP, \delta, q_0, \Phi]$ and $OG(N') = [Q', MP', \delta', q_0', \Phi']$ be the corresponding operating guidelines. Then, $OG(N) \sqsubseteq OG(N')$ (i.e., $OG(N')$ refines $OG(N)$) iff there is a simulation relation $\xi \subseteq Q \times Q'$ such that for all $[q, q'] \in \xi$, the formula $\Phi(q) \Rightarrow \Phi'(q')$ is a tautology.*

Theorem 2 (Checking accordance [32]). *Let N and N' be two open nets and let $OG(N)$ and $OG(N')$ be the corresponding operating guidelines. Then, $OG(N) \sqsubseteq OG(N')$ iff $Strat(N) \subseteq Strat(N')$.*

Accordance can be checked using operating guidelines!

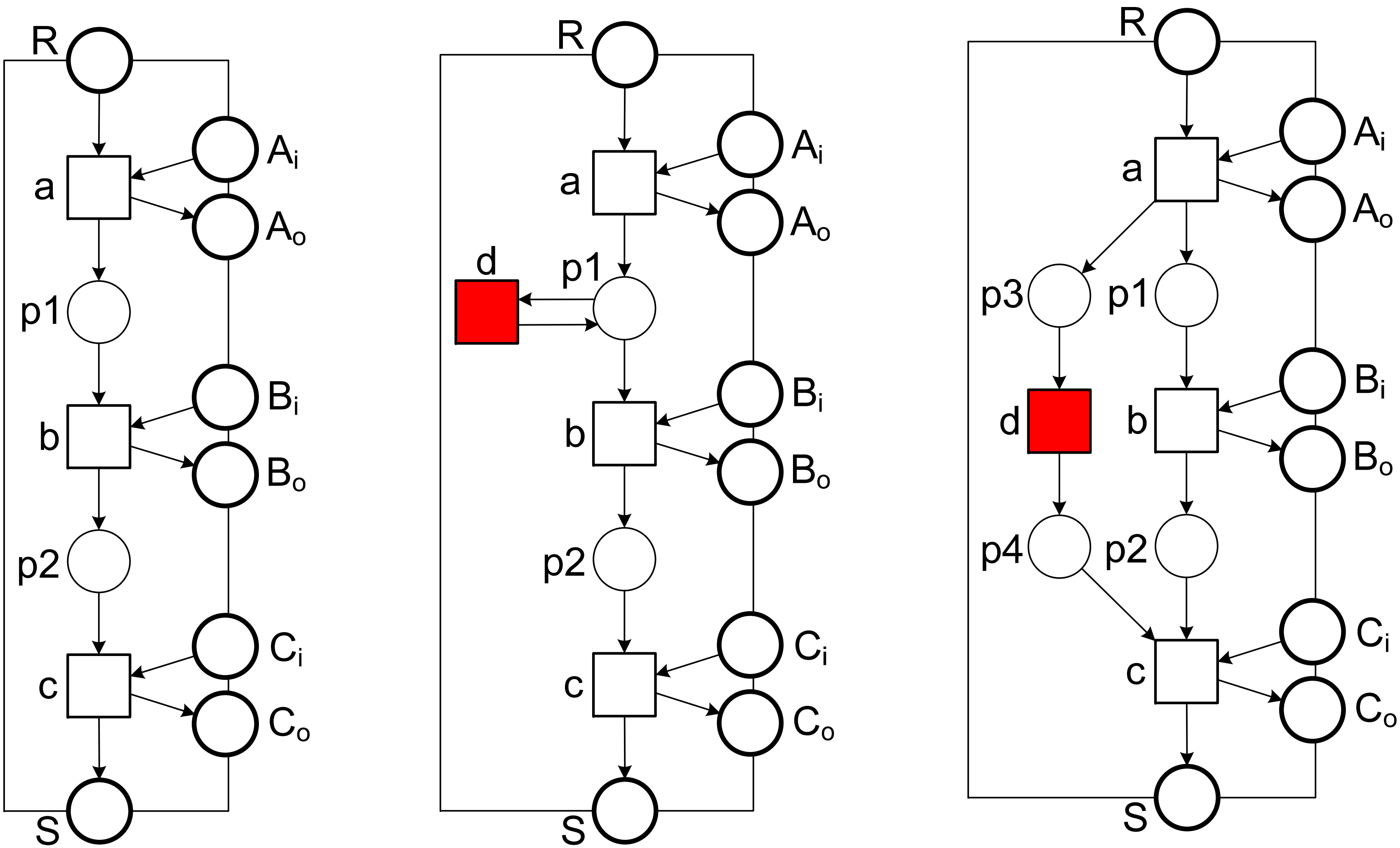
Details not important at this stage.

Transformation Rules

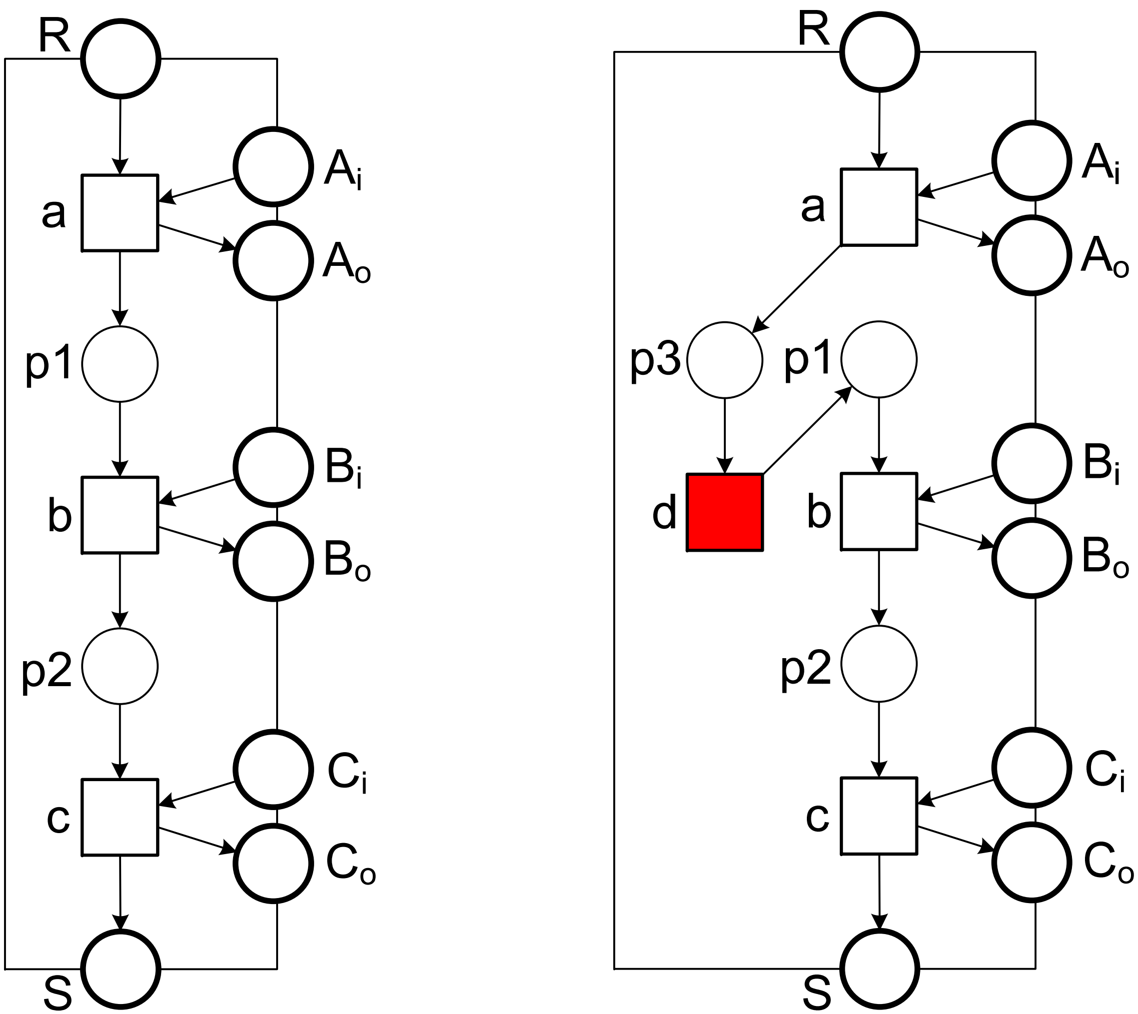


Strat(X) = Strat(Y)

Inheritance Preserving Transformation Rules (1/2)



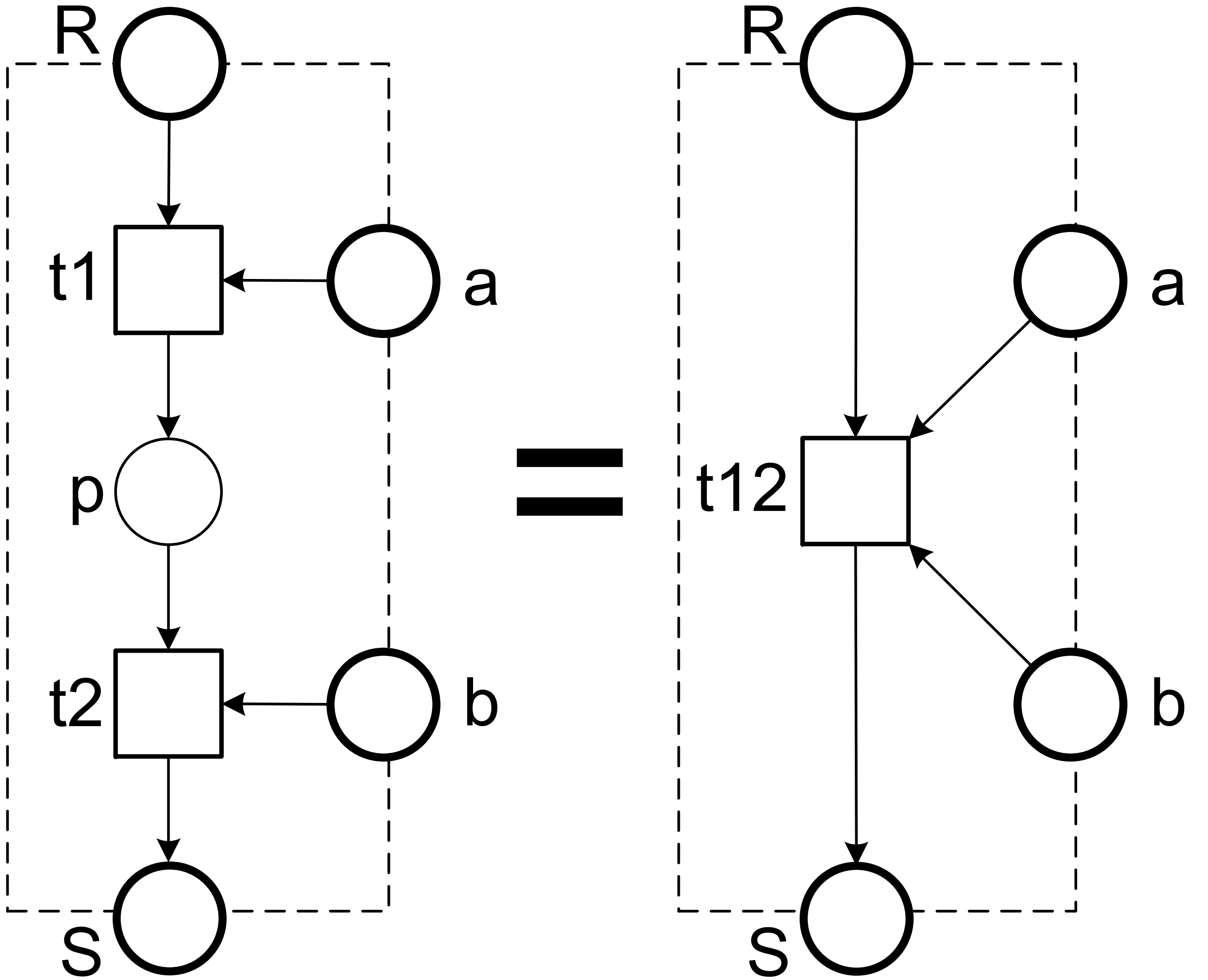
Inheritance Preserving Transformation Rules (2/2)



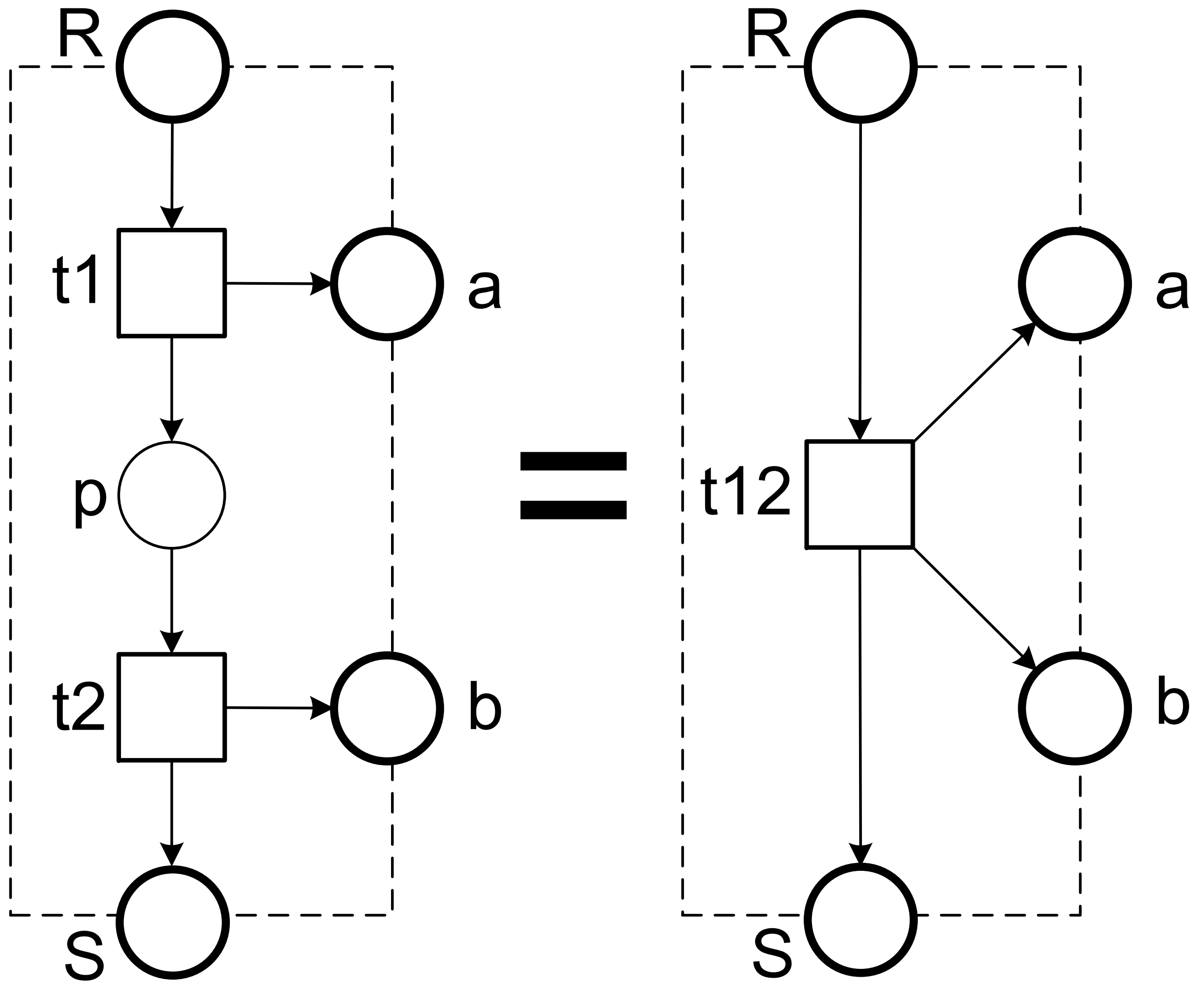
Inheritance preserving transformation rules also preserve accordance!

But are too strong ...

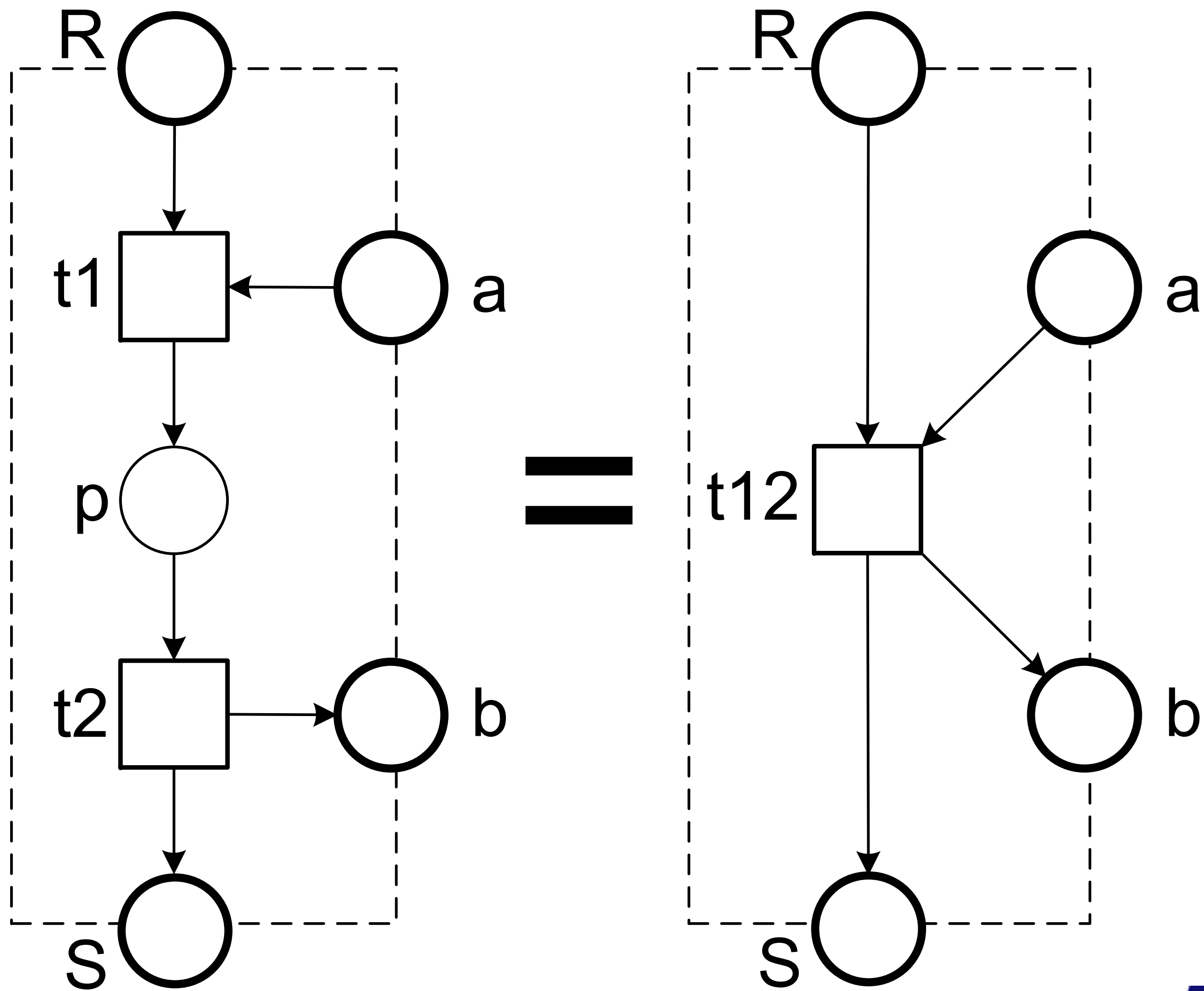
Accordance Preserving Transformation Rule 1



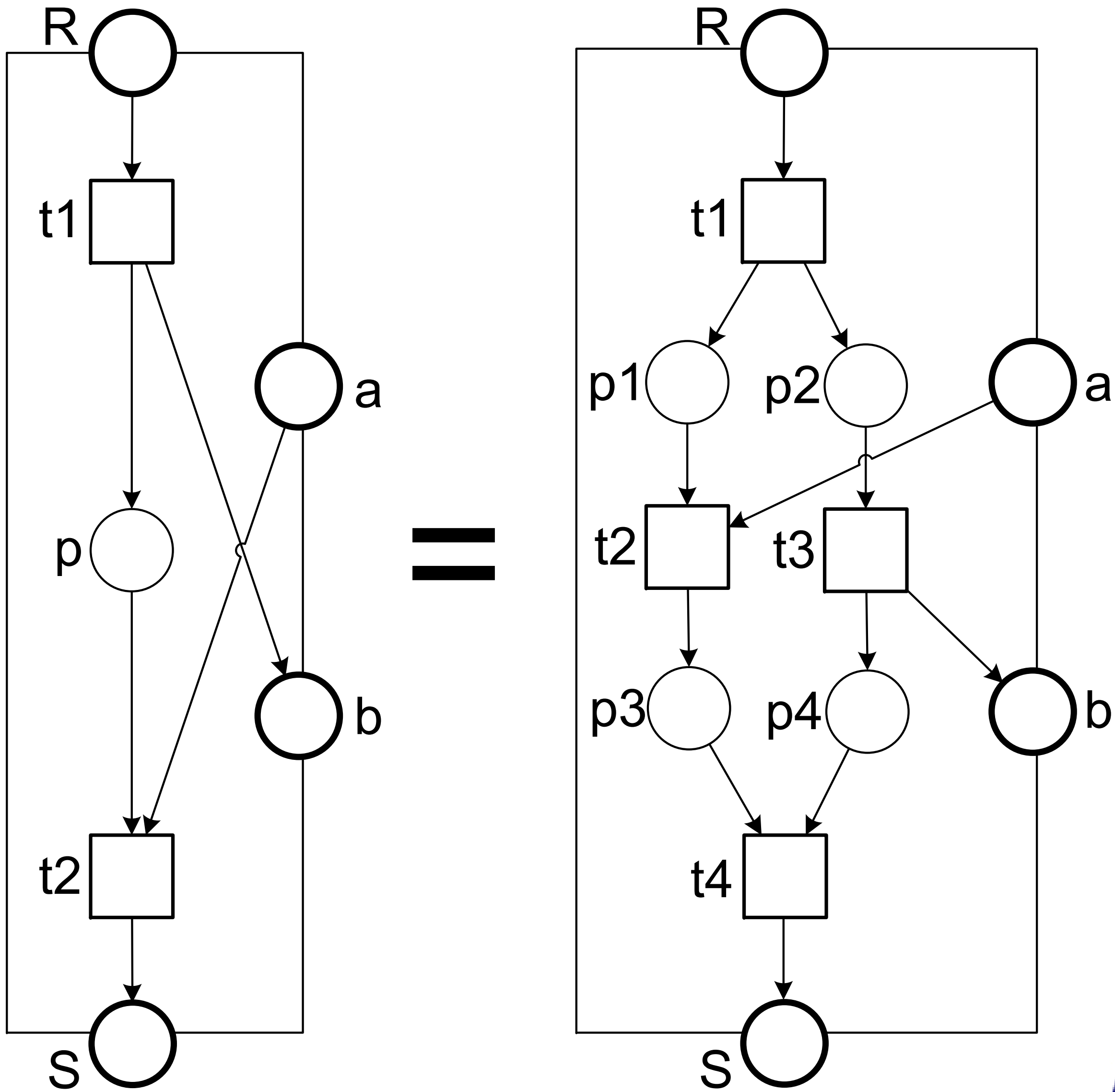
Accordance Preserving Transformation Rule 2



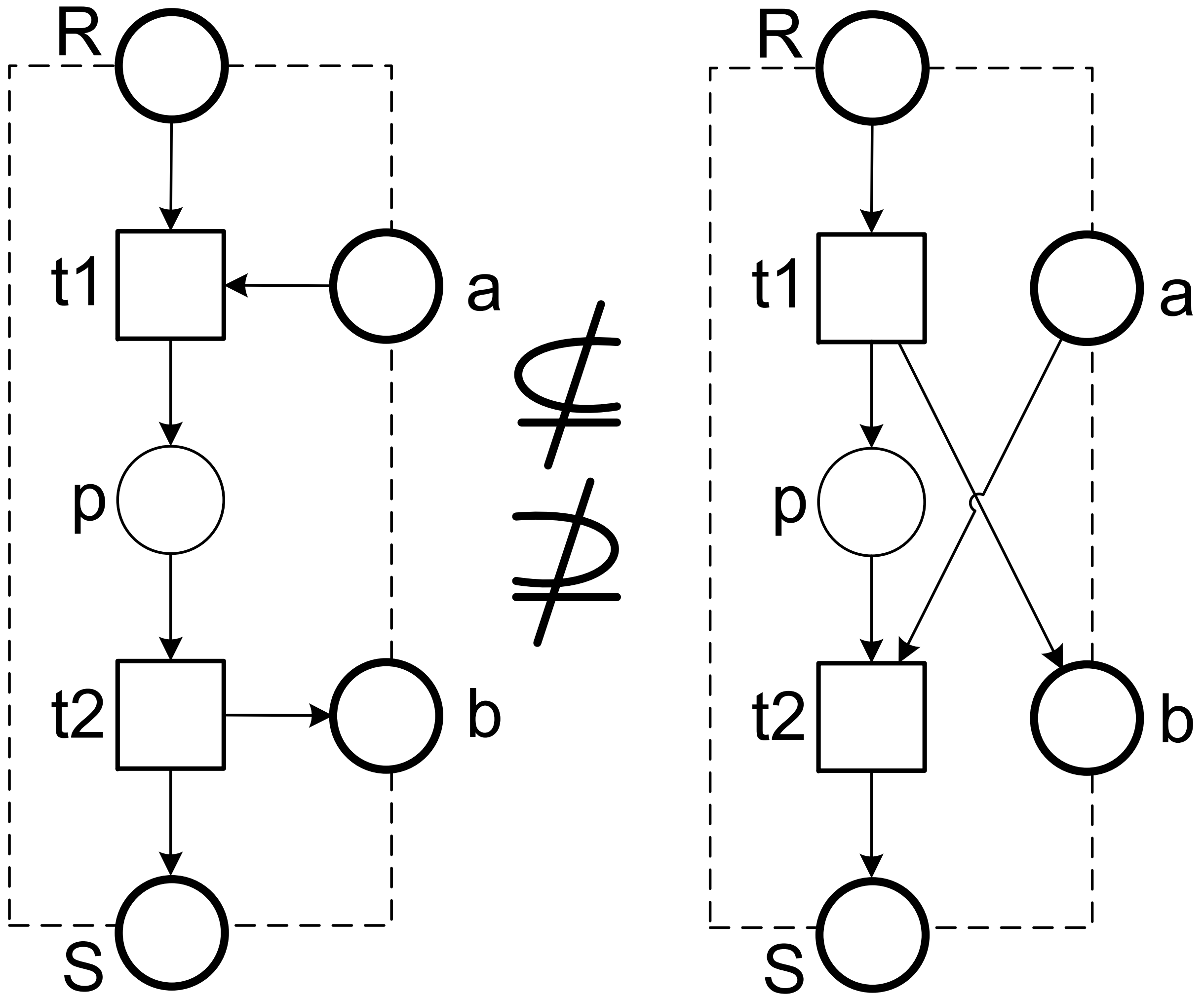
Accordance Preserving Transformation Rule 3



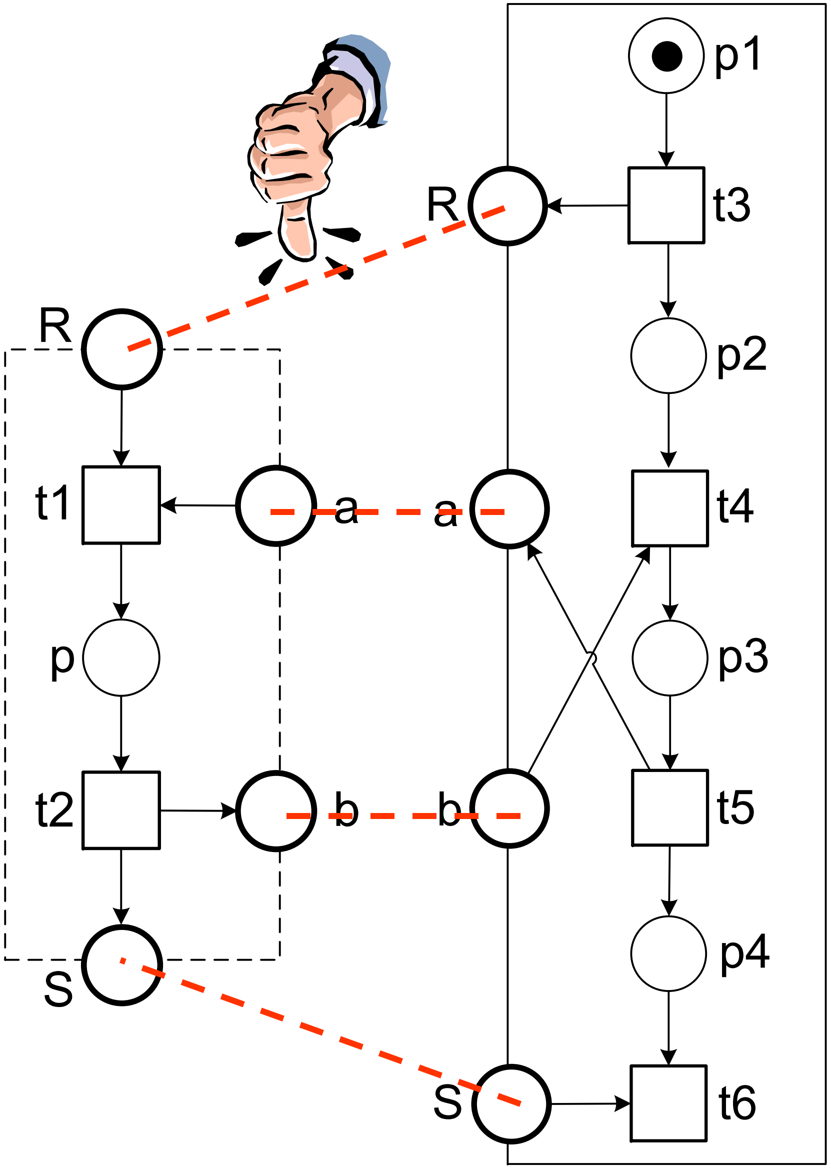
Accordance Preserving Transformation Rule 4

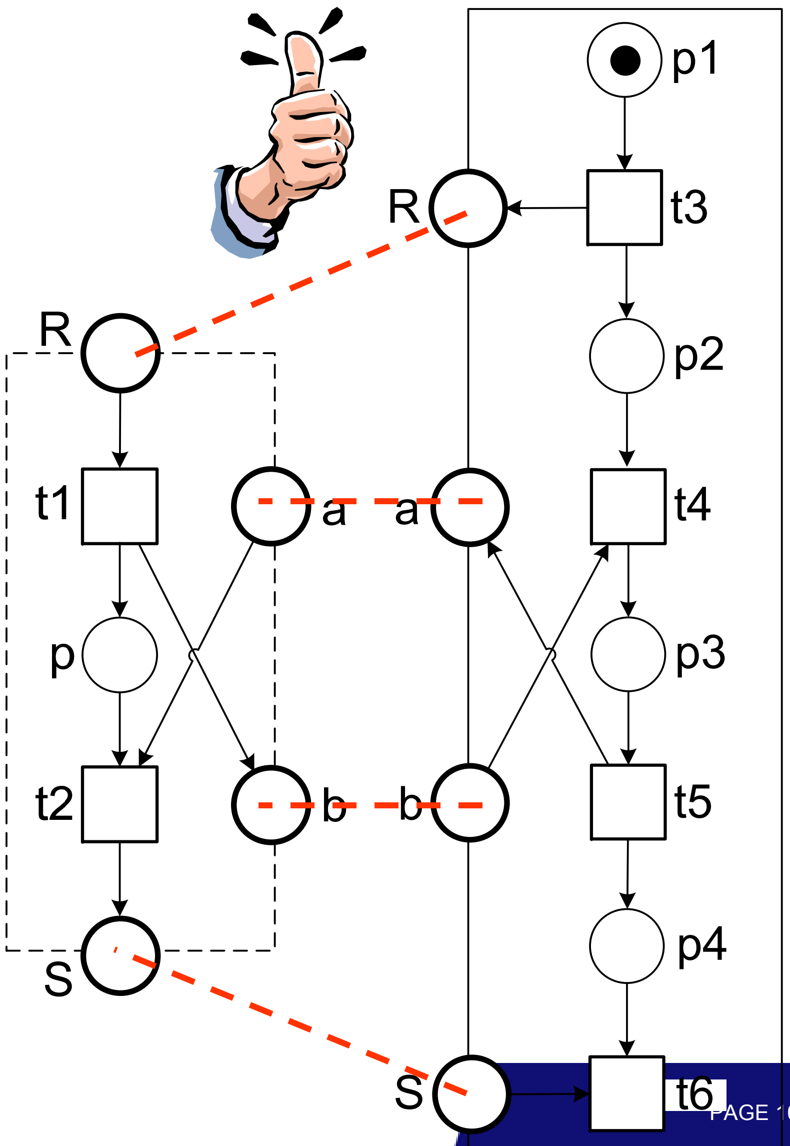


Another Type of Anti Pattern

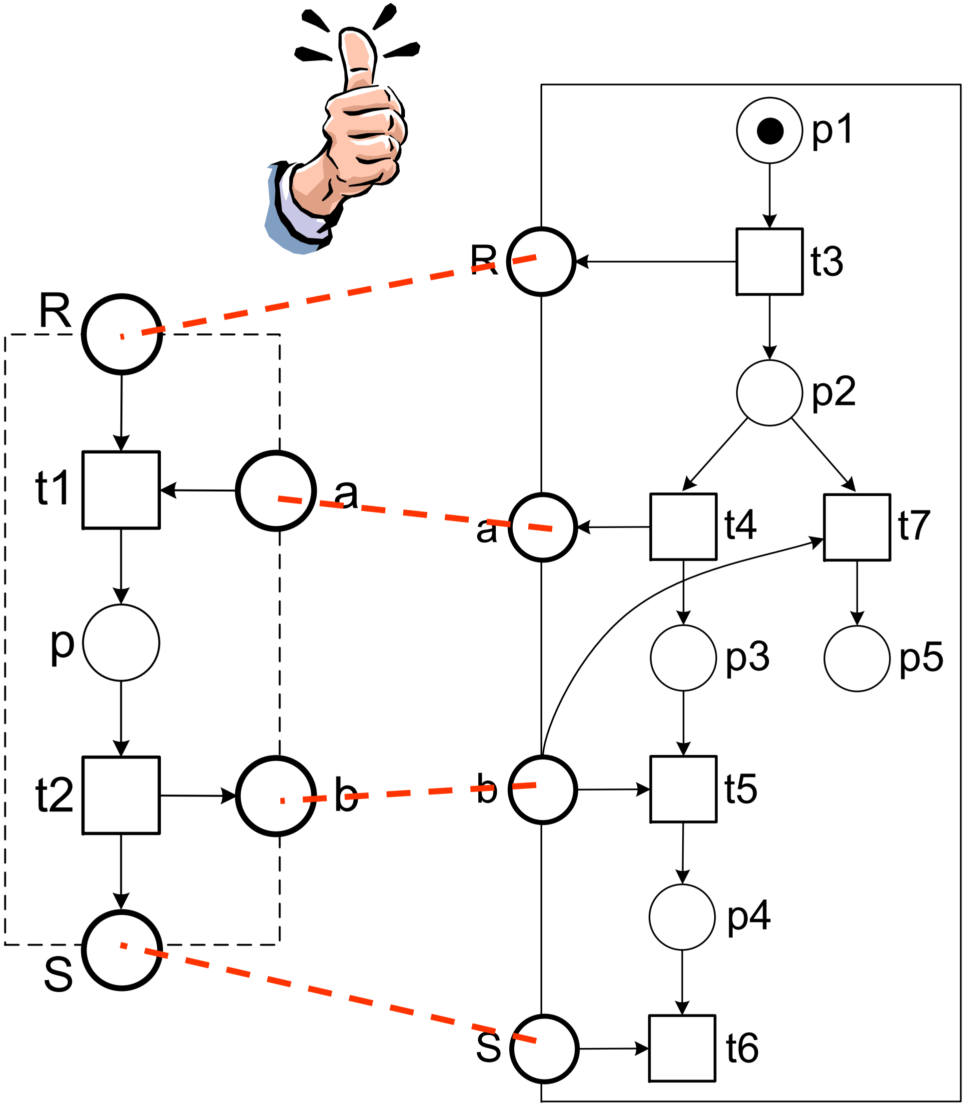


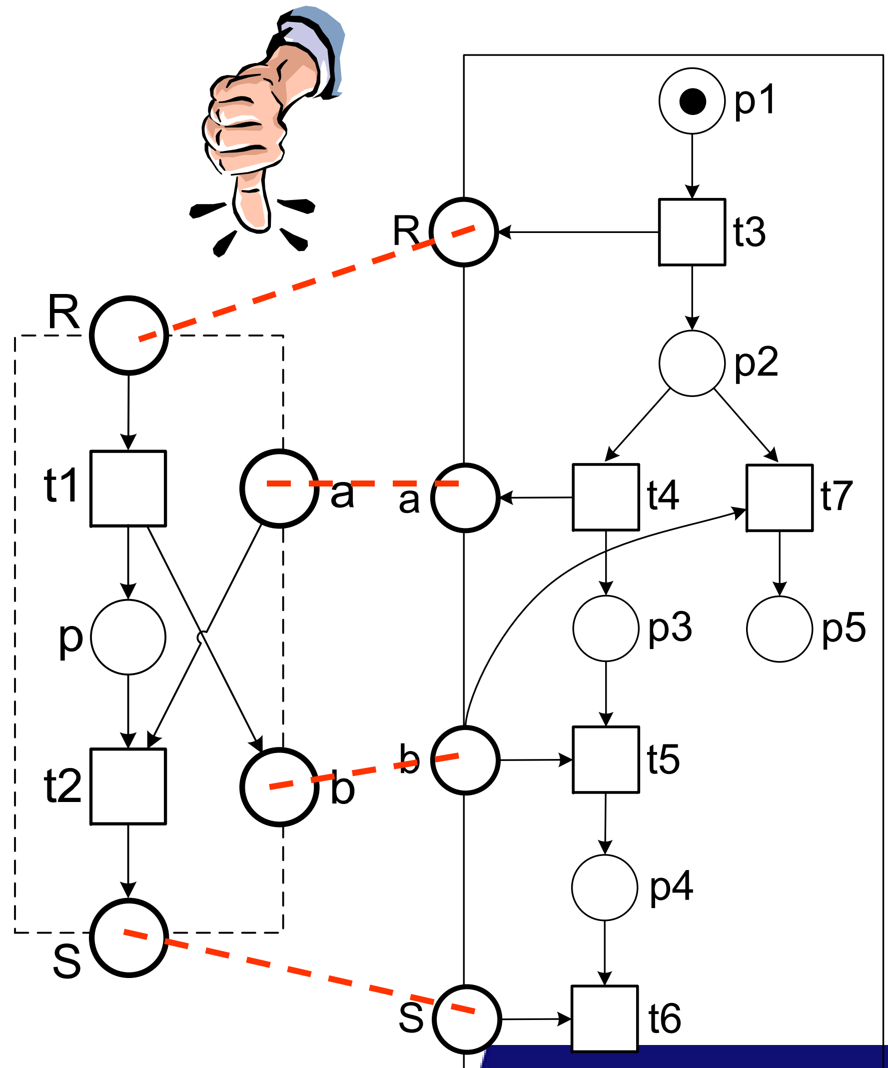
Strategy for one net and not the other



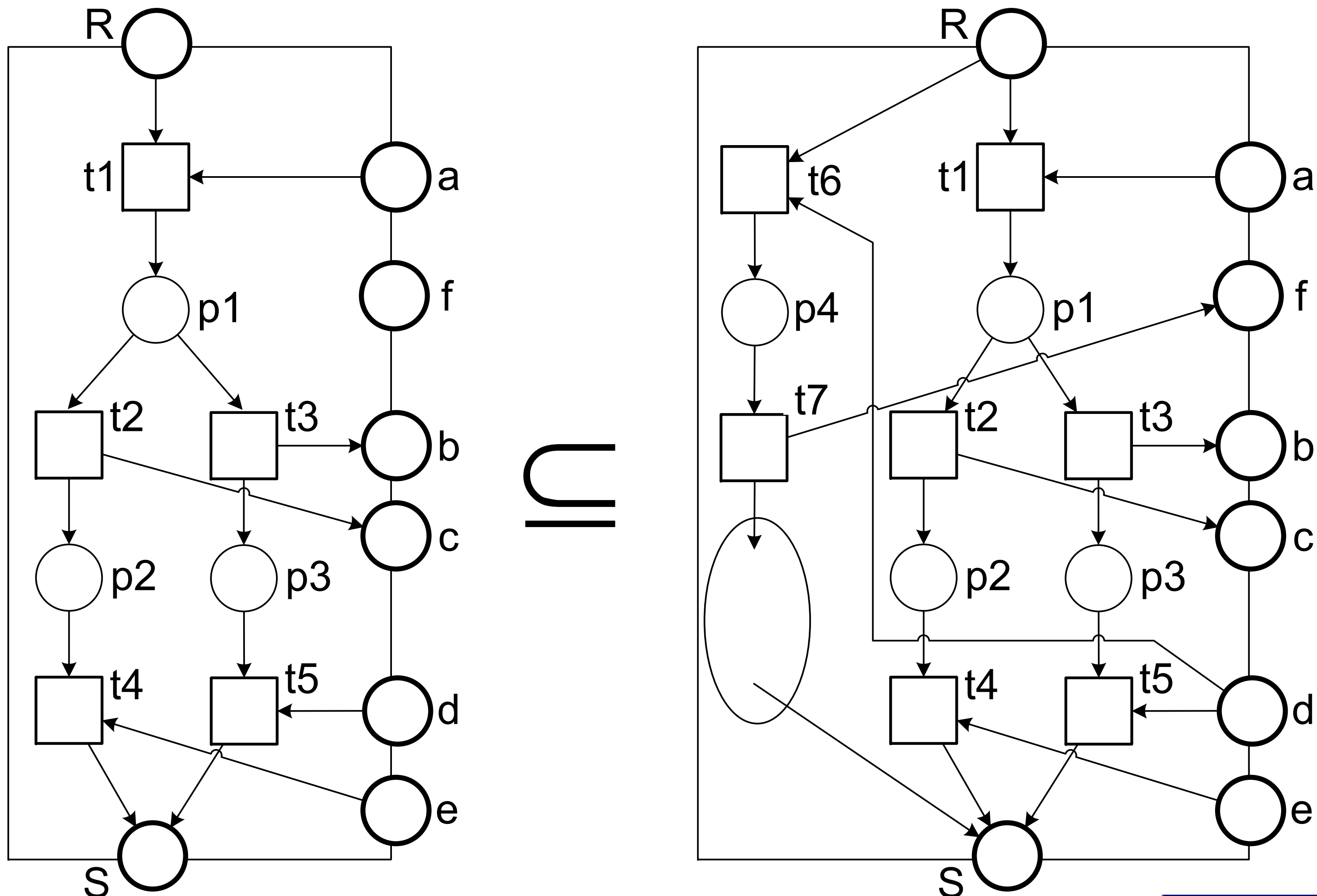


Strategy for one net and not the other





Accordance Preserving Transformation Rule 5



Recommended Reading

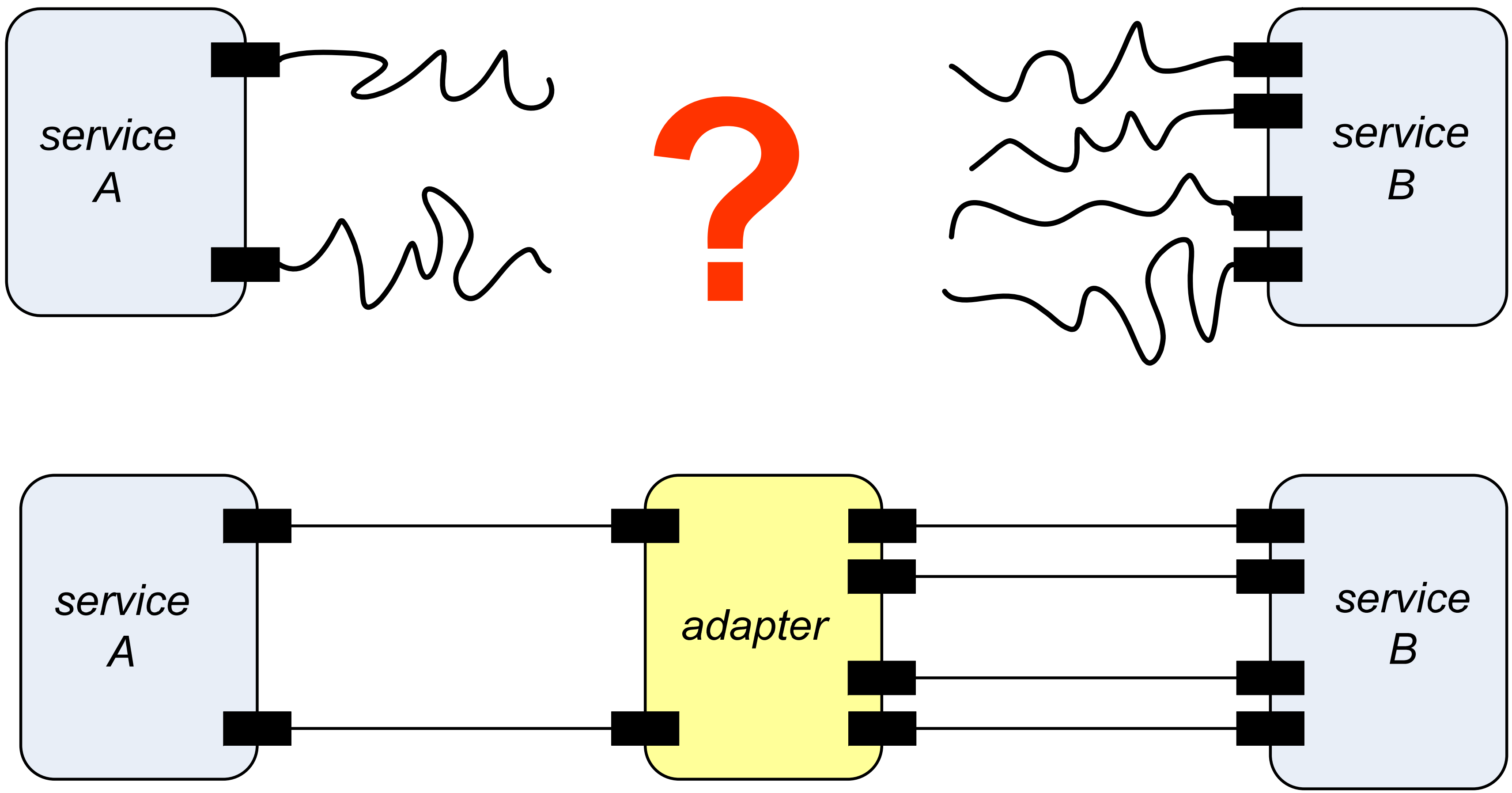
- van der Aalst, W., Mooij, A.J., Stahl C., Wolf, K.. Service Interaction: Patterns, Formalization, and Analysis. In SFM 2009, volume 5569 of Lecture Notes in Computer Science, pages 42-88. Springer-Verlag, Berlin, (2009)
- van der Aalst, W., Lohmann, N., Massuthe, P., Stahl, C., Wolf, K.: From Public Views to Private Views: Correctness-by-Design for Services. In: Dumas, M., Heckel, H. (eds.) WS-FM 2007. LNCS, vol. 4937, pp. 139–153. Springer, Heidelberg (2008)
- Basten, T., Aalst, W.: Inheritance of Behavior. Journal of Logic and Algebraic Programming 47(2), 47–145 (2001)
- van der Aalst, W.M.P., Basten, T.: Inheritance of Workflows: An Approach to Tackling Problems Related to Change. Theoretical Computer Science, 270(1-2):125-203 (2002)

Integrating Services Using Adapters

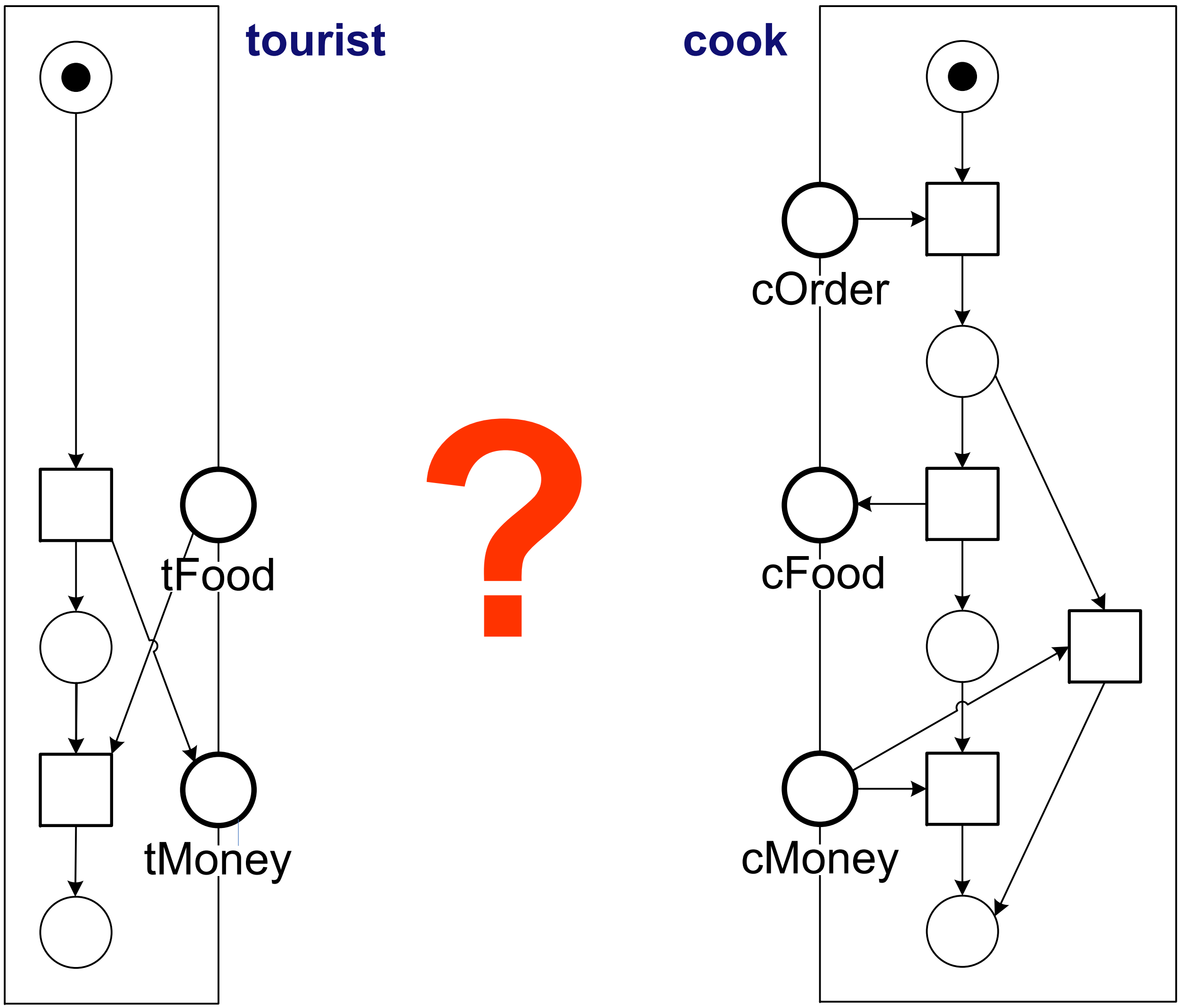
Technische Universiteit
Eindhoven
University of Technology

Where innovation starts

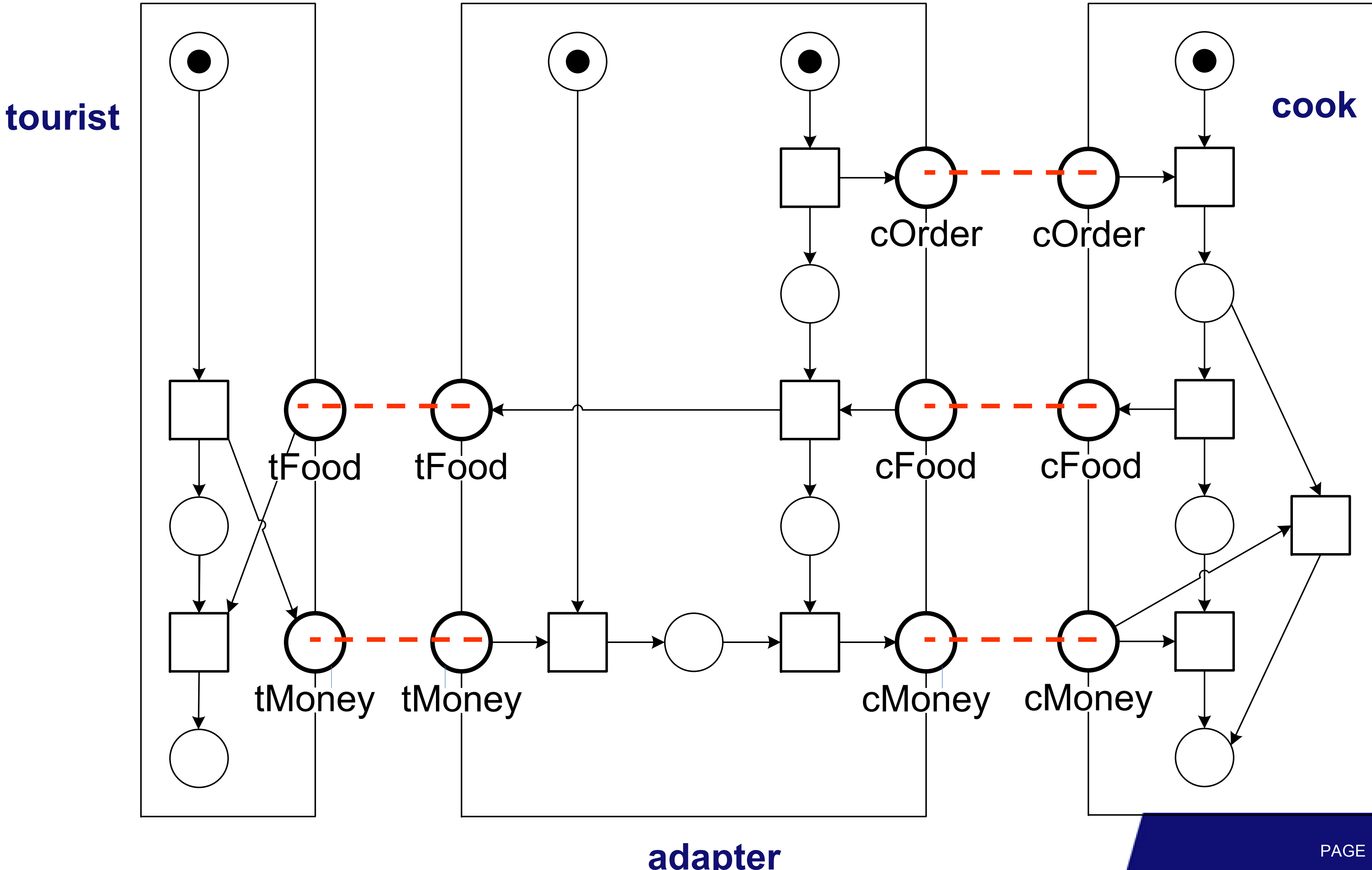
The Need For Adapters



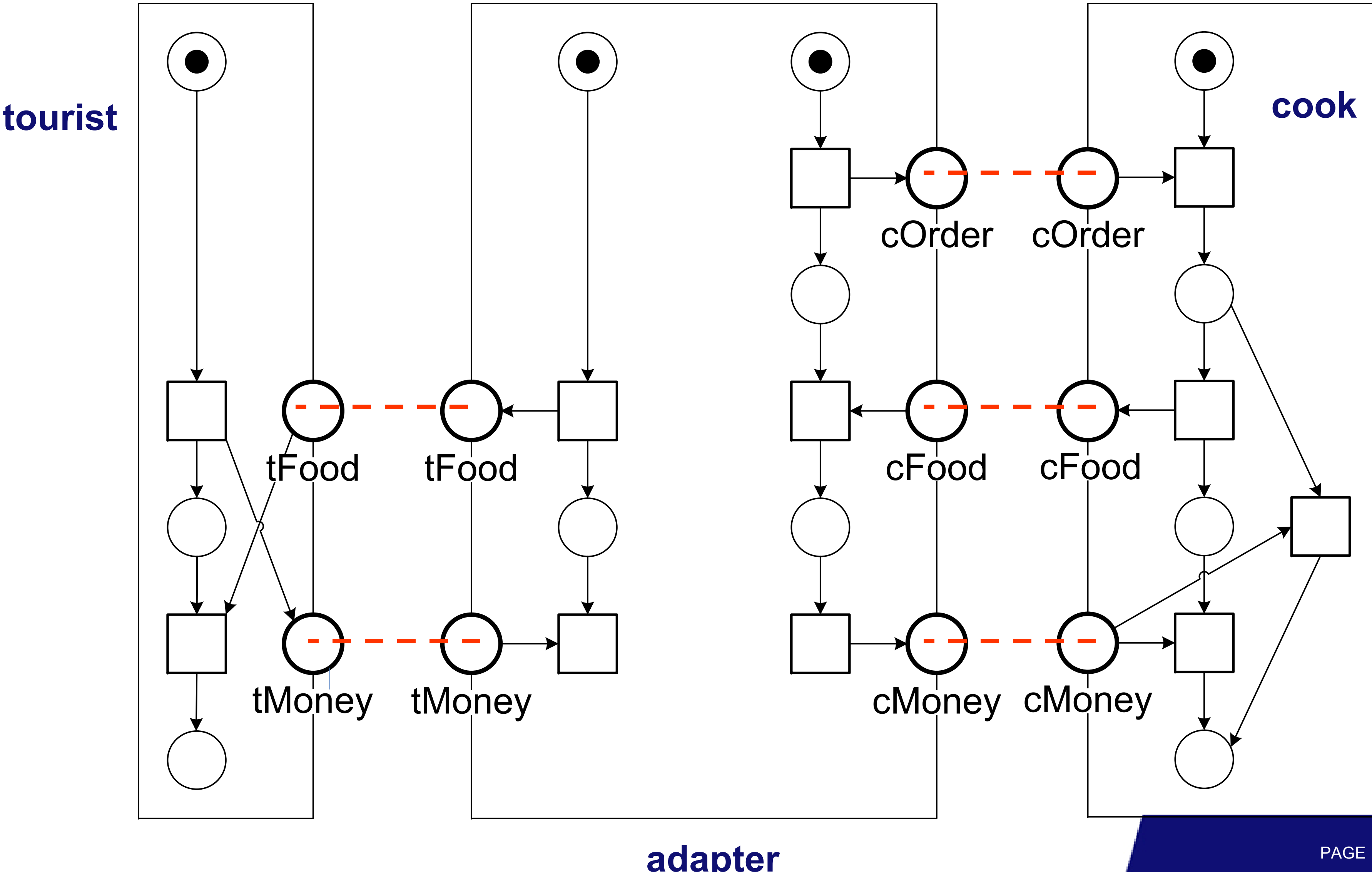
Example



Adapter


tourist
cook
cOrder
cOrder
tFood
tFood
cFood
cFood
tMoney
tMoney
cMoney
cMoney
adapter

Adapter ?????


tourist
cook
cOrder
cOrder
tFood
tFood
cFood
cFood
tMoney
tMoney
cMoney
cMoney
adapter

Specification of Elementary Activities (SEA)

- **To avoid creating money, deleting evidence, confusing meters with liters, etc.**

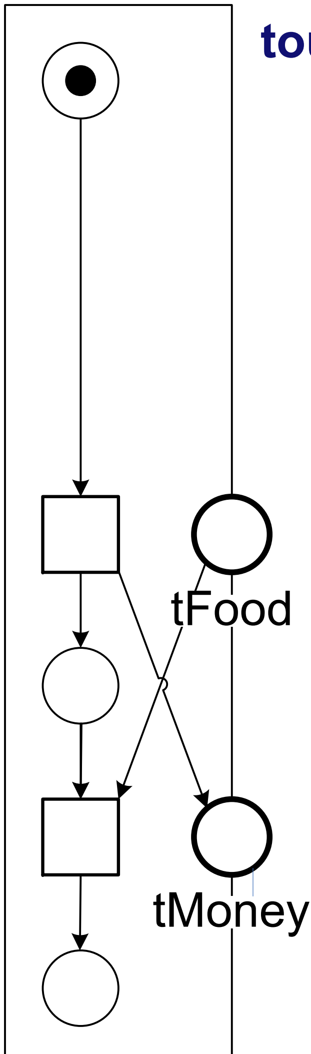
Elementary activity	Possible transformation rule
Create a	$\mapsto \ a$
Copy a	$a \ \mapsto \ a, a$
Delete a	$a \ \mapsto$
Transform a, b, c into d, e	$a, b, c \ \mapsto \ d, e$ or $a, b, c \ \mapsto \ a, b, c, d, e$
Split a into b, c, d	$a \ \mapsto \ b, c, d$ or $a \ \mapsto \ a, b, c, d$
Merge a, b, c into d	$a, b, c \ \mapsto \ d$ or $a, b, c \ \mapsto \ a, b, c, d$

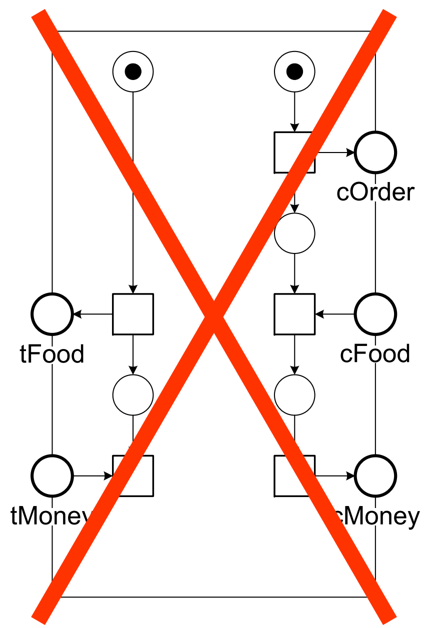
SEA Example

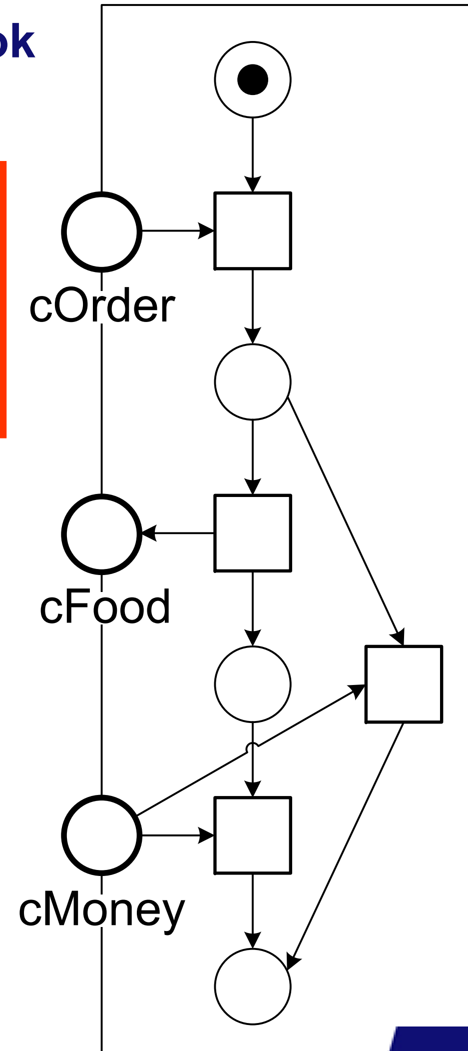
tourist

cook

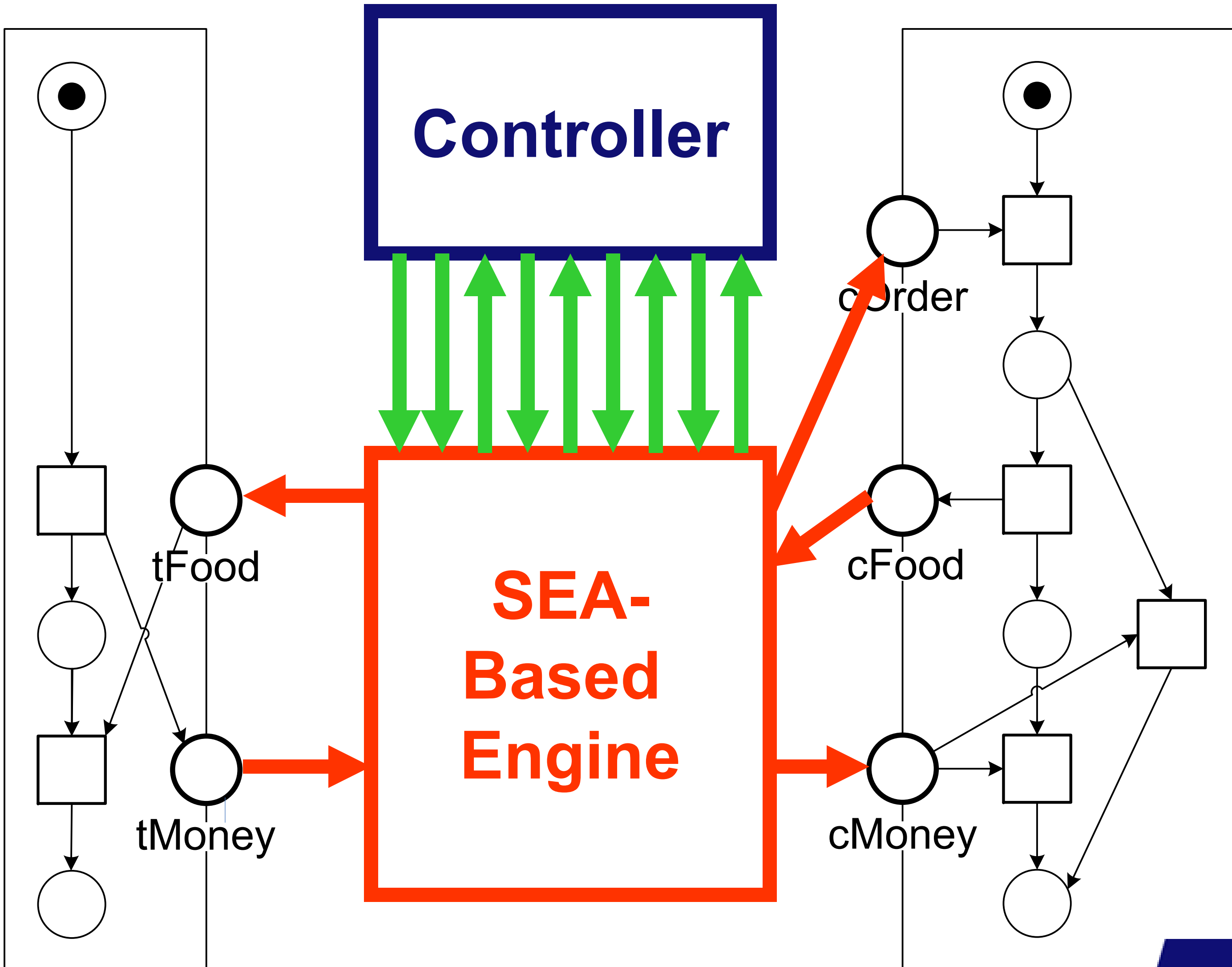
$$\begin{array}{lll} & \mapsto & \mathrm{cOrder} \\ \mathrm{cFood} & \mapsto & \mathrm{tFood} \\ \mathrm{tMoney} & \mapsto & \mathrm{cMoney} \end{array}$$





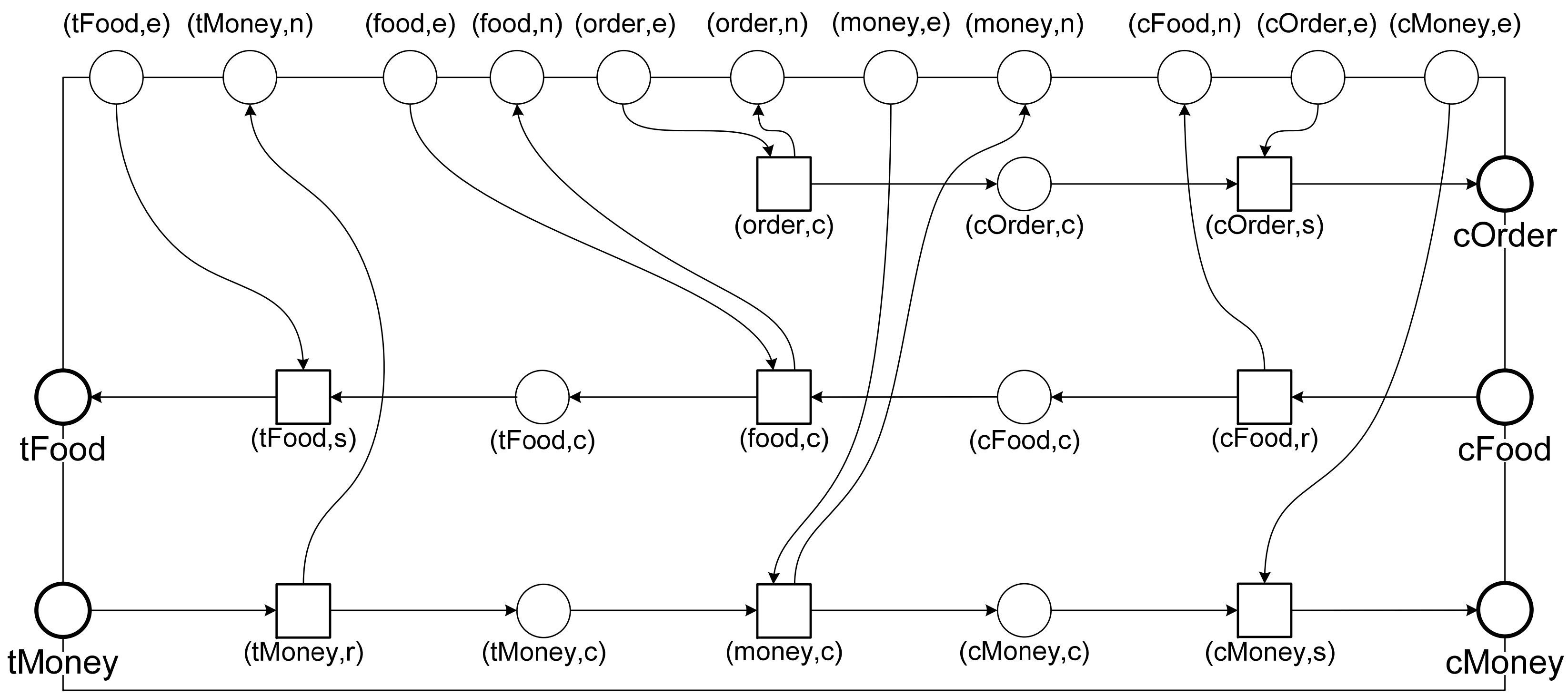


Overall Idea



SEA-Based Engine

n = notify
e = enable



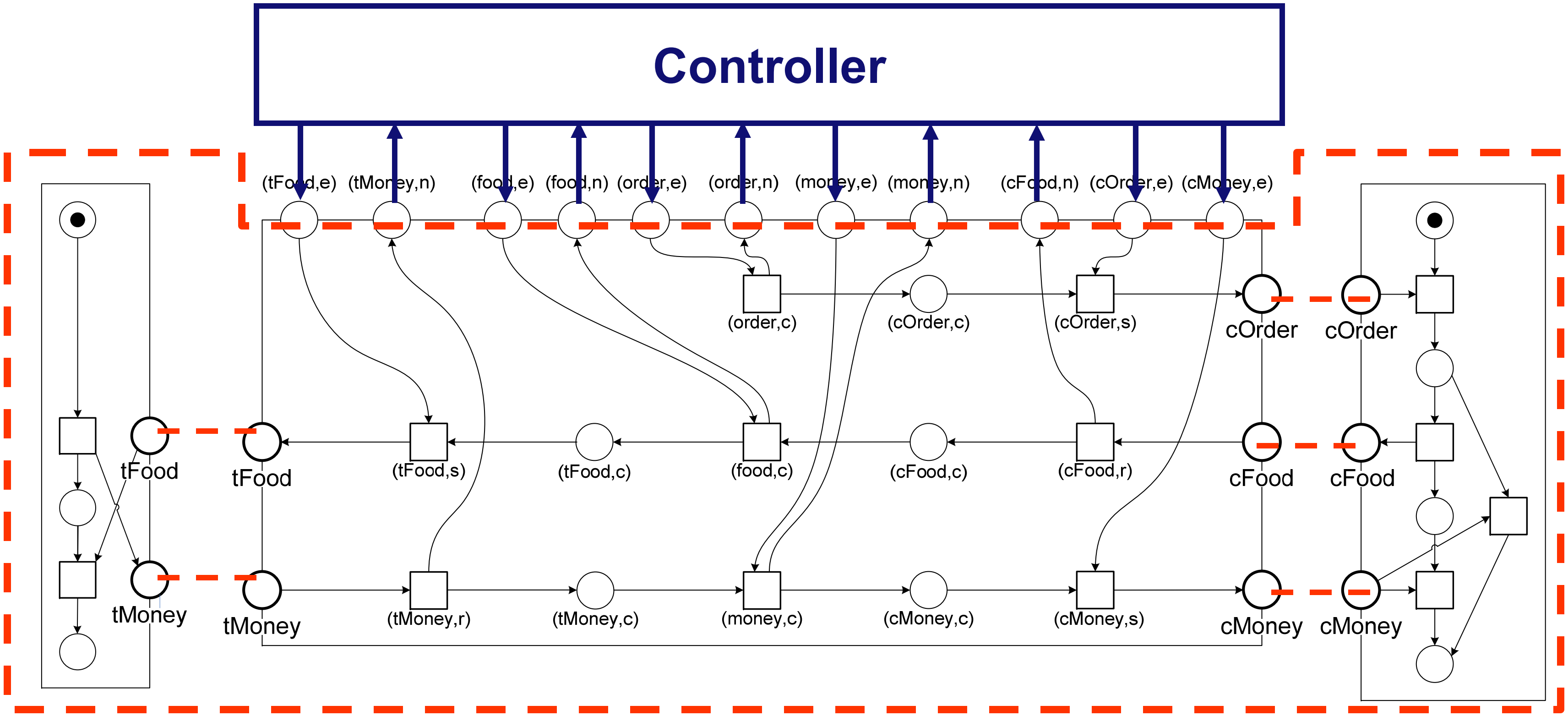
	$\mapsto$	cOrder
cFood	$\mapsto$	tFood
tMoney	$\mapsto$	cMoney

s=send
r=receive
c=actual transformation

n = notify
e = enable

Business As Usual ...

- Selecting a controller is like selecting a strategy.
- One approach is to construct "the" most permissive one



Recommended Reading

- van der Aalst, W., Mooij, A.J., Stahl C., Wolf, K.. Service Interaction: Patterns, Formalization, and Analysis. In SFM 2009, volume 5569 of Lecture Notes in Computer Science, pages 42-88. Springer-Verlag, Berlin, 2009.
- Gierds, C., Mooij, A., Wolf, K.: Specifying and generating behavioral service adapters based on transformation rules. Preprints CS-02-08, Institut fur Informatik, Universität Rostock (2008)
- Mooij, A., Voorhoeve, M.: Proof techniques for adapter generation. In: Proc. WSFM (2008)

Service Mining

TU/e Technische Universiteit
Eindhoven
University of Technology

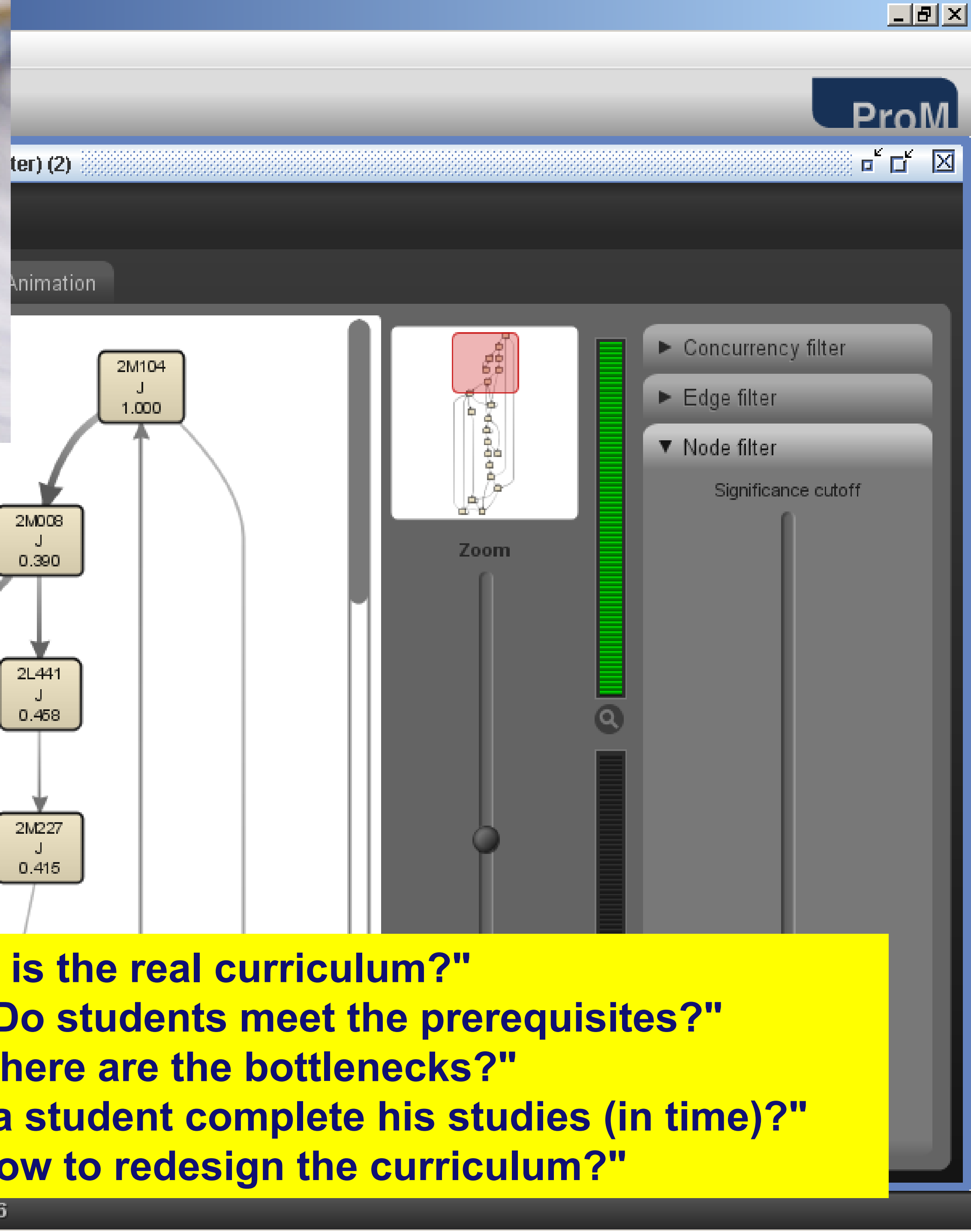
Where innovation starts


Correctness at "model-time" is irrelevant!


ProM [5.0]
File Mining Analysis Conversion Exports Window Help
ProM
Results - Fuzzy Miner on Raw MiningBronbestand_Goed_Ontology.mxml (unfiltered)
Fuzzy Model Toolkit
Transformer Editor Unary Metrics Binary Metrics Animation
Concurrency filter
Edge filter
Node filter
Significance cutoff
Zoom
08:49:29 [M] Measured fuzzy conformance is 0.6839424087761667

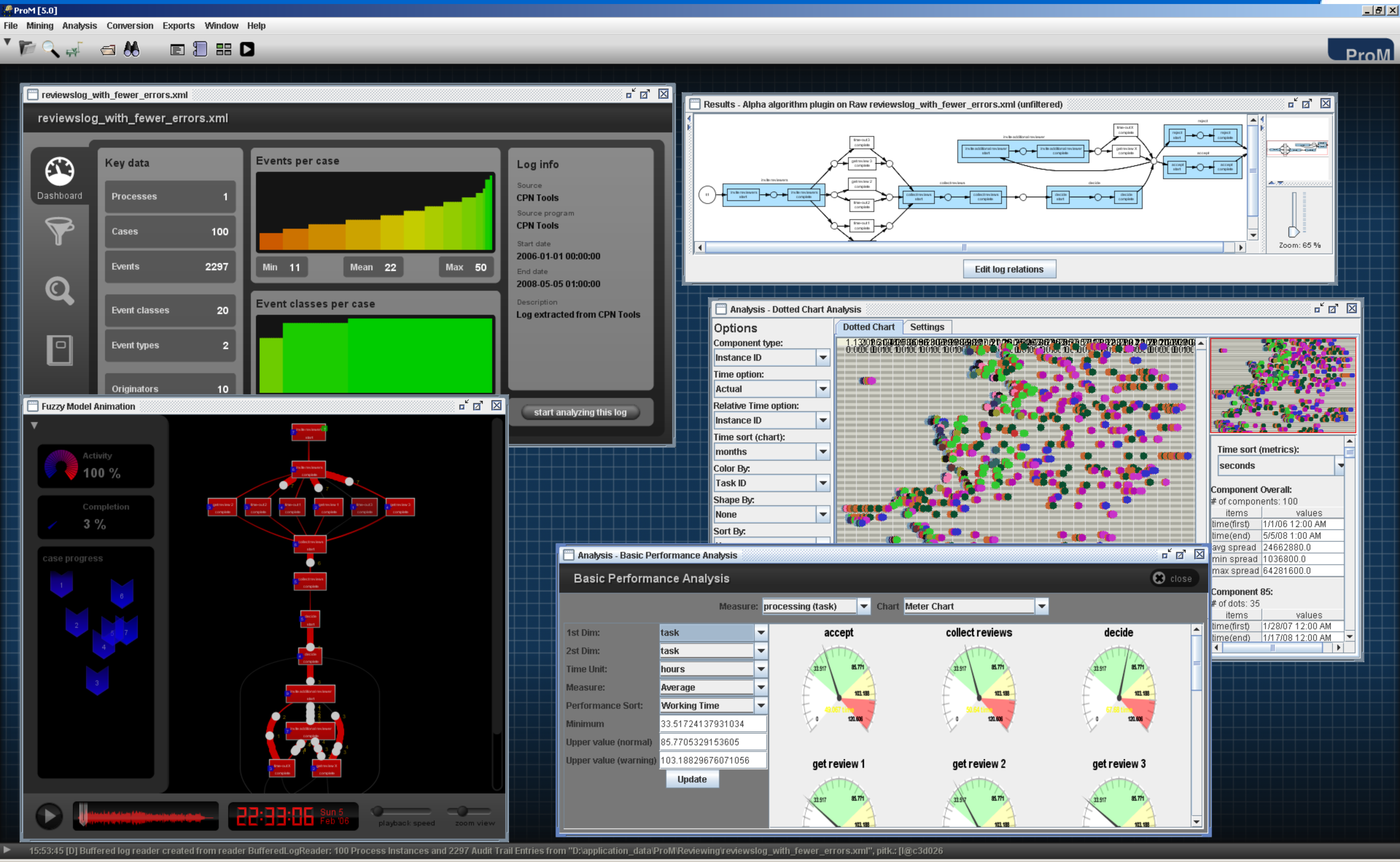
Process Mining

- Process discovery: "What is really happening?"
- Conformance checking: "Do we do what was agreed upon?"
- Performance analysis: "Where are the bottlenecks?"
- Process prediction: "Will this case be late?"
- Process improvement: "How to redesign this process?"
- Etc.

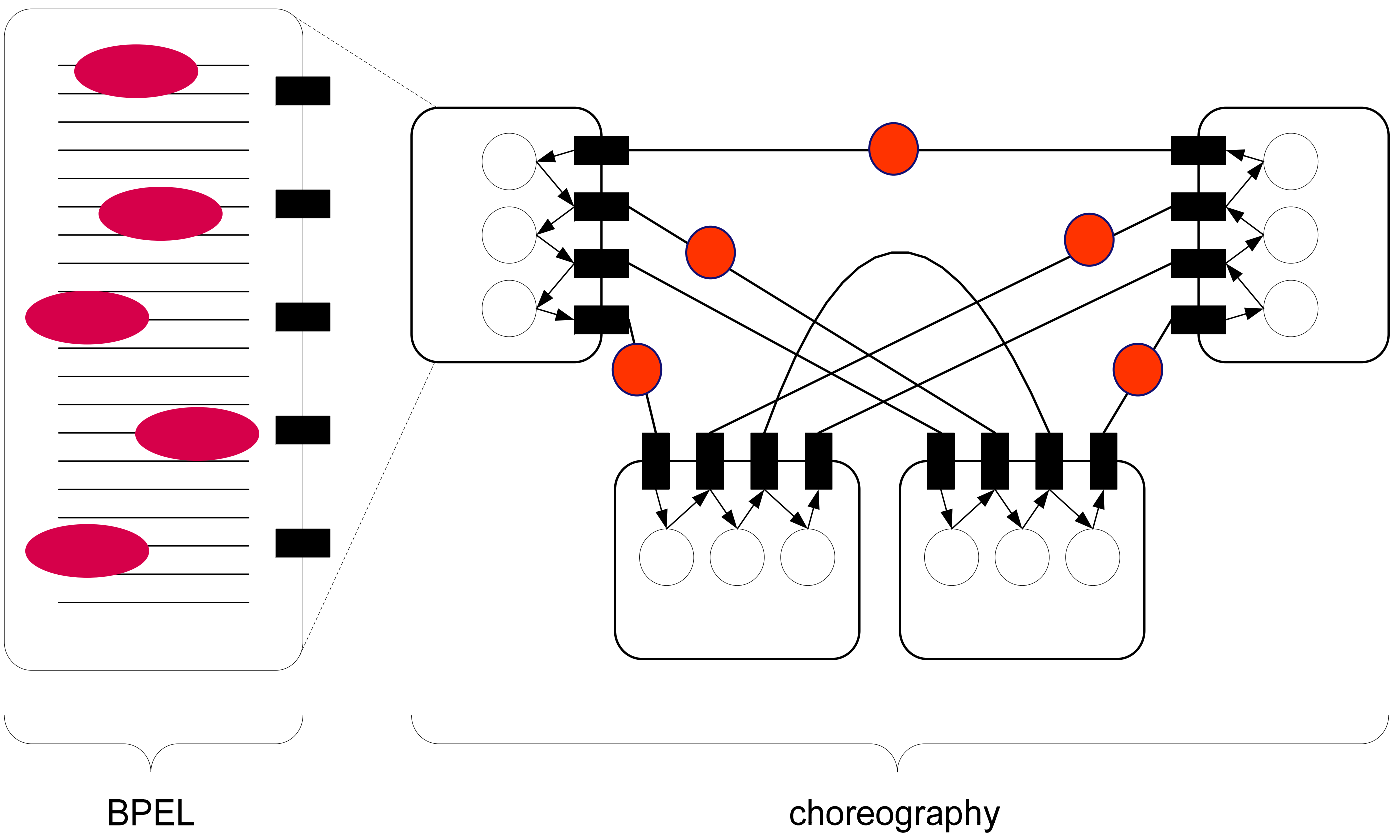


- **Process discovery: "What is the real curriculum?"**
- **Conformance checking: "Do students meet the prerequisites?"**
- **Performance analysis: "Where are the bottlenecks?"**
- **Process prediction: "Will a student complete his studies (in time)?"**
- **Process improvement: "How to redesign the curriculum?"**

Screenshot of ProM 5.0



Example Setting for Service Mining



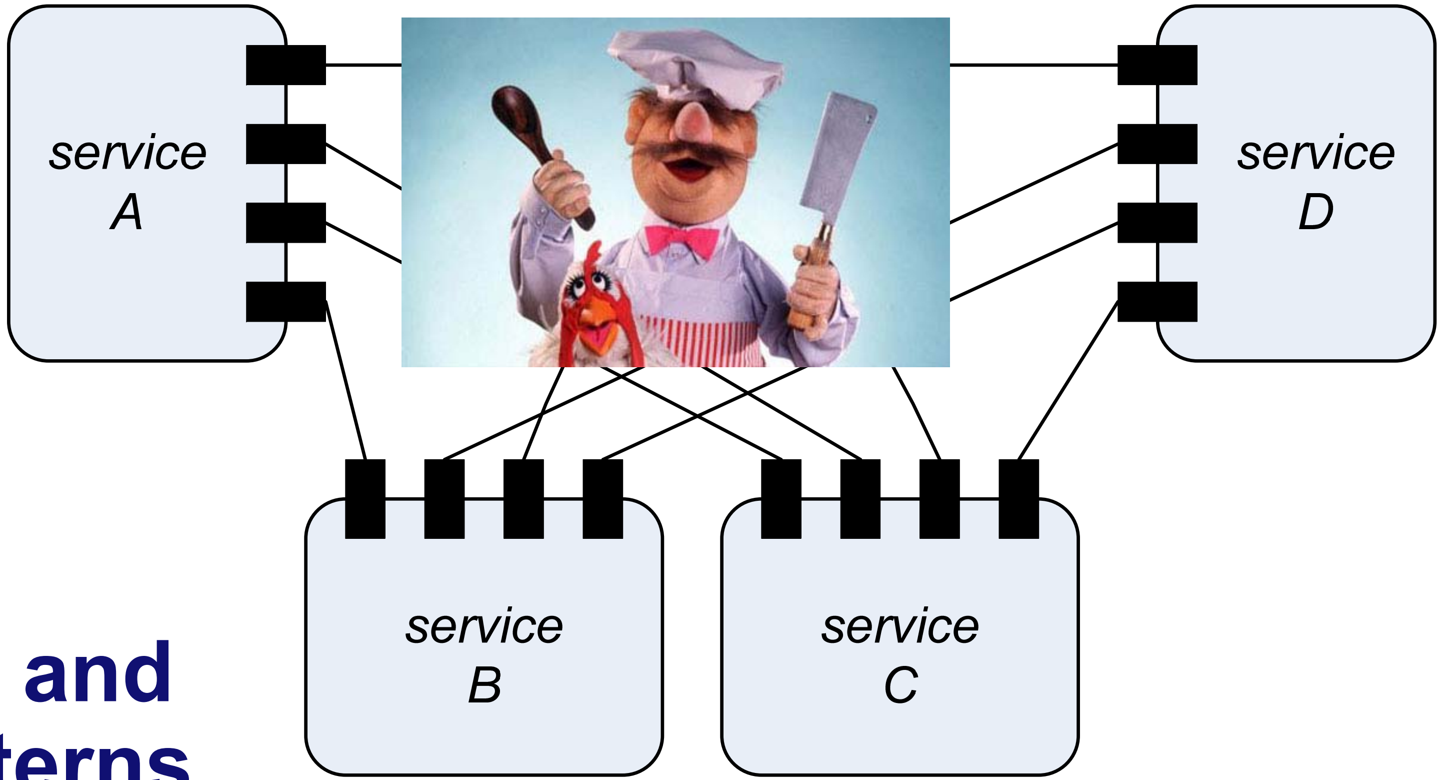
ProM
process mining workbench

Recommended Reading

- van der Aalst, W.M.P., Dumas, M., Ouyang, C., Rozinat, A., Verbeek, H.M.W.: Conformance Checking of Service Behavior. ACM Transactions on Internet Technology, 8(3):29-59, 2008.
- Rozinat, A., van der Aalst, W.M.P.: Conformance Checking of Processes Based on Monitoring Real Behavior. Information Systems, 33(1):64-95, 2008.
- van der Aalst, W.M.P., Reijers, H.A., Weijters, A.J.M.M., van Dongen, B.F., Alves de Medeiros, A.K., Song, M., Verbeek, H.M.W.: Business Process Mining: An Industrial Application. Information Systems, 32(5):713-732, 2007.
- www.processmining.org

Conclusion

Service Interaction Demystified



- **Patterns and Anti-Patterns**
- **Formalization**
- **Analysis**

Questions Addressed

1. Exposing Services
   - How to inform others about me such that cooperation is possible?
   - Two approaches: (a) expose own behavior and (b) provide operating guideline.
2. Replacing and Refining Services
   - How to replace or refine a service without introducing problems?
   - Inheritance, accordance, transformation rules, etc.
3. Integrating Services Using Adapters
   - How to resolve behavioral incompatibilities?
   - Adapter generation.
4. Service Mining
   - How to analyze the run-time behavior?

Relevant WWW sites

- http://www2.informatik.hu-berlin.de/top/best/
- http://www.service-technology.org
- http://www.workflowpatterns.com
- http://www.processmining.org
- http://promimport.sourceforge.net
- http://prom.sourceforge.net
- http://www.workflowcourse.com
- http://www.vdaalst.com