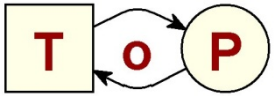
SUMMERSOC

Hersonissos, Wednesday, June 28, 2017. 9.30 – 10.30

Tutorial
Conceptual Foundations of SOC



Theory of Programming

Prof. Dr. W. Reisig

Wolfgang Reisig

Humboldt-Universität zu Berlin

Foundations of SOC

Why needed?

to make it conceptually simpler,

better teachable,

better usable by non-experts

What could this be?

Identify the (??) fundamental concepts

and build a theory on top of this …

We are so wonderfully progressing without

for a while

There is this deep “Theoretical Informatics” stuff. That’s enough.

No. There is something fundamentally new

Foundations of SOC

1. SOC exceeds classical Theoretical Informatics
2. Services are made to be composed!
3. Required: mechanisms to compose *many* services.
4. Composition must retain or guarantee *properties.*
5. Composition is surprisingly expressive!

Foundations of SOC

1. SOC exceeds classical Theoretical Informatics
2. Services are made to be composed!
3. Required: mechanisms to compose *many* services.
4. Composition must retain or guarantee *properties.*
5. Composition is surprisingly expressive!

Let’s start fundamentally …

What constitutes a science?

The example: Physics
A system of notions
and relations among them,
stating laws of nature,
described in terms of *models*.

Models in Science

Typical example:

The term “energy”
+ all laws about energy.

There is nothing like *energy* in nature.
The notion of “energy” is an *abstraction*
from many aspects in nature,

intended to describe an *invariant*.

Example: What remains invariant?

sum of energy

first residing in gasoline,

then in acceleration,

then in deformed metal sheet.

What physicists *really* did:

Searched a notion, general enough

to describe what remains invariant

… and called it *energy.*

Scientific theories

Physicist do accept intuitively hard models ("*theories*")
if they offer convincing explanations,
in particular *invariants*.

Invariant in Chemistry

$CH_4 + 2\ O_2 \rightarrow CO_2 + 2\ H_2O$

Search for good models
= Search for comprehensive invariants. $e = mc^2$

Even *Theoretical Biology* is behind (biological) invariants!
e.g. metabolism.
We should learn from this!

a further example: the Frank invariant

The sum of the length of his hair remains invariant

Our task

We must develop
integrated conceptual theories and models
in the style of the exact sciences
for … ??

… supporting nontrivial analysis and verification
by help of invariants.

Models and invariants in Informatics

Informatics has models.

Some models have invariants.

… with comprehensive invariants

What remains invariant
when using
a cash machine ?

account
+ in hand

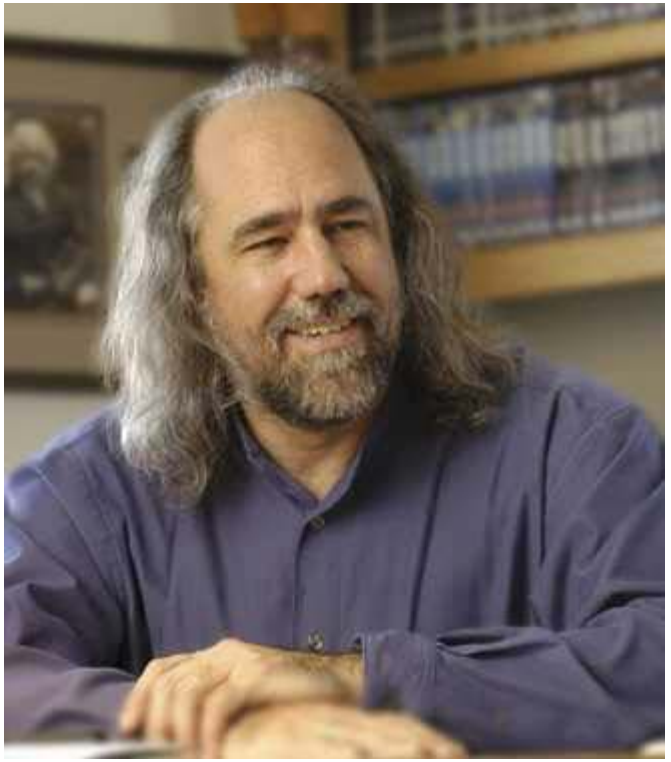
What remains invariant during running a garbage collector?

… while fulfilling a contract?

How achieve all this?

by means of models!

We must "*elevate models*
as to a first class citizenship ...
a peer of traditional text languages
(and potentially its master)".



"*models as products*".

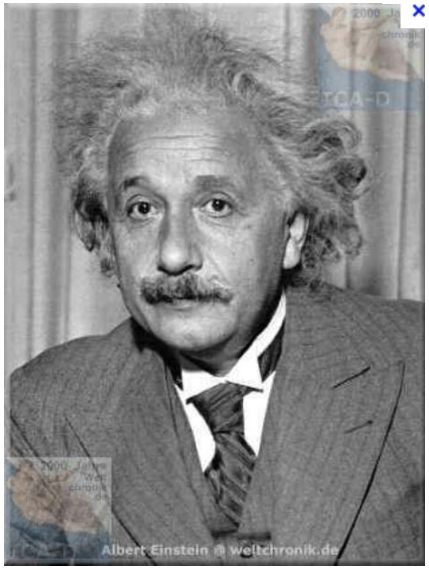
Grady Booch, (2004)

THE fundamental difference to programming:

1. Modeled behavior is not necessarily implemented.
2. The modeler freely chooses the level of abstraction

State of the art in informatics …

Certainly pre-Einstein



Probably, pre-Newton.

The three paradigms of programming:

1. Conventional (procedural) programs

follows the *IPO model:* input-process-output

theoretical foundation/ expressive power:

the computable functions, complexity theory, logic

2. Object orientation

attributes and methods

theoretical foundation:

abstract data types / algebraic specific

signatures and structures (as for 1st orc

3. Service orientation

self contained components

loosely coupled

theoretical foundation: missing

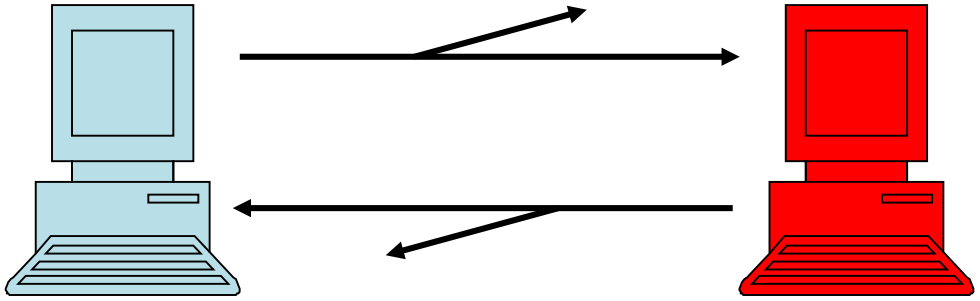
Three fundamental (?) aspects:

1. communication

2. non-ending behavior

3. causality

1.1 SOC comprises *communication*



How establish reliable communication?
By sending acknowledgements, copies, etc. ,
i.e. by means of *distributed algorithms ("protocols").*

Complexity is not in computation but in communication.

Informatics comprises formal aspects

that can't be explained as

functions $f: \Sigma^* \dashrightarrow \Sigma^*$

you can't kiss yourself

1.2 SOC comprises *non-ending behavior*

communicating not at the end, but *while* computing

SOC "always on"

cloud

elevator control

business informatics "24/7"

classical view:
terminating behavior is intended,
infinite behavior is mistaken.

new view:
infinite behavior is intended.
terminating behavior is mistaken.

How cope with this?

by means of temporal logic

"from now on, q holds" **Gq**

… invariants …

"eventually p holds" **Fp**

… a real logic

with laws such as

¬(F ¬p) = Gp

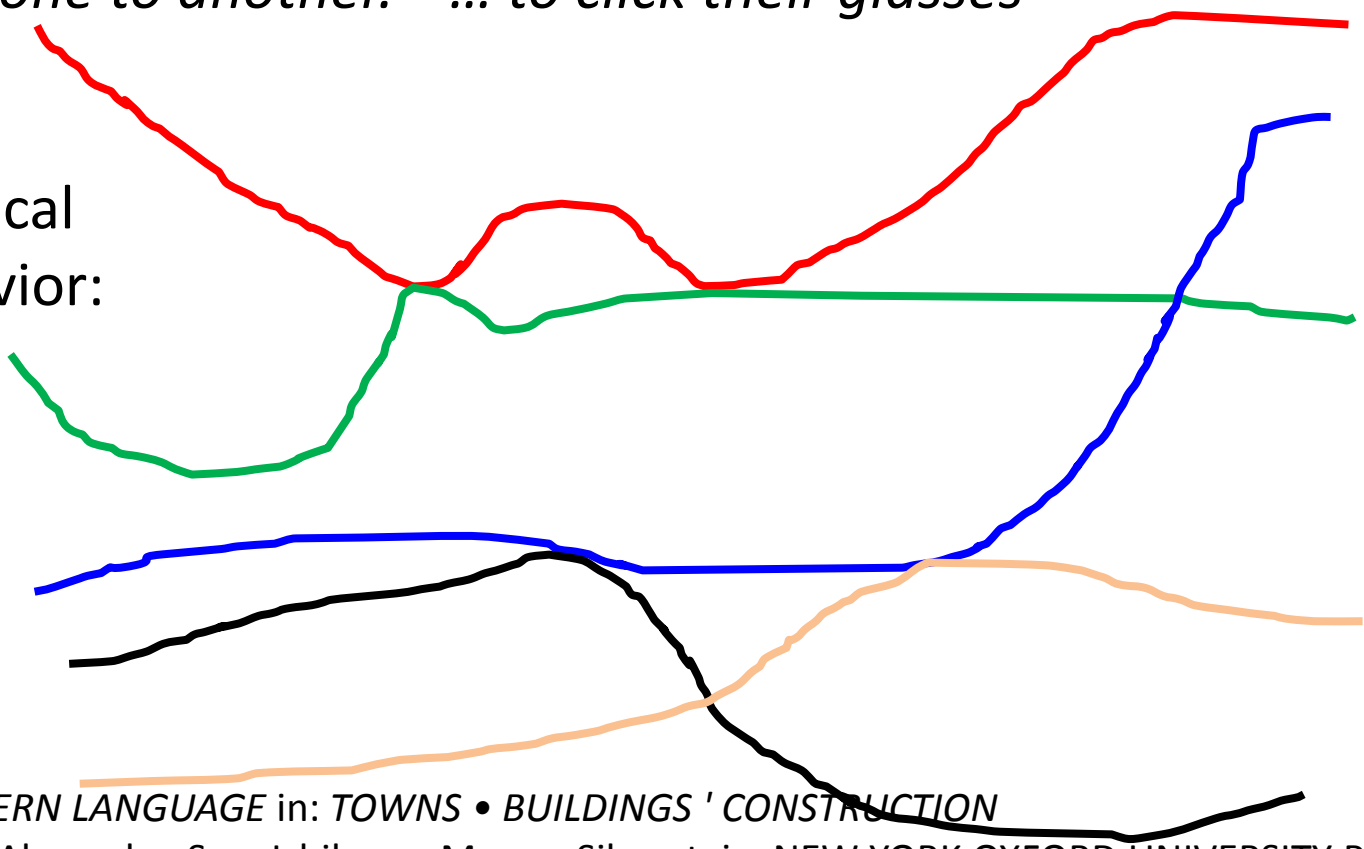
1.3 SOC comprises *causal (in)dependence*

Monday we learnded about patterns …

behavior according to the beer hall pattern (*):

" … so that people are continuously criss-crossing from one to another." … to click their glasses

a typical behavior:



(*) *A PATTERN LANGUAGE* in: *TOWNS • BUILDINGS ' CONSTRUCTION*
Christofer Alexander, Sara Ishikawa, Murray Silverstein NEW YORK OXFORD UNIVERSITY PRESS 1977

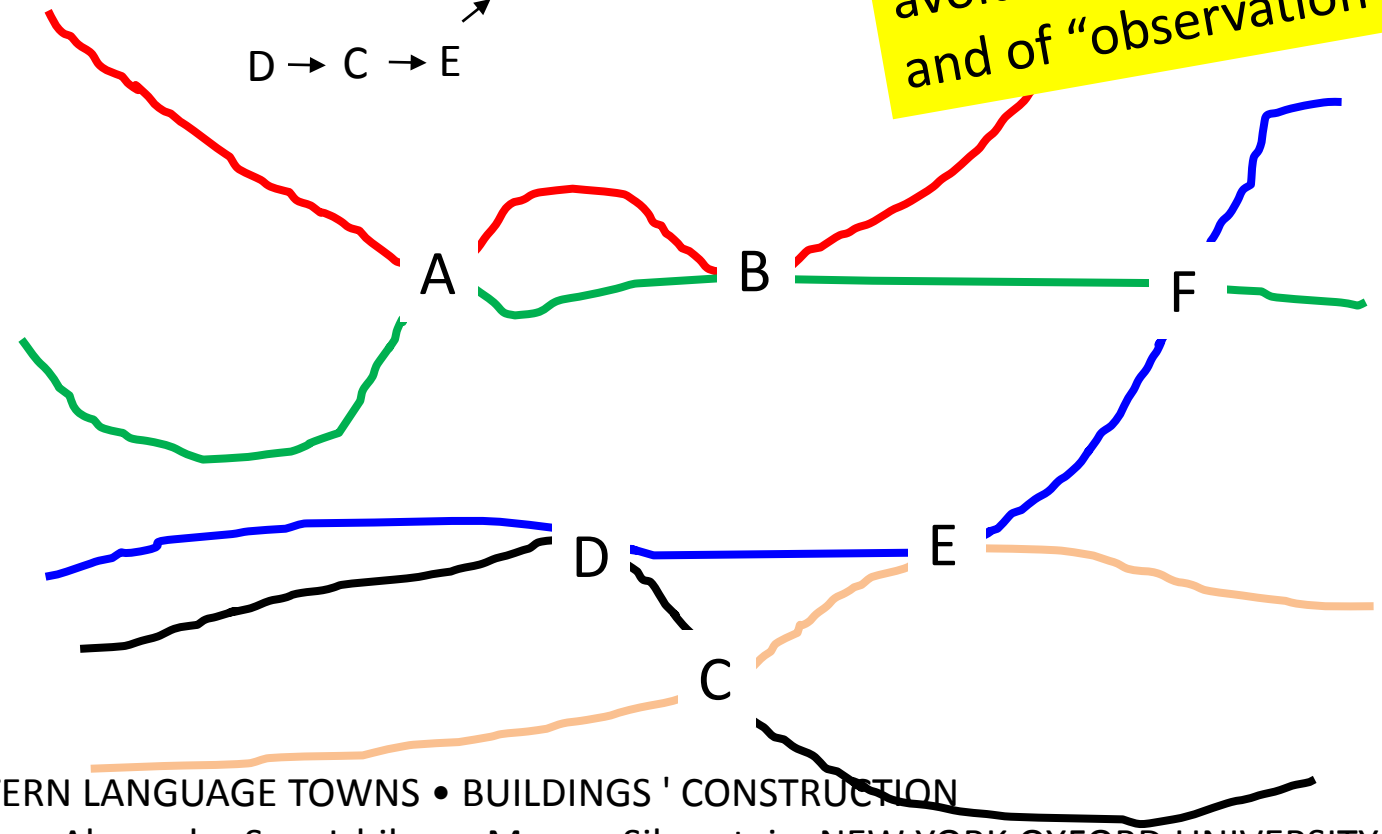
What is a behavior formally?

Causality structures the world

To understand SOC,
avoid a naïve notions of "time"
and of "observation".

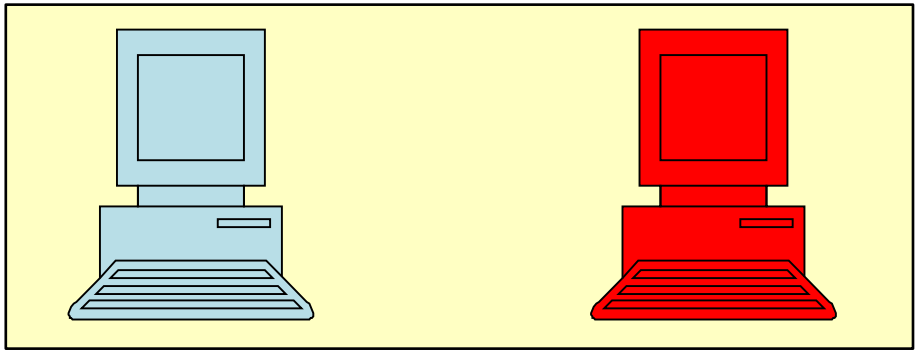
… a *partially* ordered set of events

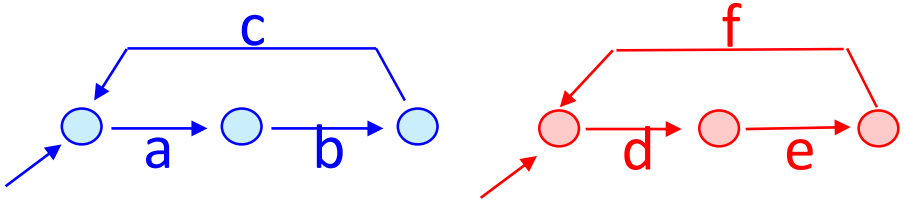
A → B → F
↗
D → C → E

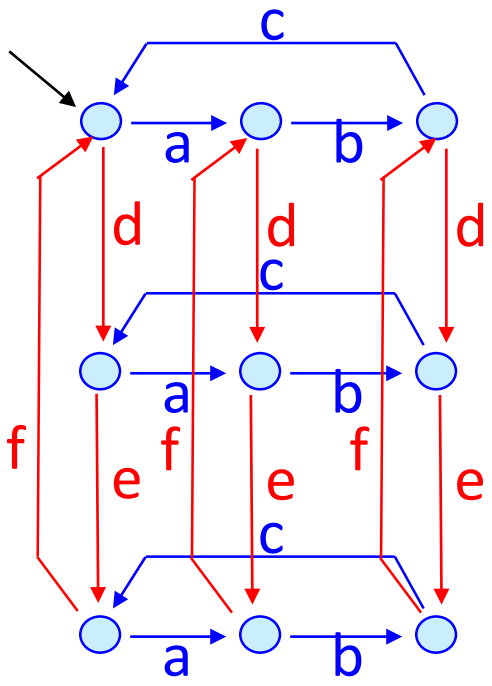


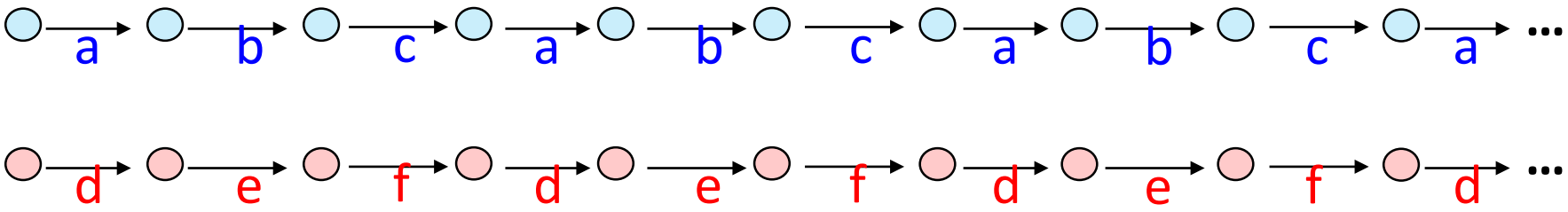
(*) A PATTERN LANGUAGE TOWNS • BUILDINGS ' CONSTRUCTION
Chris to f her Alexander Sara Ishikawa Murray Silverstein NEW YORK OXFORD UNIVERSITY PRESS 1977

This view has deep consequences!
The classical view:

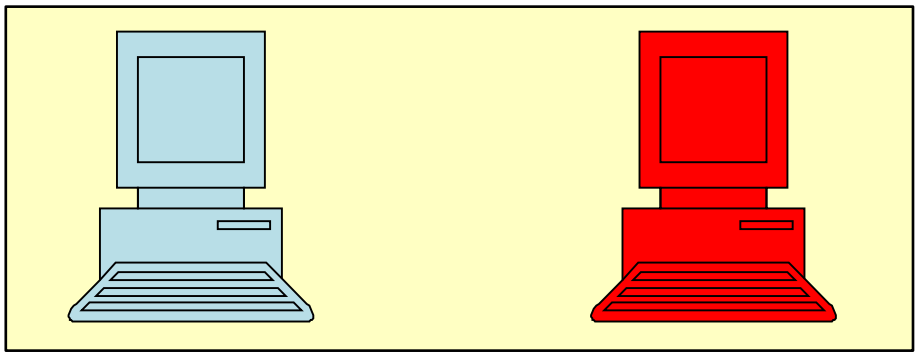


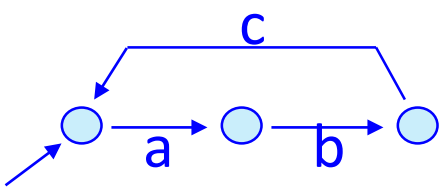


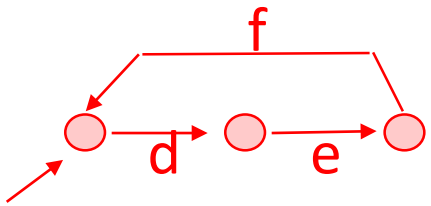


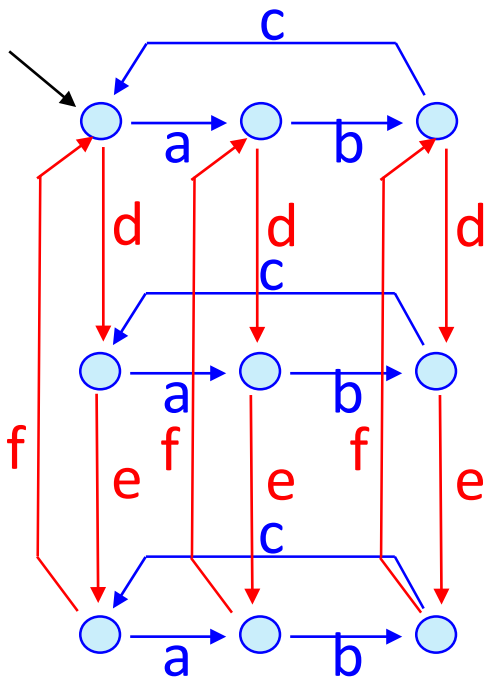


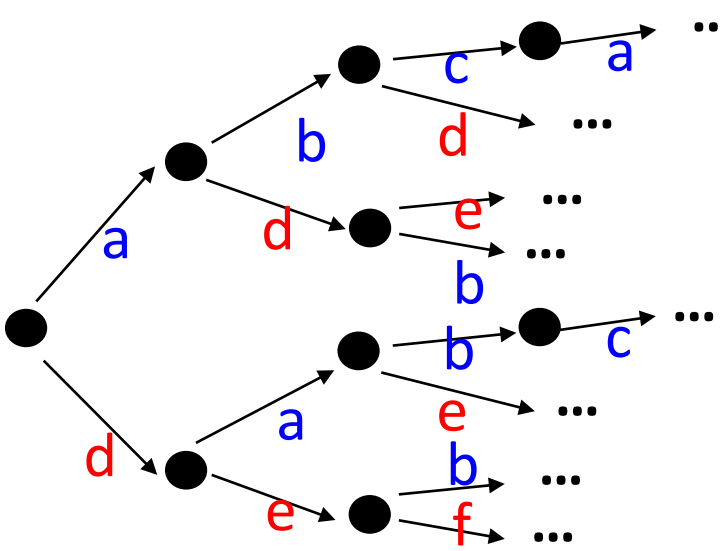
This view has deep consequences!
The classical view:







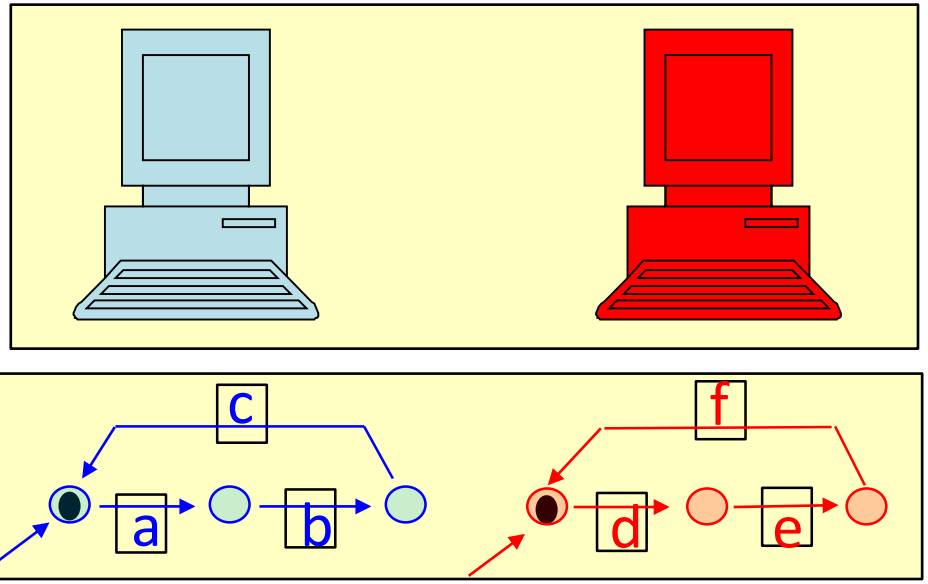


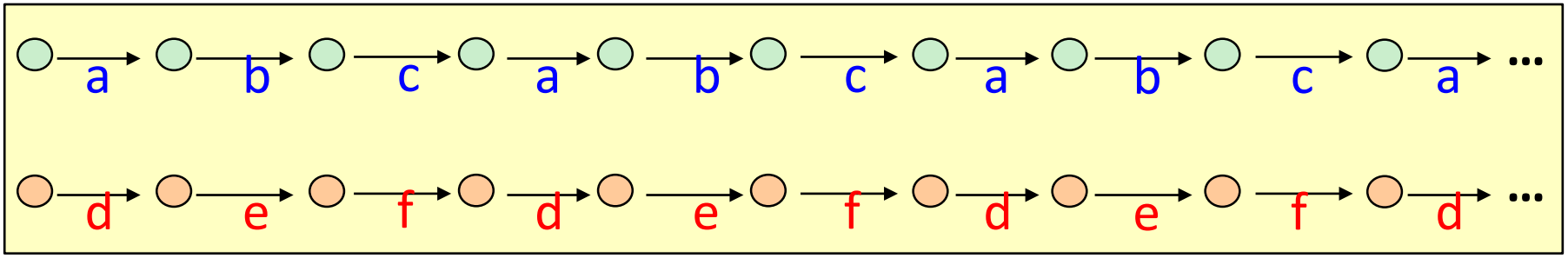


+ fairness assumption

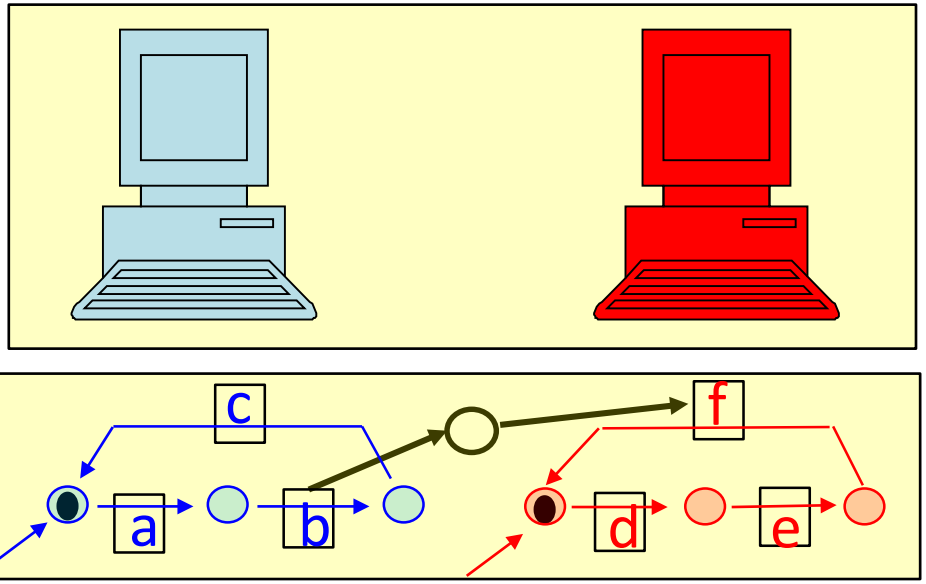
motivated by
“observation”

This view has deep consequences!
The causality based view:

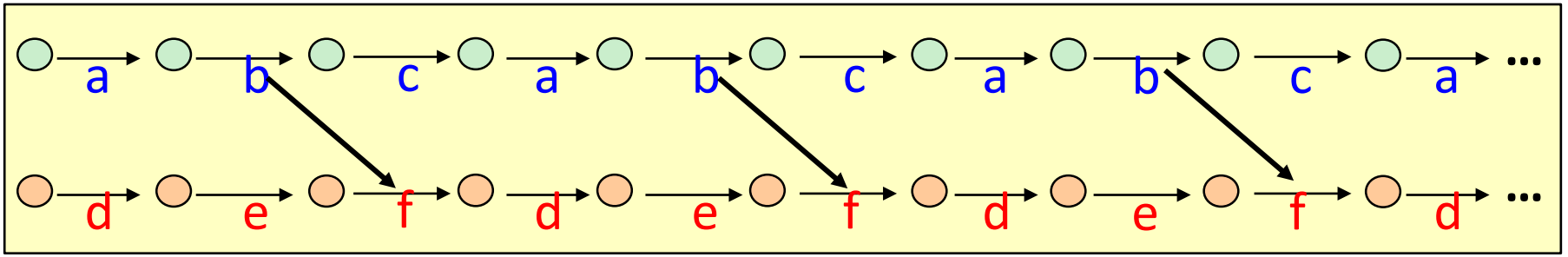




a variant: i-th b before i-th f



a *deterministic* system;
no alternatives;
one behavior (run, execution)



… summing up

Semantics of SOC
should be mathematics!

True, this is presently not the case.

BUT WE SHOULD spend effort into this!

Foundations of SOC

1. SOC exceeds classical Theoretical Informatics
2. Services are made to be composed!
3. Required: mechanisms to compose *many* services.
4. Composition must retain or guarantee *properties.*
5. Composition is surprisingly expressive!

Interaction is represented as *composition*

Requirements:

- The – elementary – notion of composition of services
  is a (simple!) mathematical (or logical!) operation.

- For services S and T,
  the composition $S \oplus T$
  is a service again.

"One cannot not communicate. (*)

(Frequently, $S \oplus T$ does not interact any more.)

ticketing $=_{def}$
sell_ticket $\oplus$ buy_ticket

(*) Paul Watzlawick, 1967

a general goal

Description of

semantics and (in particular) composition of services:

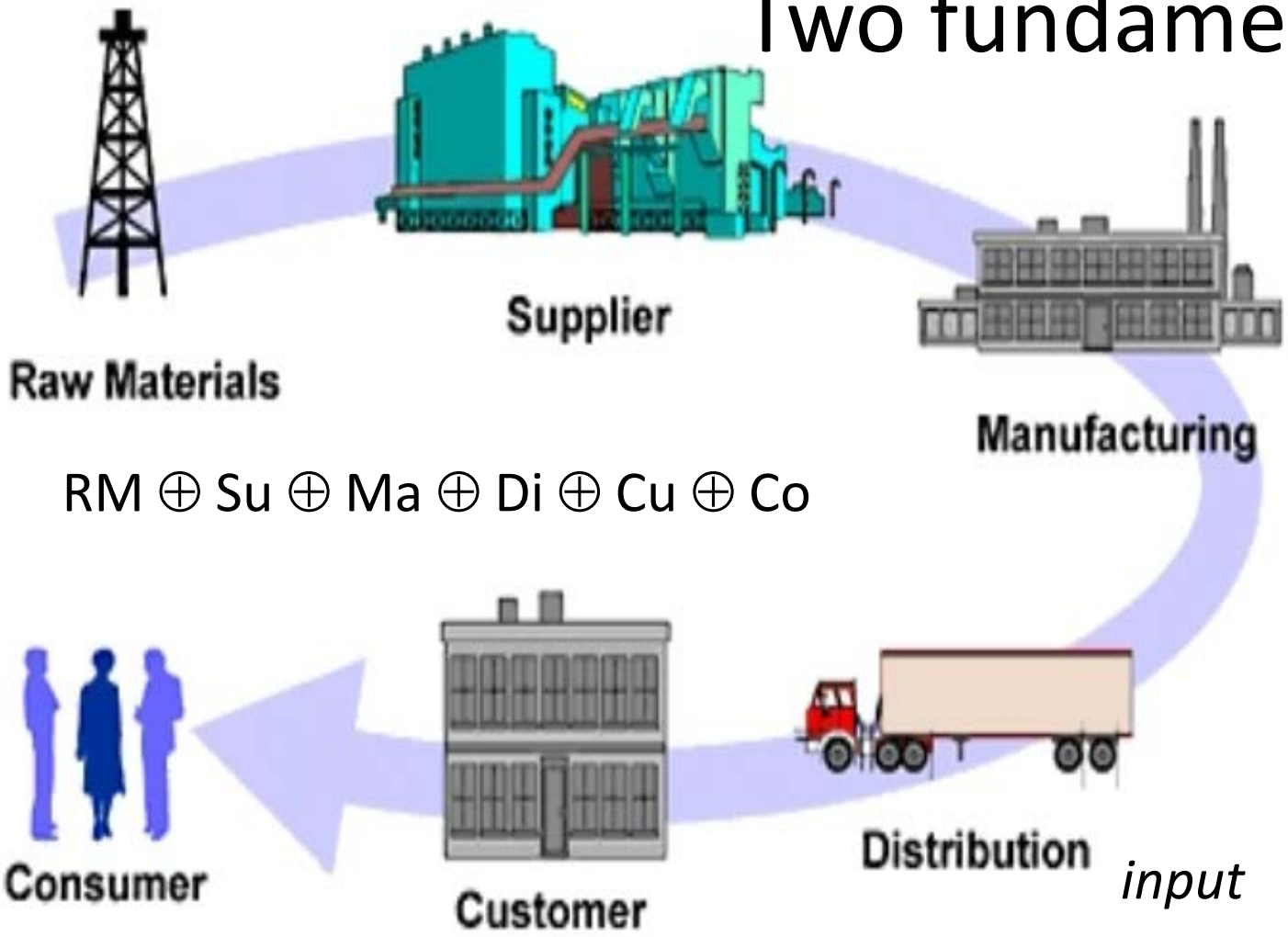
- on a high level of business logic.
- not on a low level of implementation details.

Describe system *properties* !

Foundations of SOC

1. SOC exceeds classical Theoretical Informatics
2. Services are made to be composed!
3. Required: mechanisms to compose *many* services.
4. Composition must retain or guarantee *properties.*
5. Composition is surprisingly expressive!

Two fundamental ideas:



RM ⊕ Su ⊕ Ma ⊕ Di ⊕ Cu ⊕ Co

2. Composition must be associative!

1. A service *S* has an interface.
The interface of S is partitioned into a *left* and a *right* port S_l and S_r !

input	and	*output*
customer	and	*supplier*
provider	and	*requester*
producer	and	*consumer*
buy side	and	*sell side*

Two fundamental ideas:

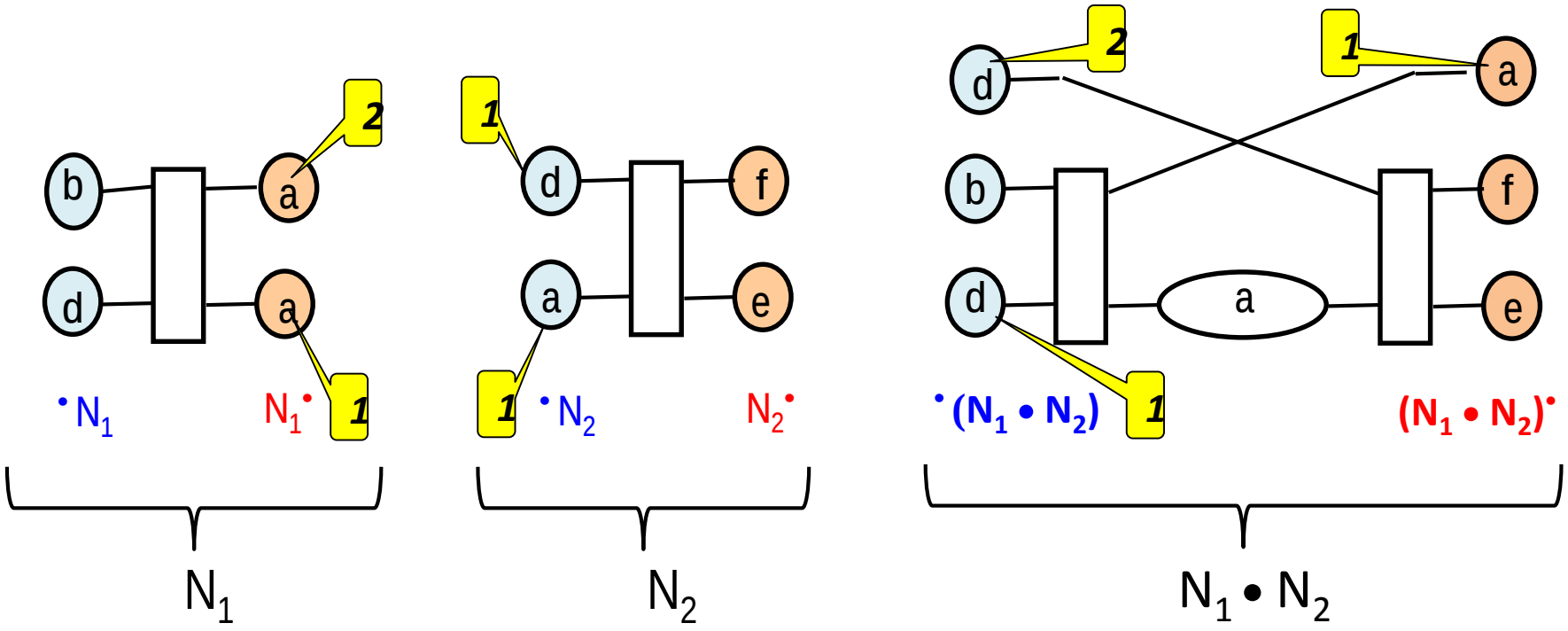
2. Composition must be associative!

socket ⊕ adapter ⊕ plug *no brackets!*

1. A service *S* has an interface.
The interface of S is partitioned into a *left* and a *right* port S_l and S_r !

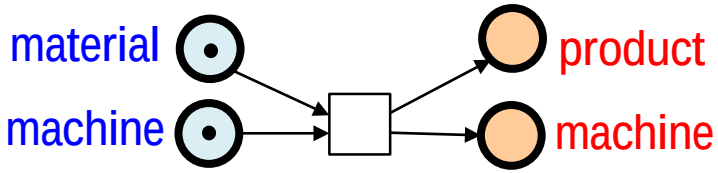
input	and	*output*
customer	and	*supplier*
provider	and	*requester*
producer	and	*consumer*
buy side	and	*sell side*

left-right interface

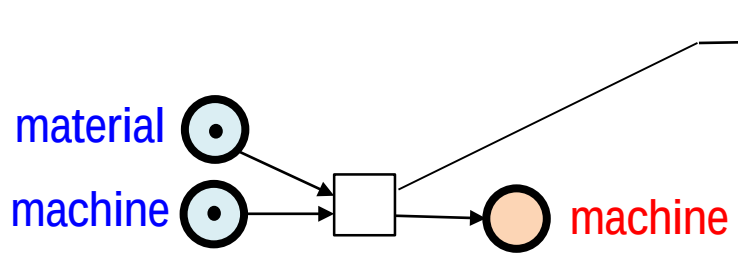


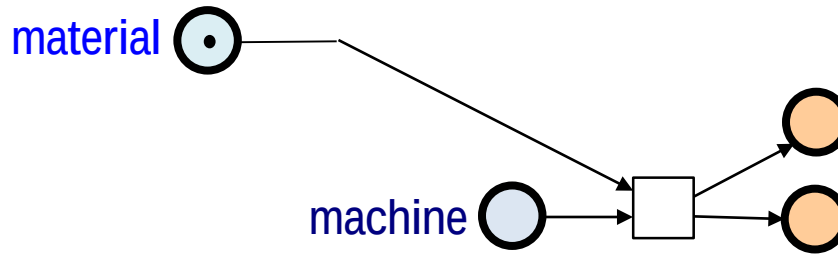
Left ports: ***blue***;
right ports: ***red*** .
Indices of equally labelled elements: **yellow**
(index “1” is mostly skipped).
The inner nodes of N_1 and N_2 are sketched as boxes.

Associative composition



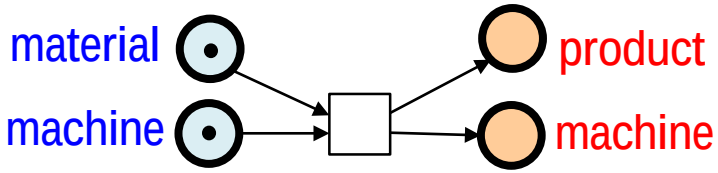
a. workflow N, transforming material into products by help of a machine



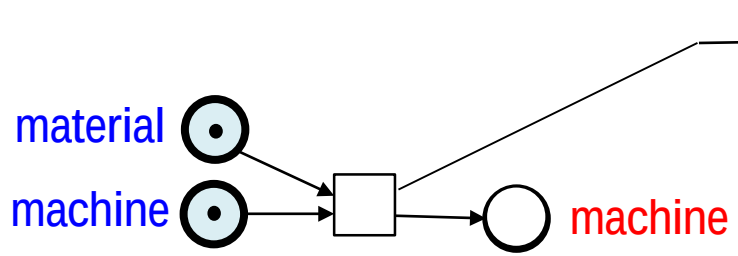


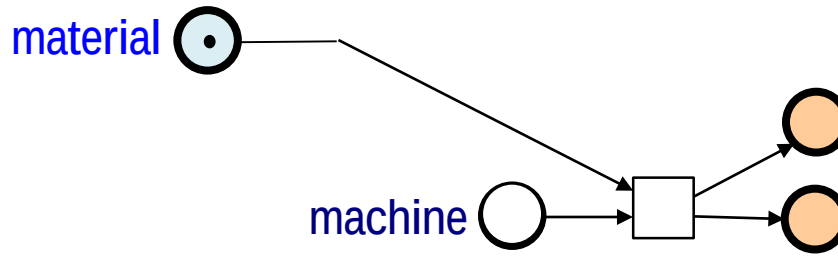
b. composed workflow, N•N

Associative composition



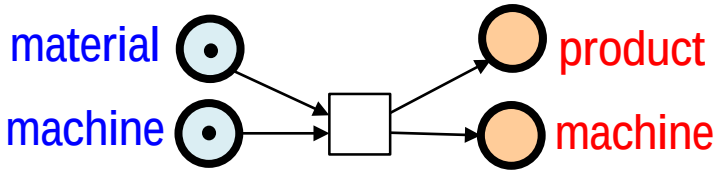
a. workflow N, transforming material into products by help of a machine



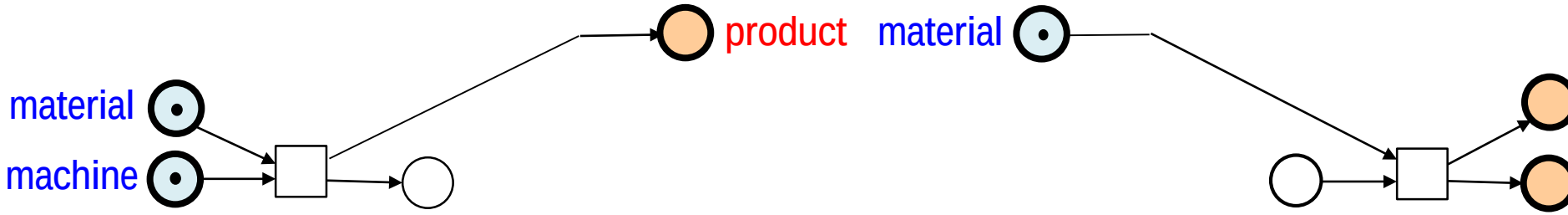


b. composed workflow, N•N

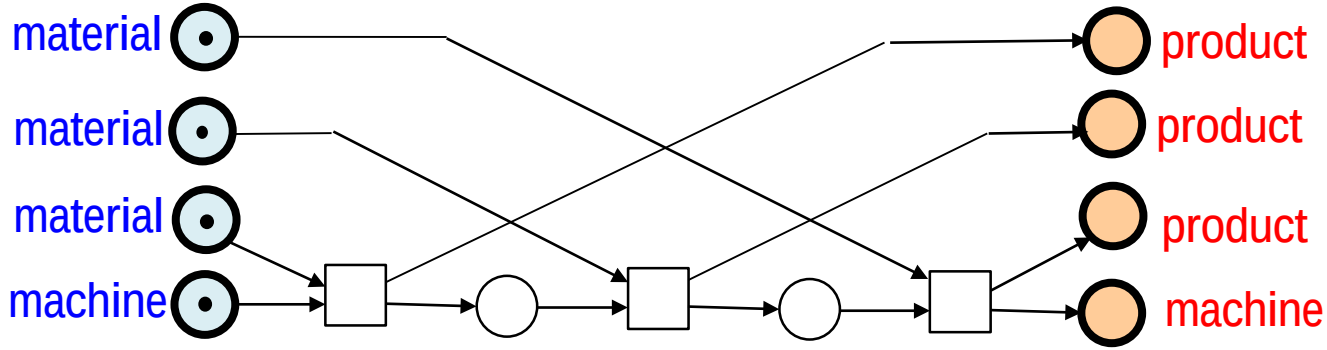
Associative composition



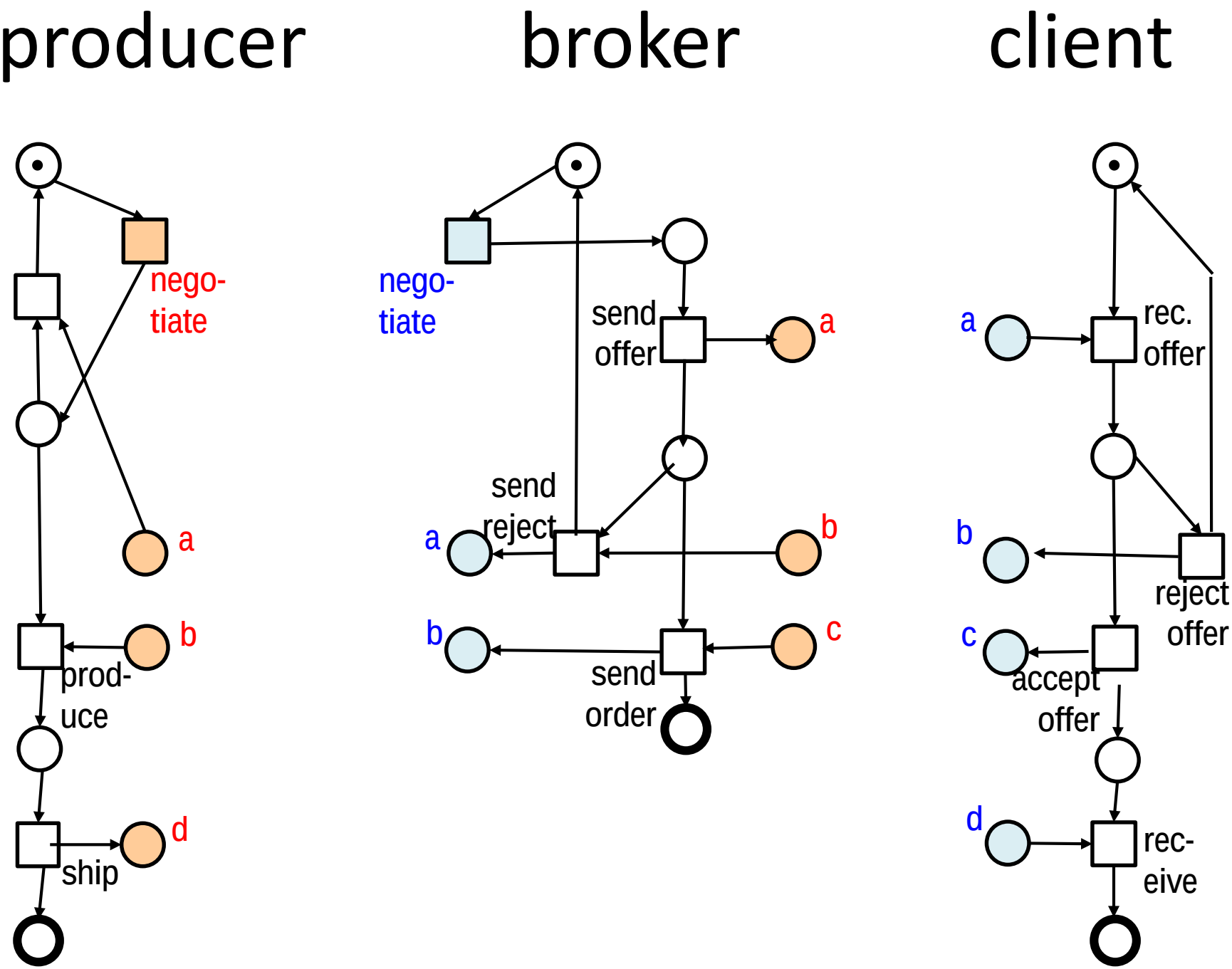
a. workflow N, transforming material into products by help of a machine



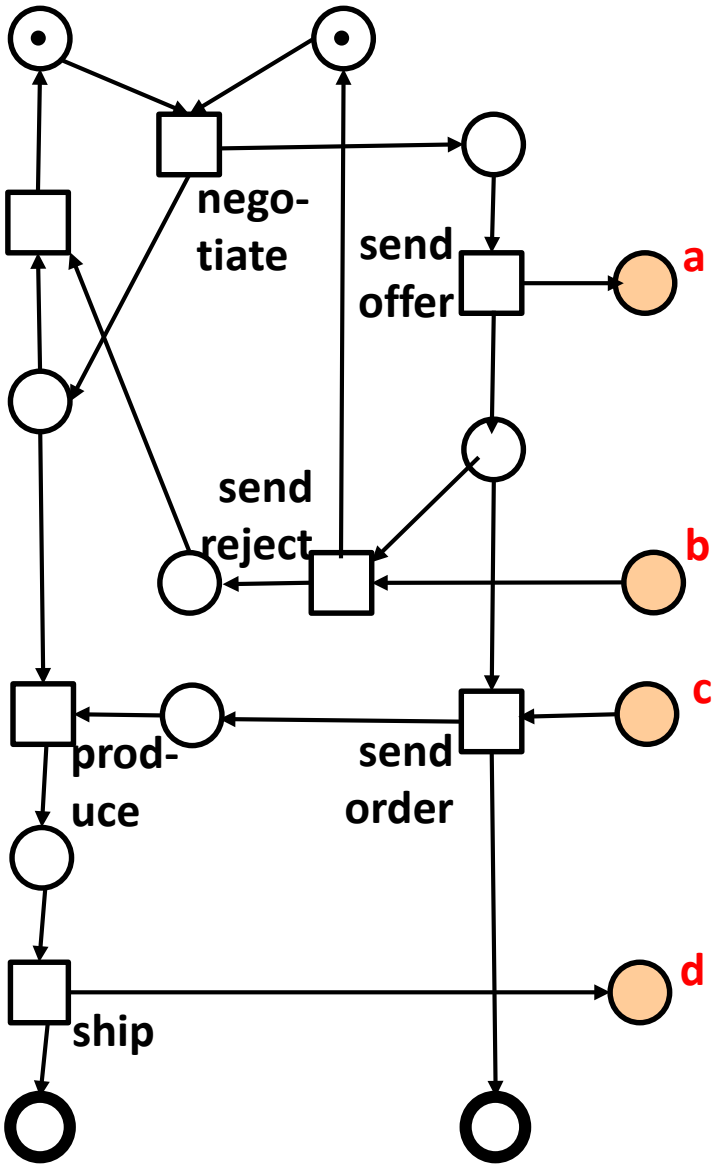
b. composed workflow, N·N



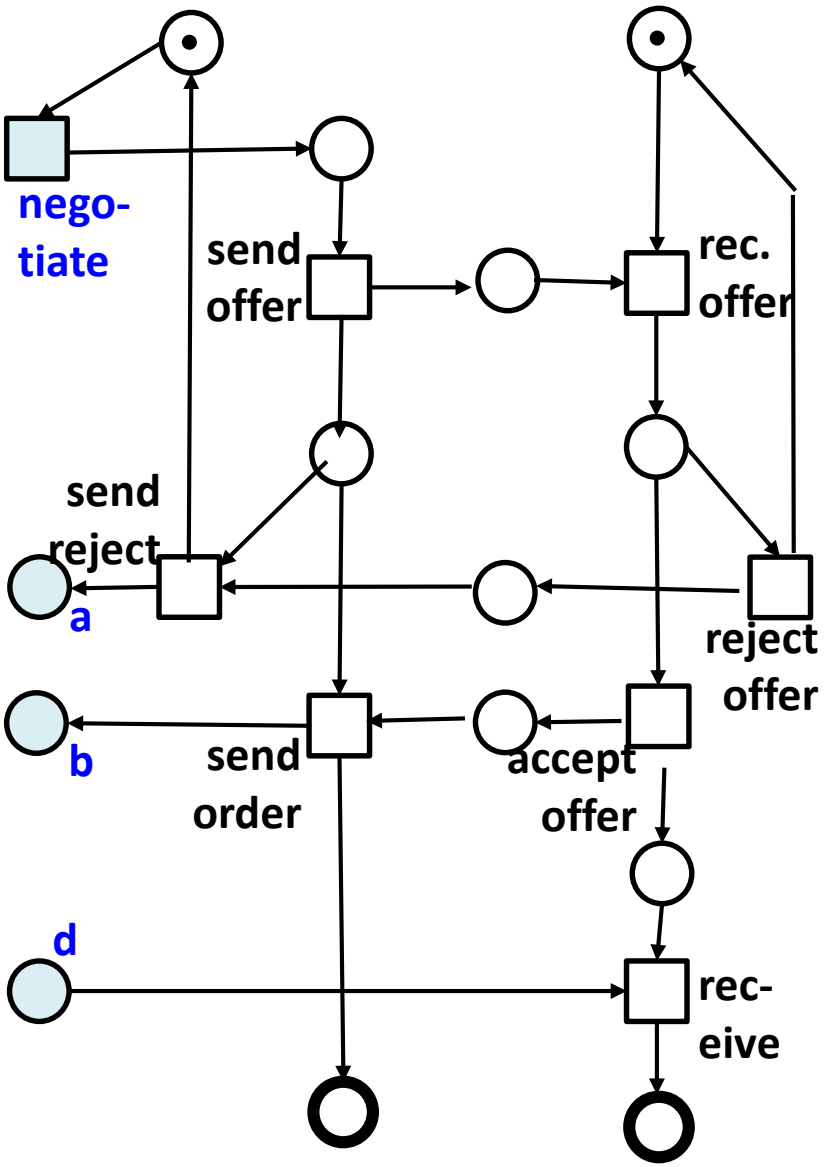
c. composed workflow, N·N·N


producer
broker
client
nego-
tiate
a
b
prod-
uce
d
ship
nego-
tiate
send
offer
a
send
reject
a
b
b
c
send
order
a
rec.
offer
b
reject
offer
c
accept
offer
d
rec-
eive

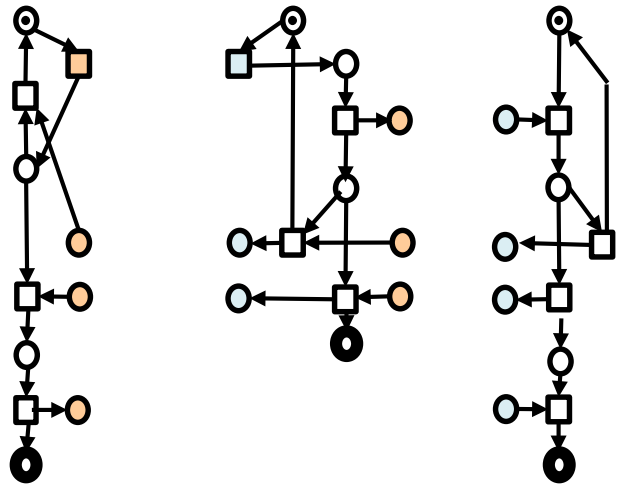
producer • broker

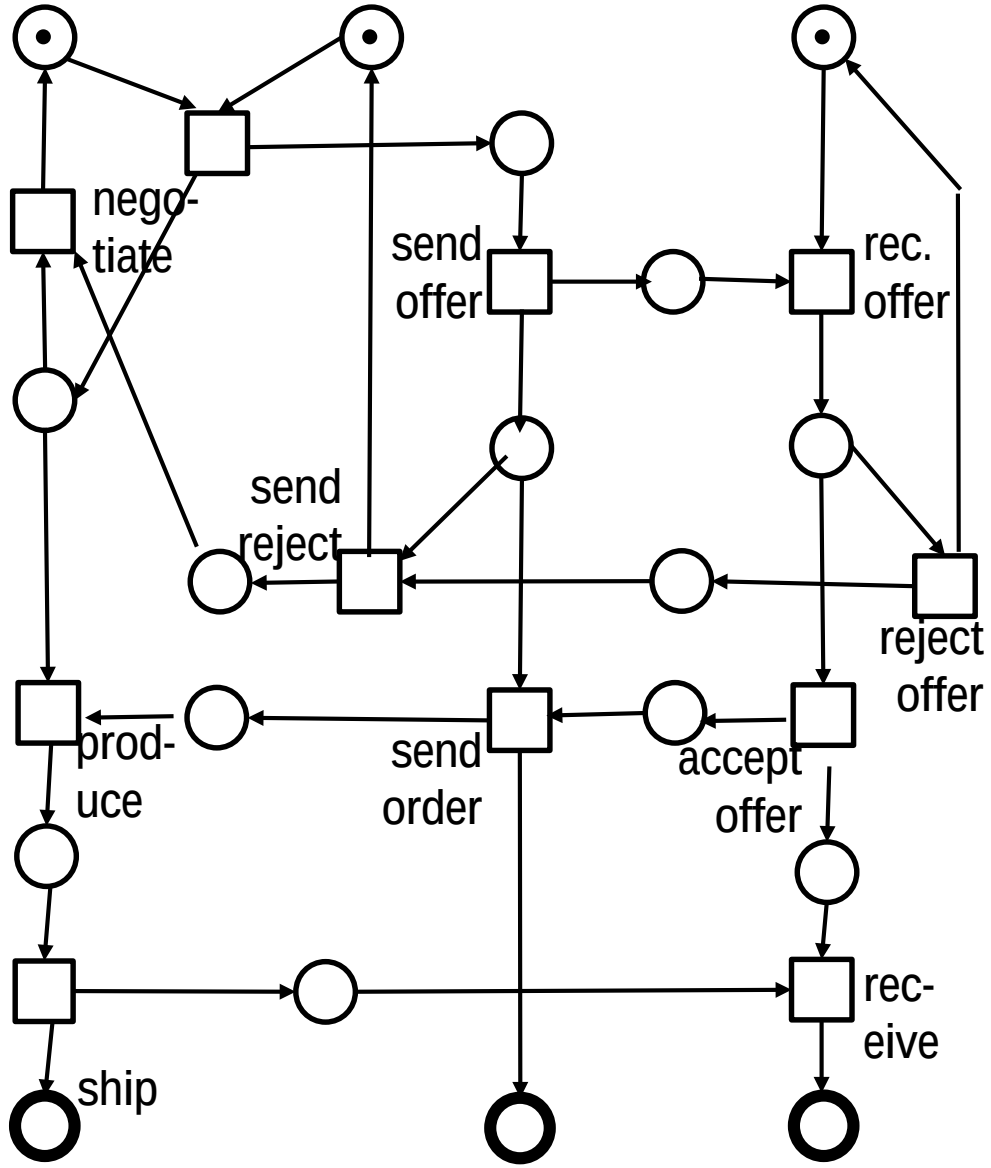


broker • client



producer • broker • client

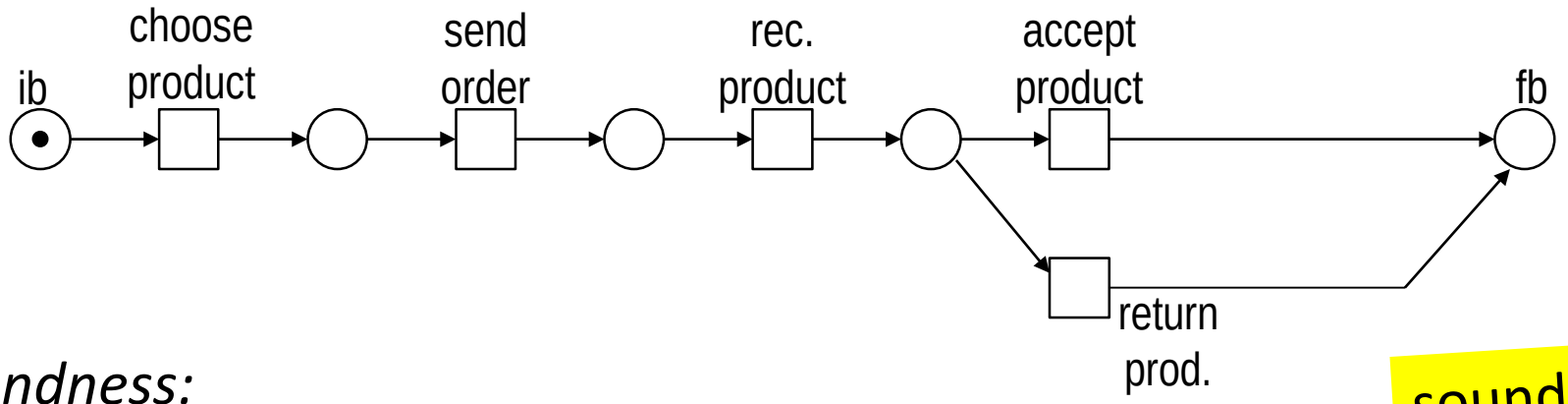




Foundations of SOC

1. SOC exceeds classical Theoretical Informatics
2. Services are made to be composed!
3. Required: mechanisms to compose *many* services.
4. Composition must retain or guarantee *properties.*
5. Composition is surprisingly expressive!

Example: Internet Shopping



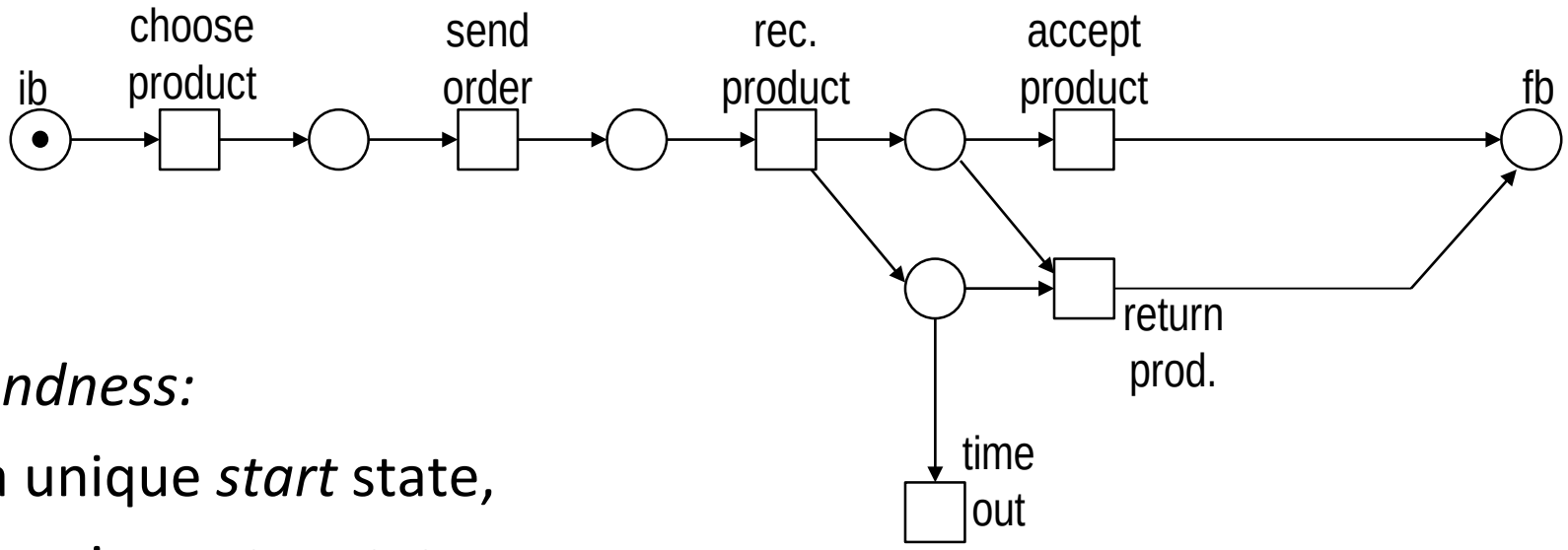
sound!

Soundness:

- a unique *start* state,
- a unique *stop* state,
- each transition is reachable,
- you always can reach *stop*
- no litter remains

Soundness is efficiently decidable. (v.d. Aalst)

… not sound

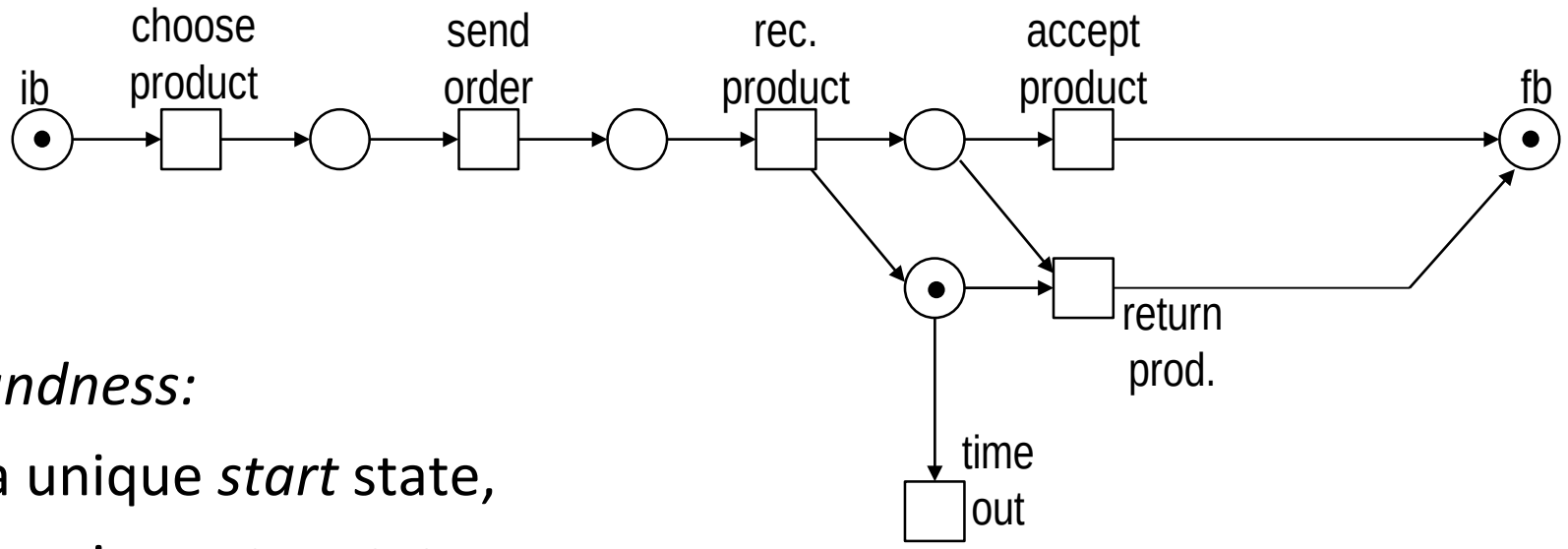


Soundness:

- a unique *start* state,
- a unique *stop* state,
- each transition is reachable,
- you always can reach *stop*
- no litter remains

Soundness is efficiently decidable. (v.d. Aalst)

…because

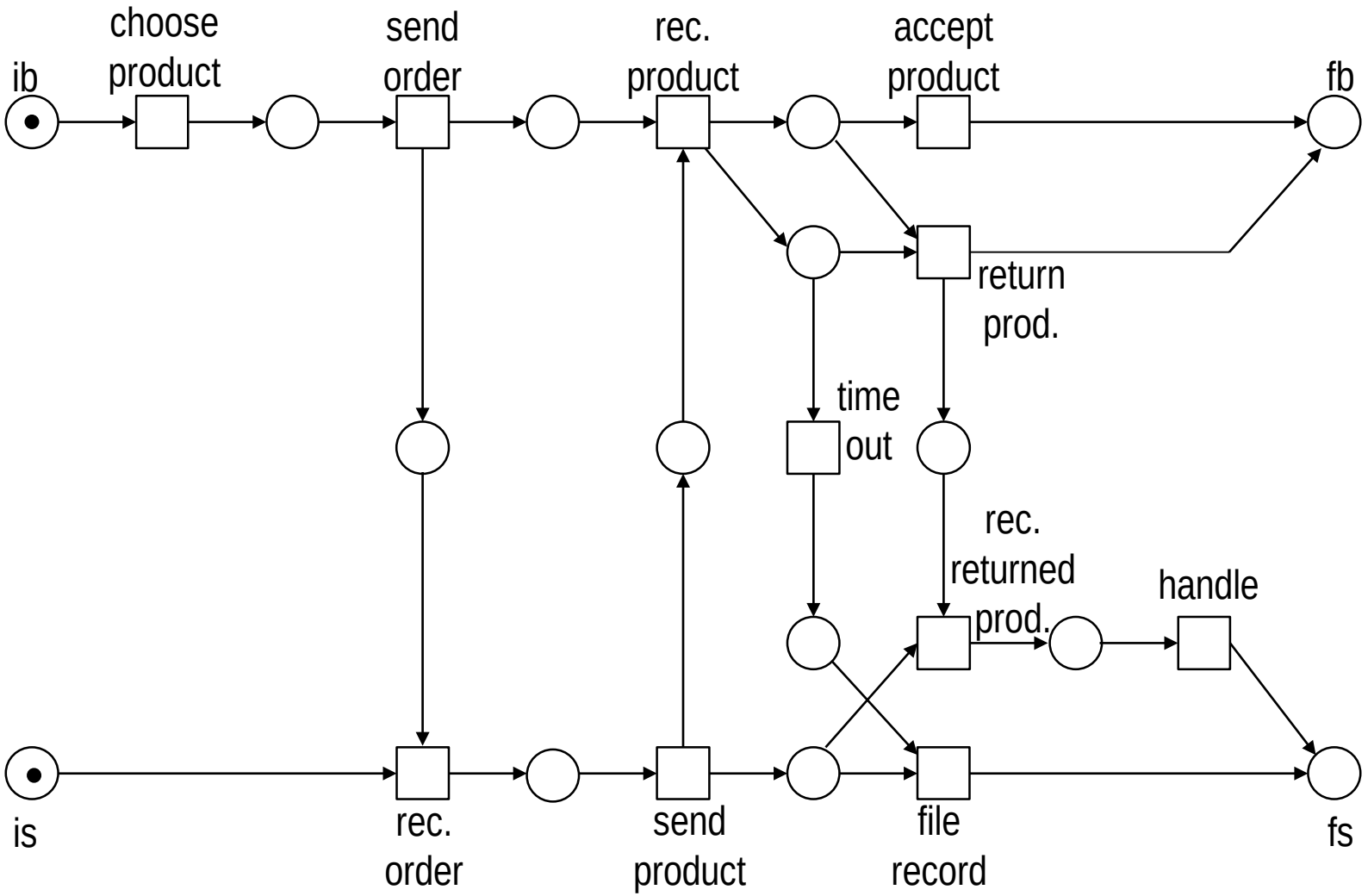


Soundness:

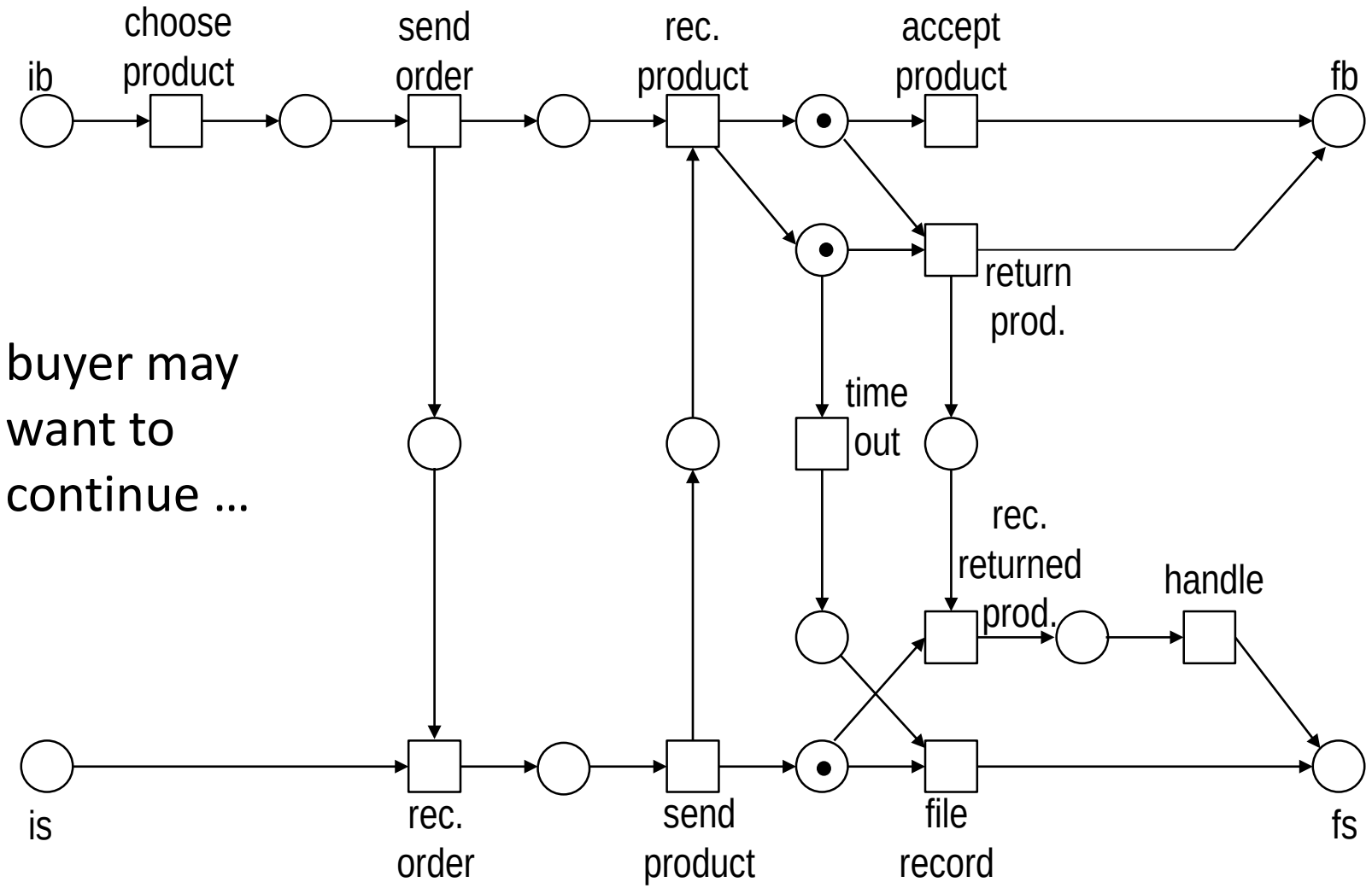
- a unique *start* state,
- a unique *stop* state,
- each transition is reachable,
- you always can reach *stop*
- no litter remains

Soundness is efficiently decidable. (v.d. Aalst)

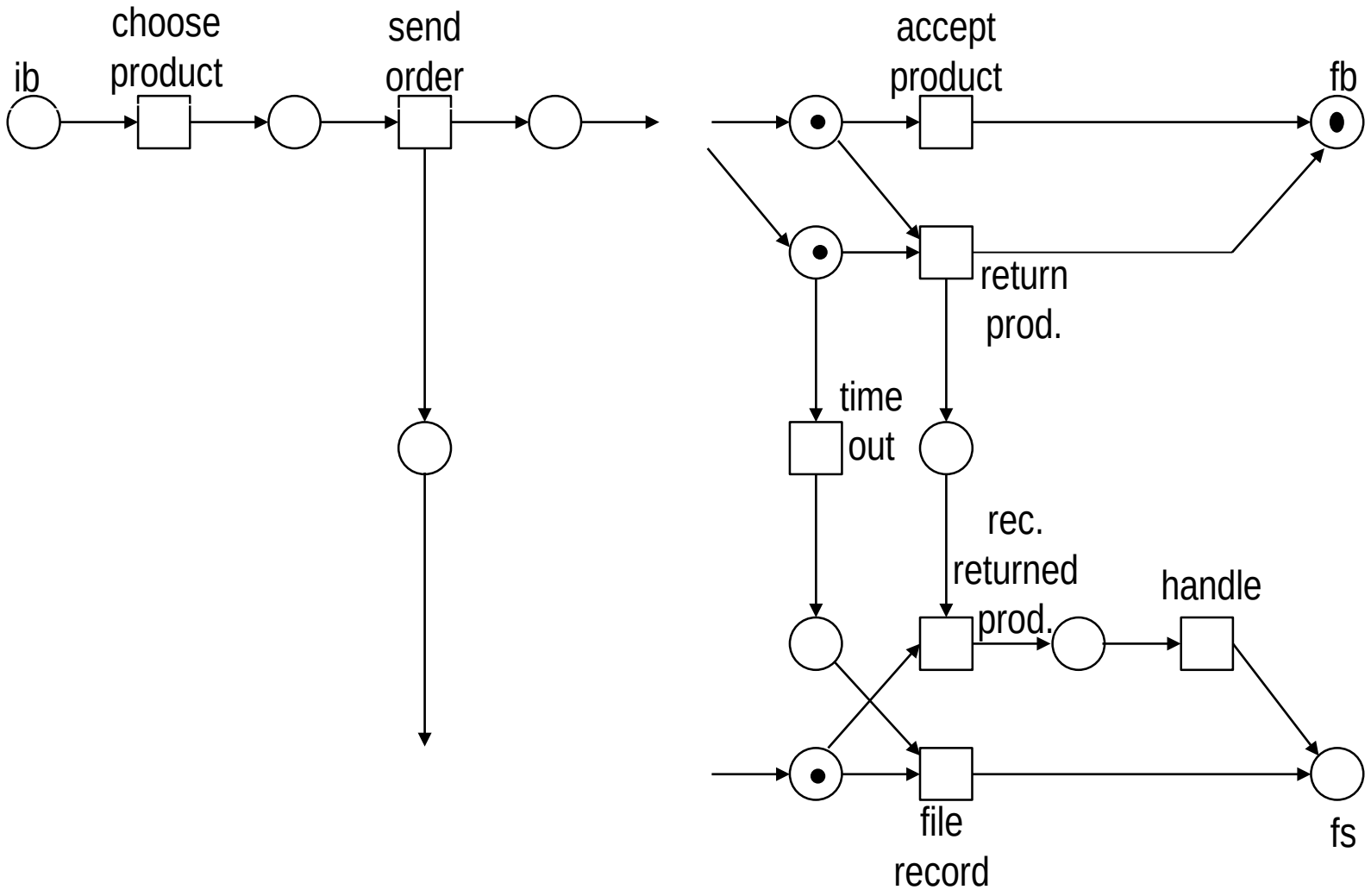
… is sound


ib
choose product
send order
rec. product
accept product
fb
return prod.
time out
rec. returned prod.
handle
is
rec. order
send product
file record
fs

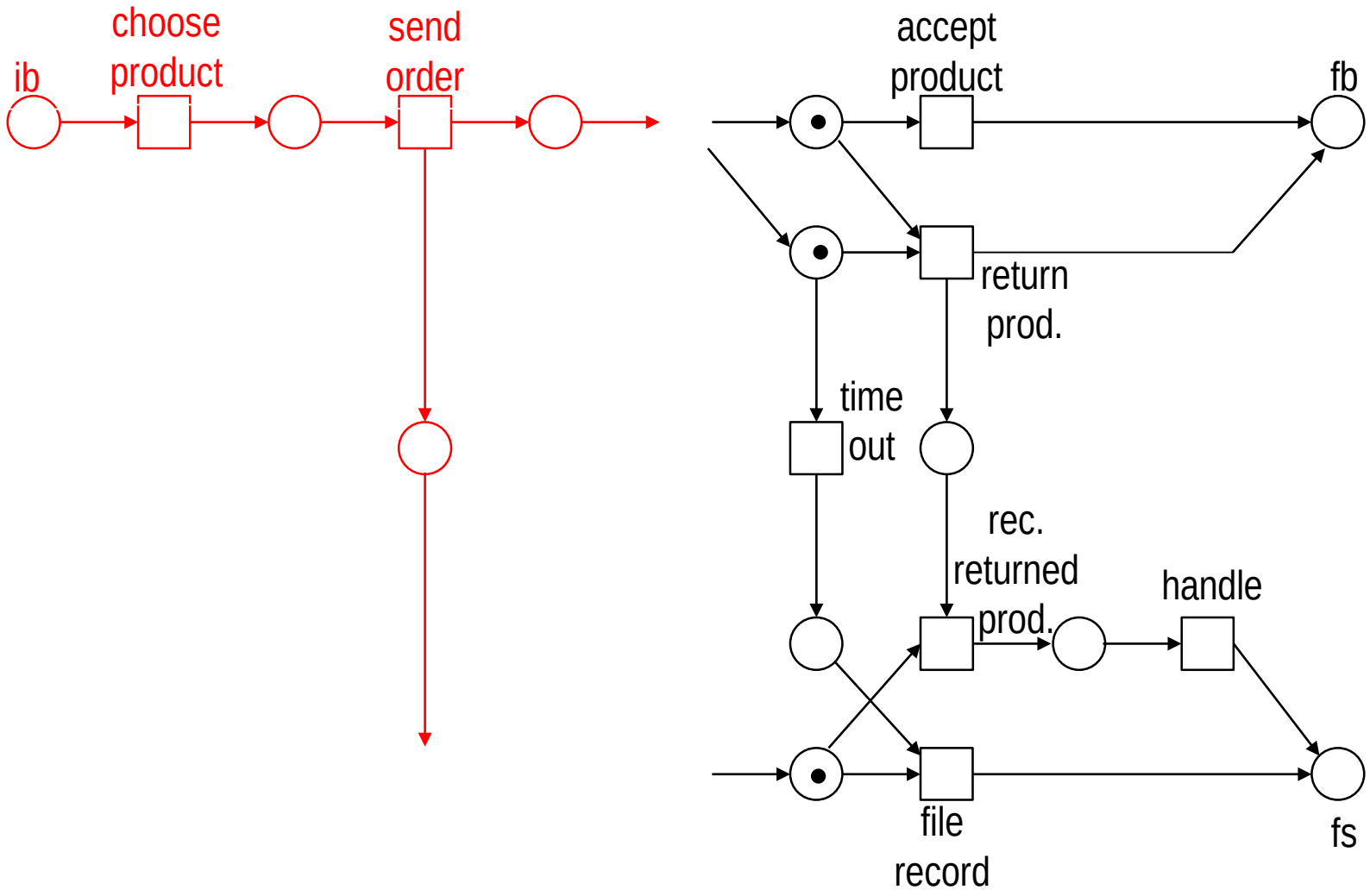
… a reachable state …



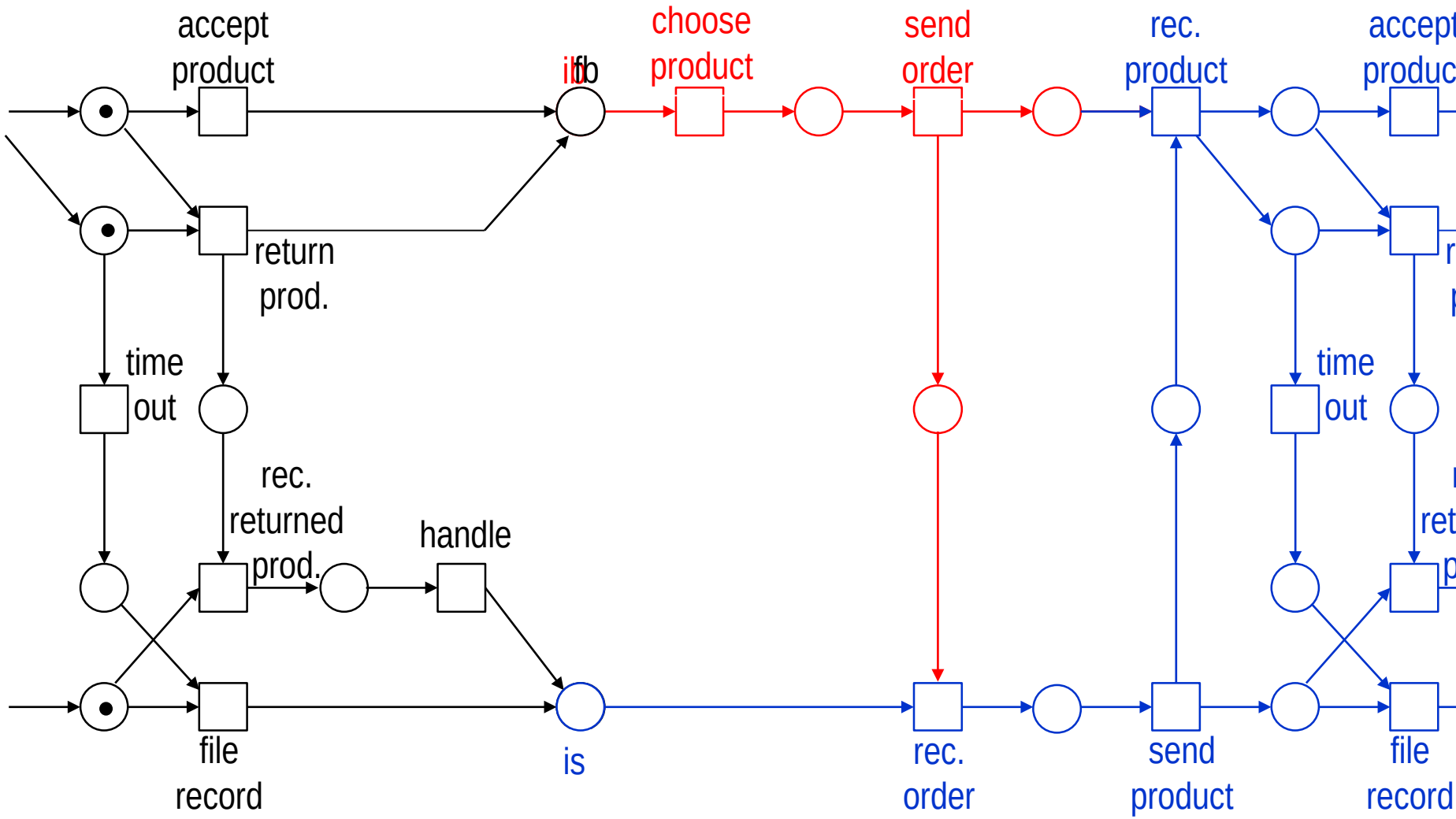
second purchase, N•N



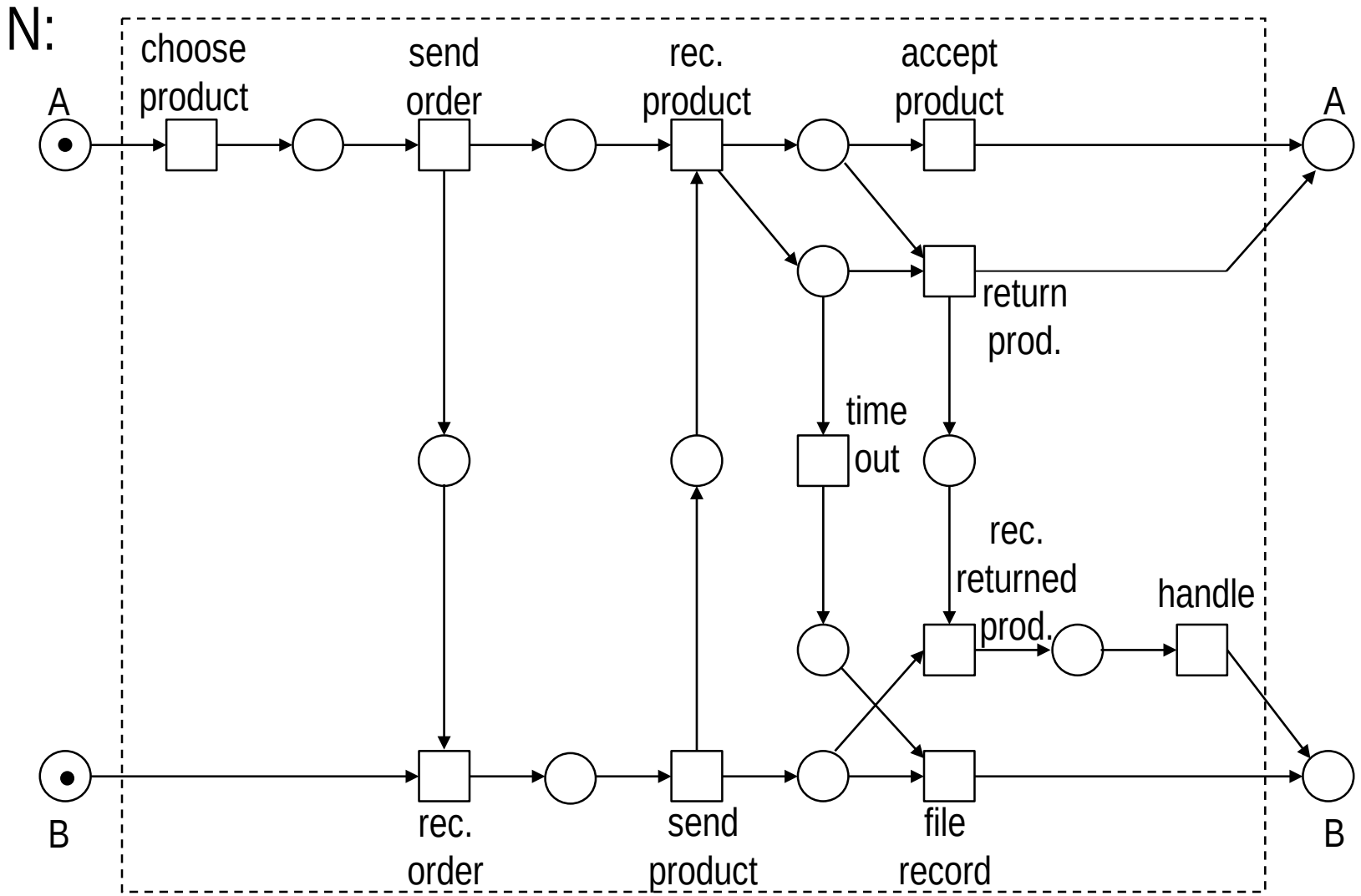
second purchase



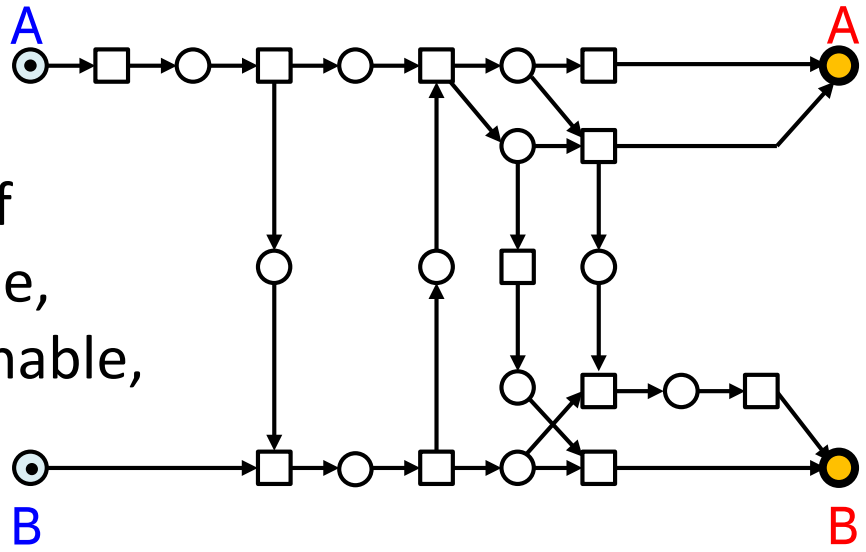
second purchase



N•N is sound, too!



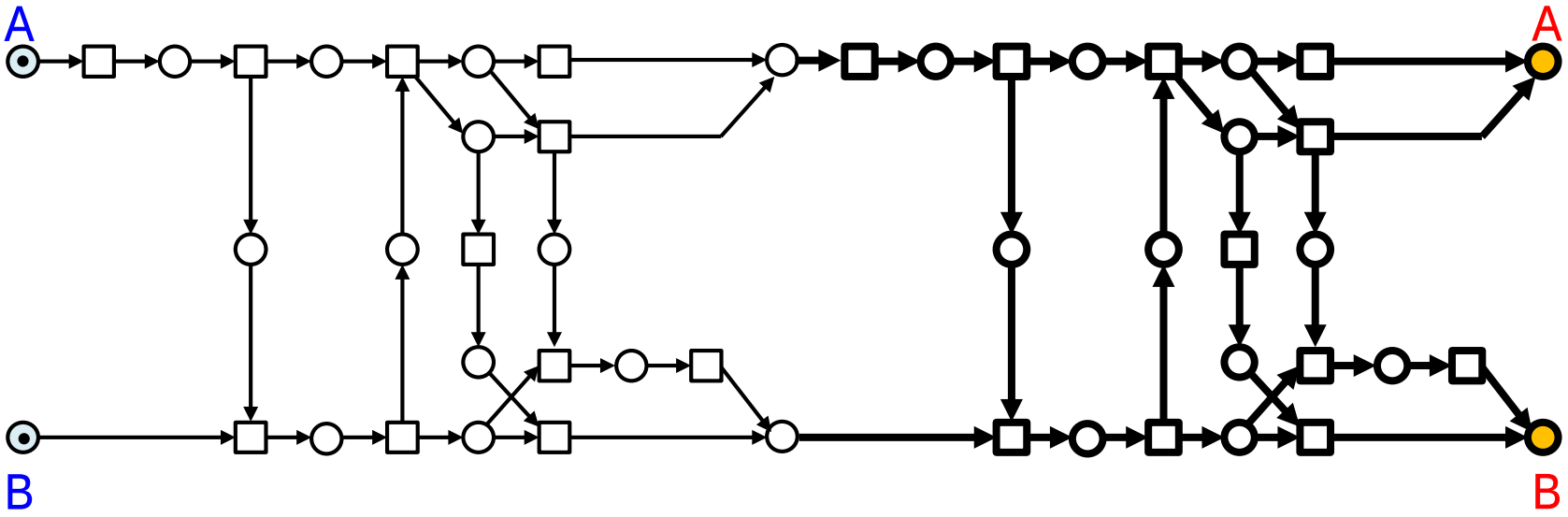
Soundness



remember: A service is *sound* iff

- all its activities are executable,
- the final state is always reachable,
- upon termination,
  no token remains.

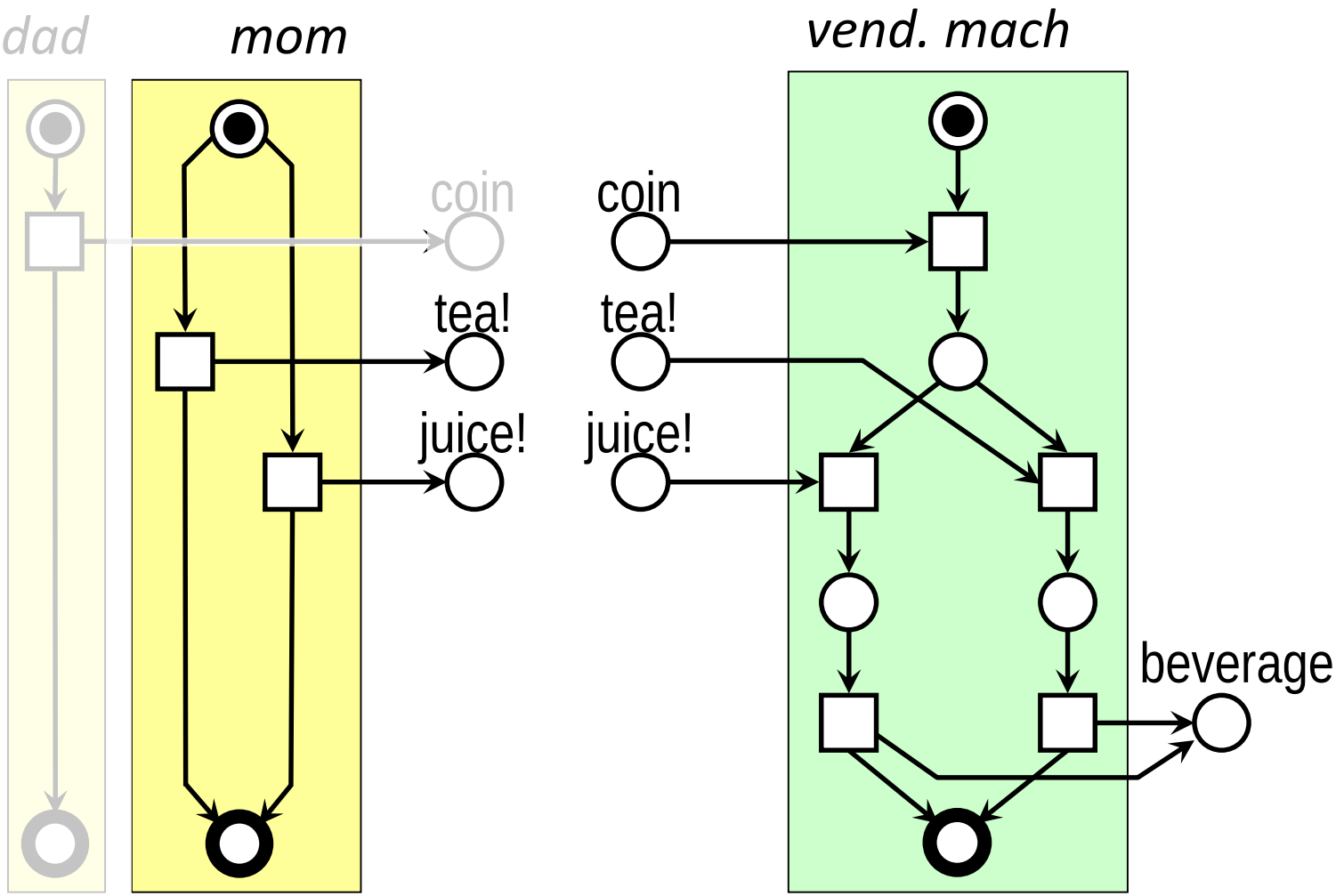
composition N • N (second instance is bold faced)



Theorem: Composition of sound services is sound.

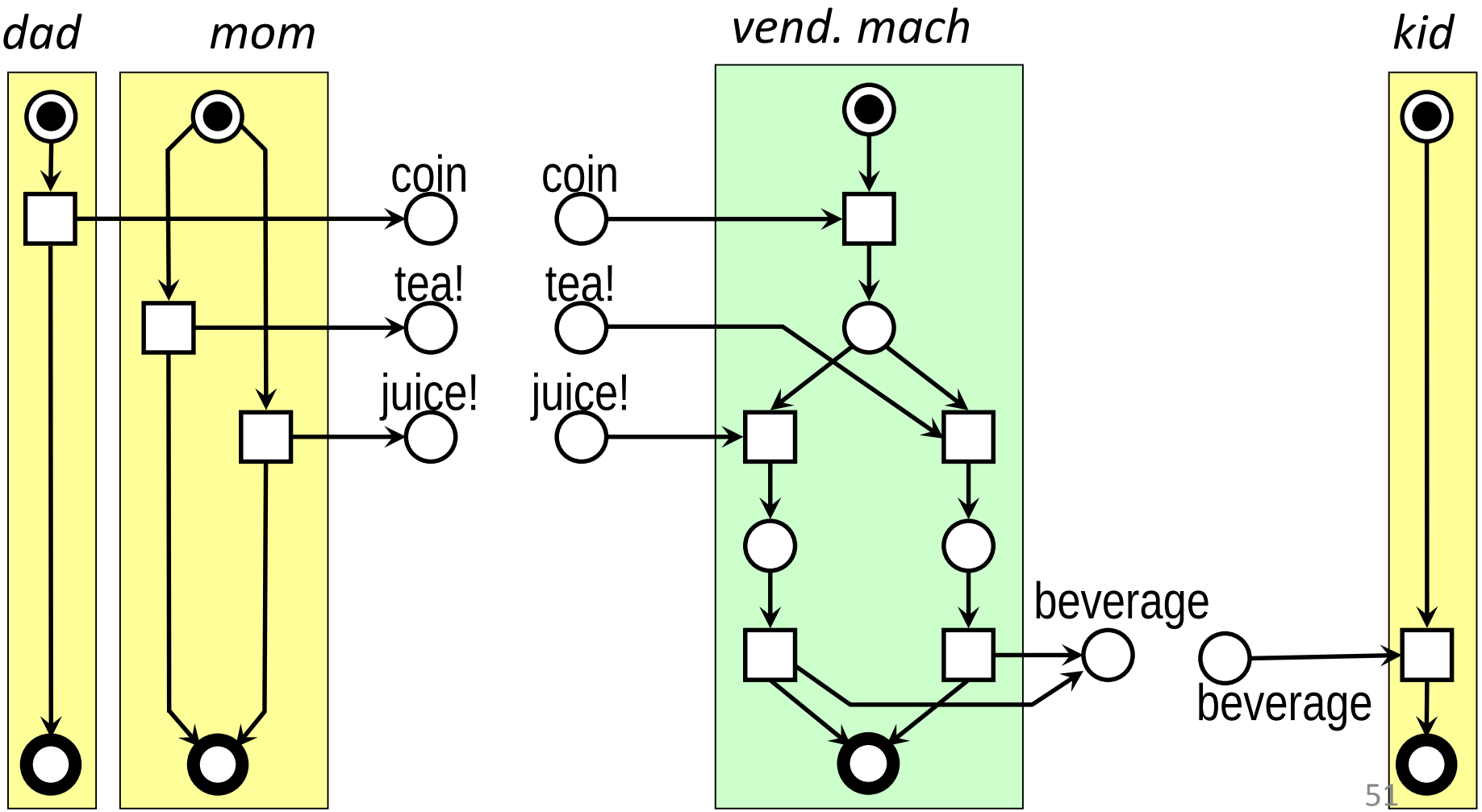
… a tricky property

dad pays, *mom* selects,



… a tricky property

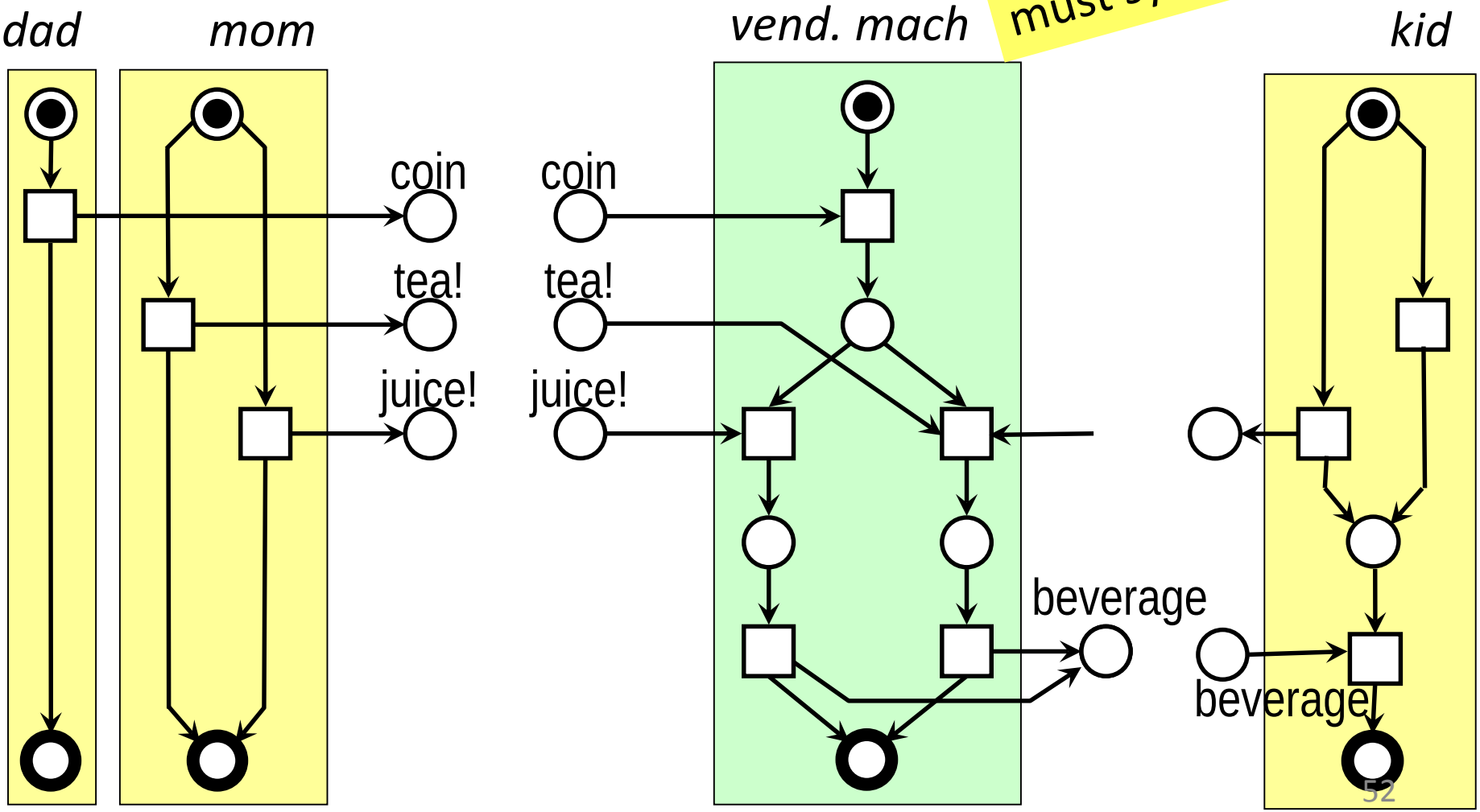
dad pays, *mom* selects, *kid* drinks.



A variant of the vending machine

dad pays, *mom* selects, *kid* drinks.

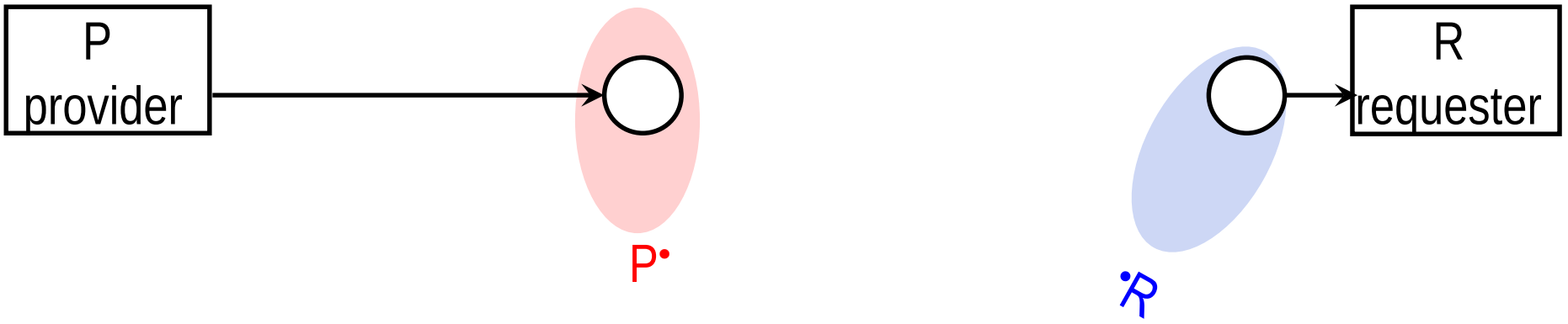
mom and kid must synchronize!



Foundations of SOC

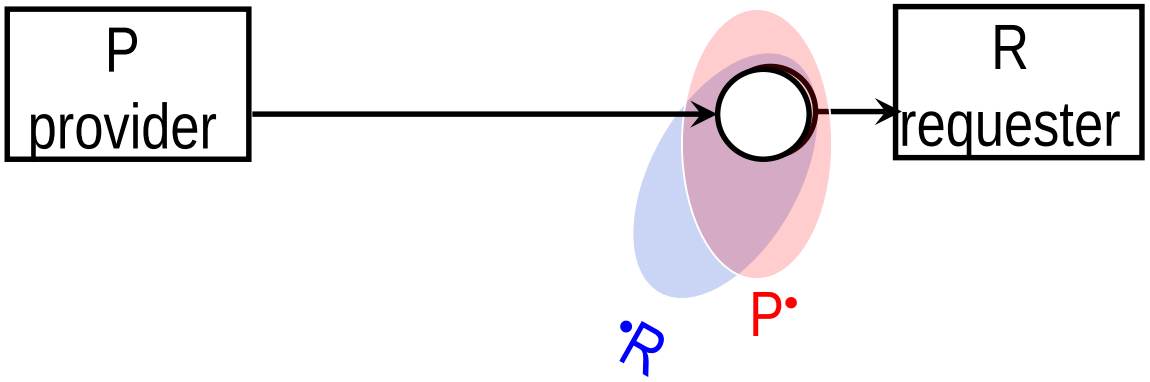
1. SOC exceeds classical Theoretical Informatics
2. Services are made to be composed!
3. Required: mechanisms to compose *many* services.
4. Composition must retain or guarantee *properties.*
5. Composition is surprisingly expressive!

left and right Ports may overlap!

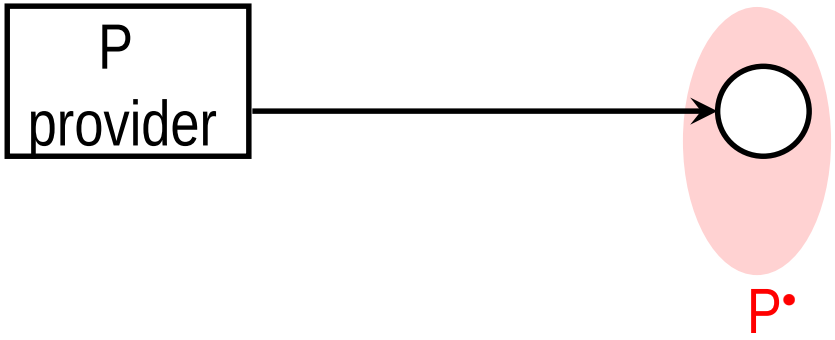


exclusive requester

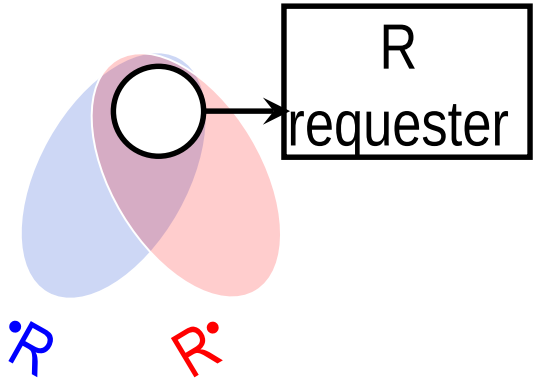
a variant:



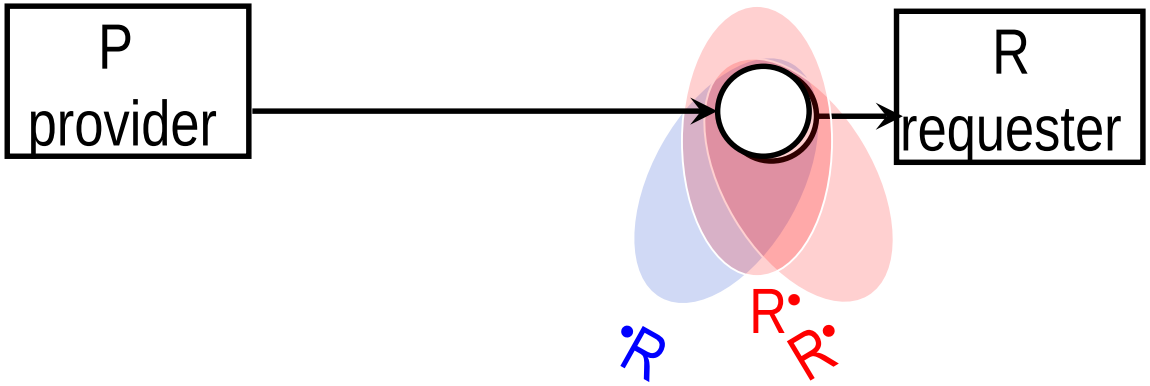
overlapping ports



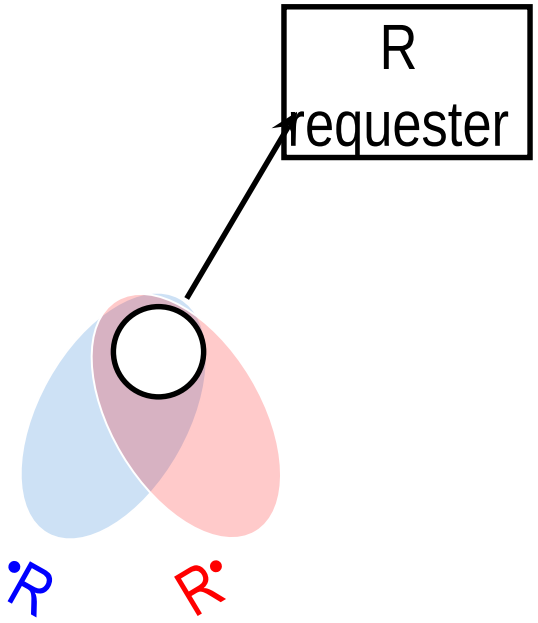
a variant:

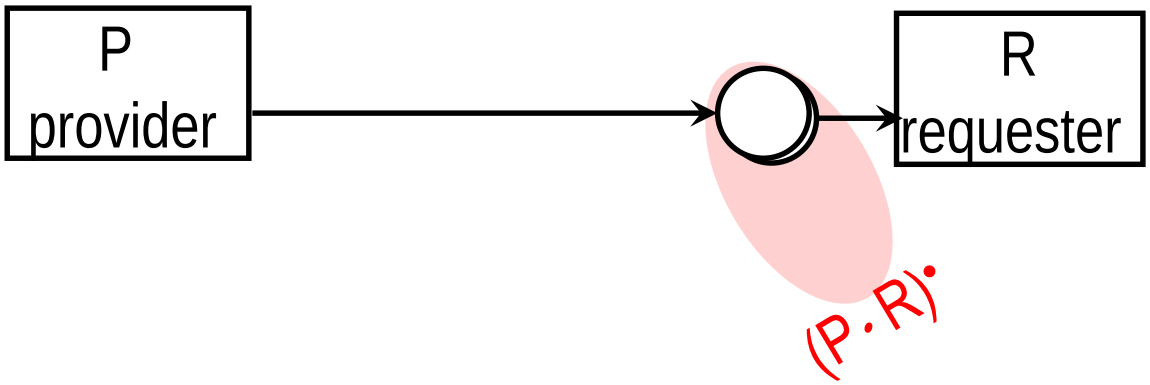


sharing requester

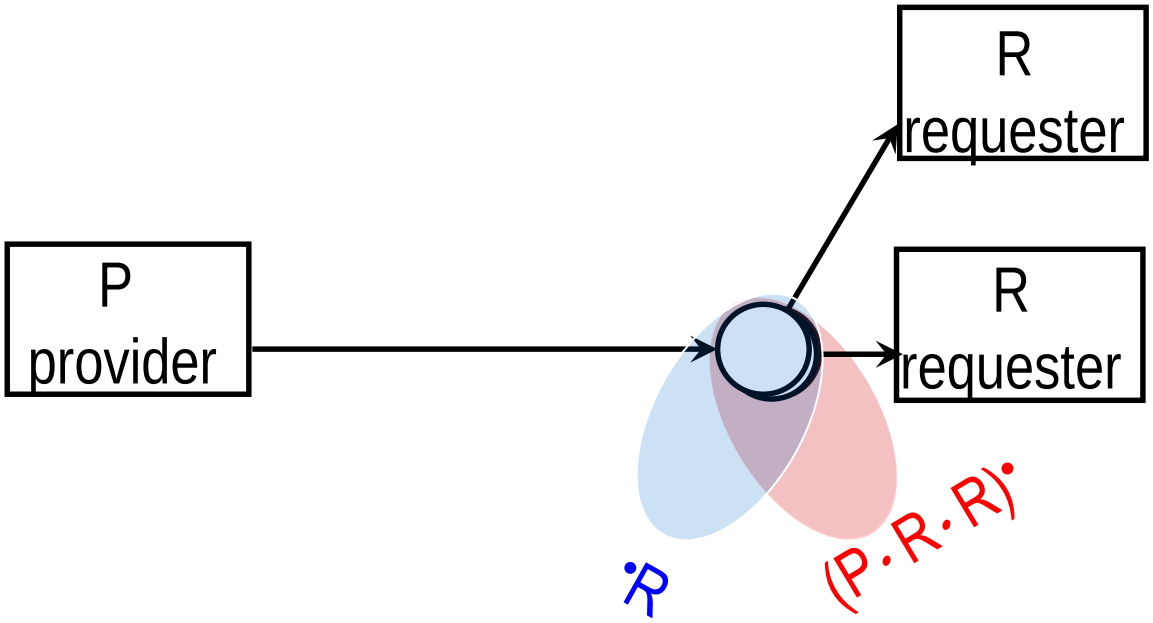


second requester

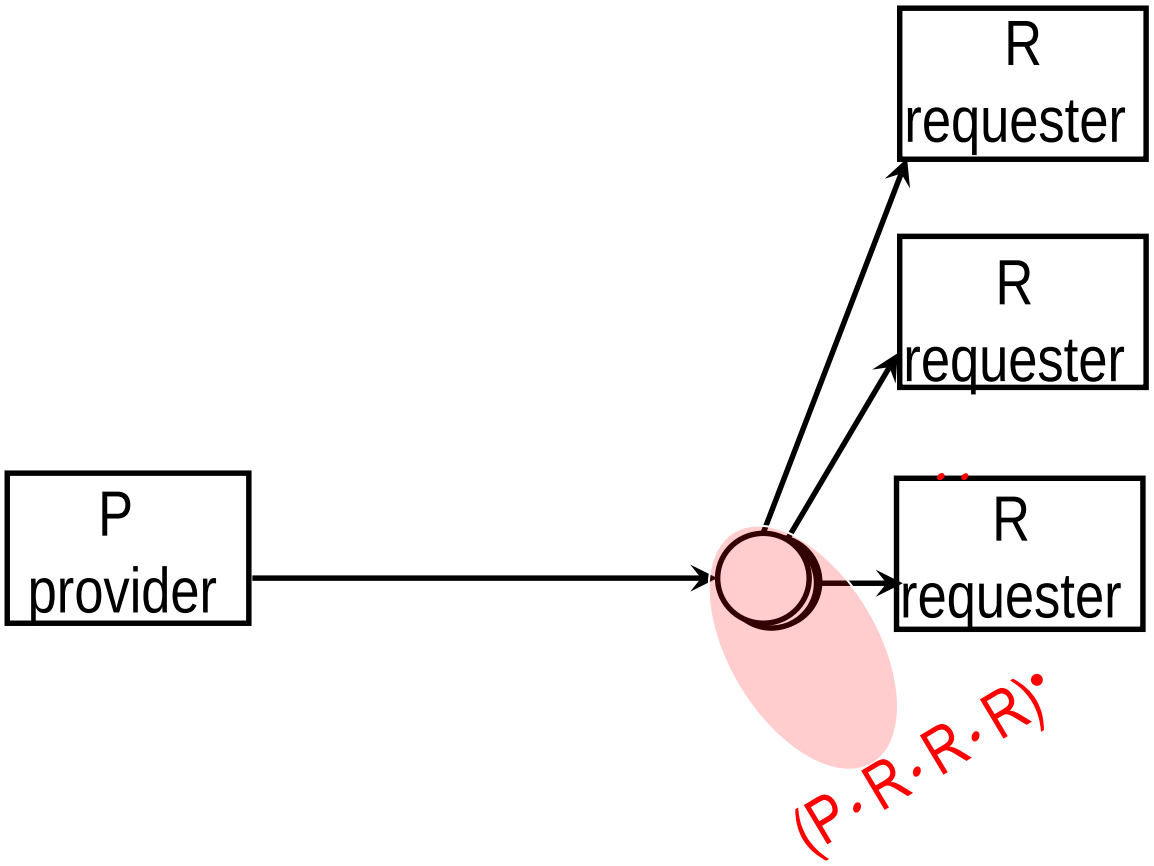




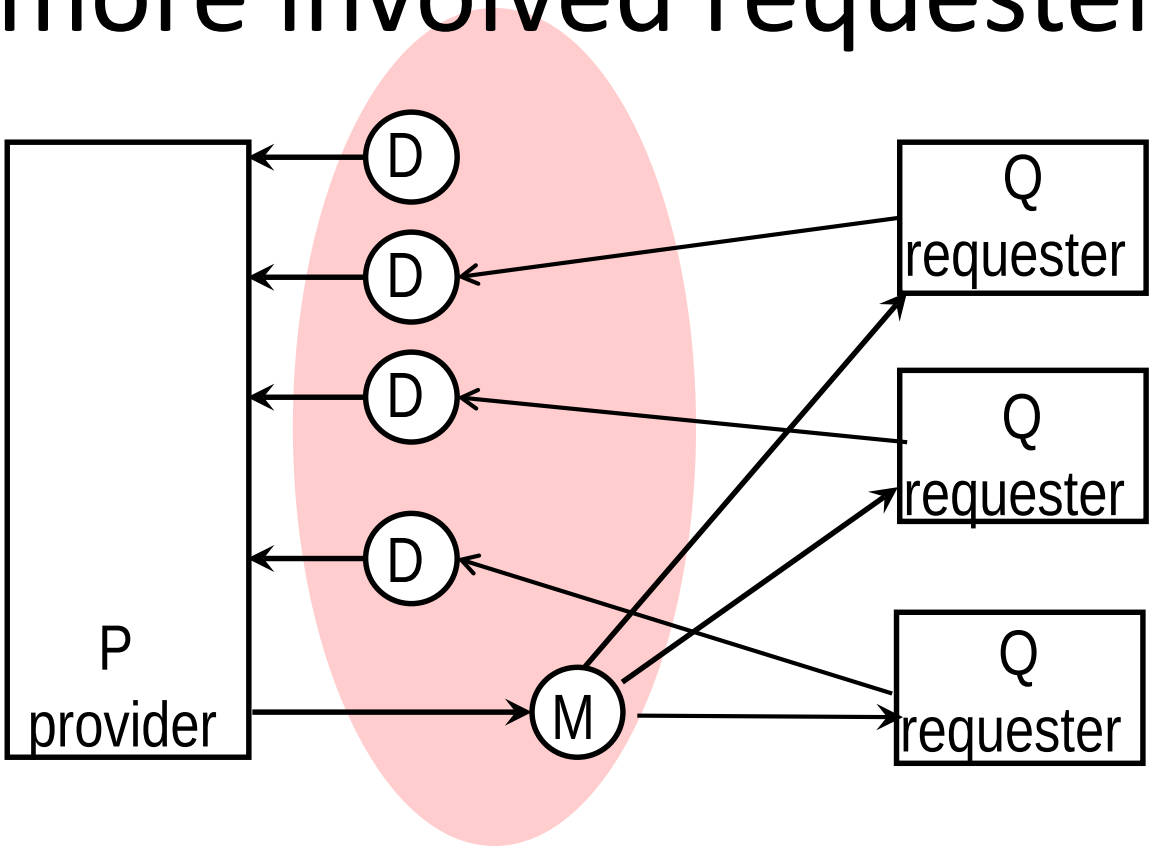
second sharing requester



third requester



more involved requester

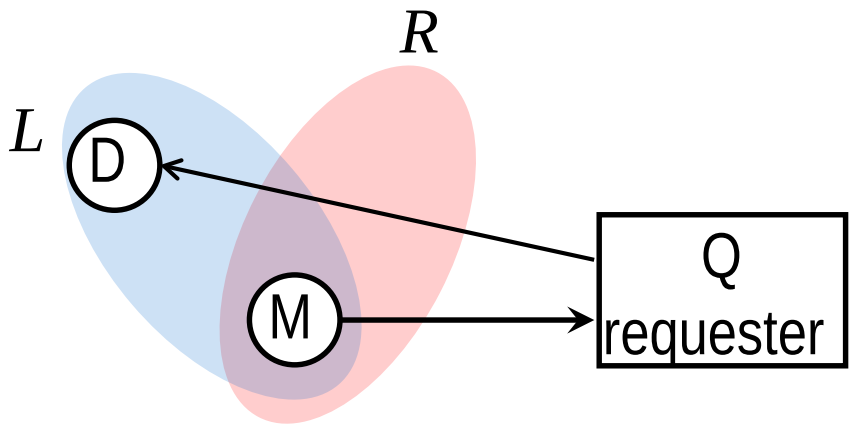


P · Q · Q · Q

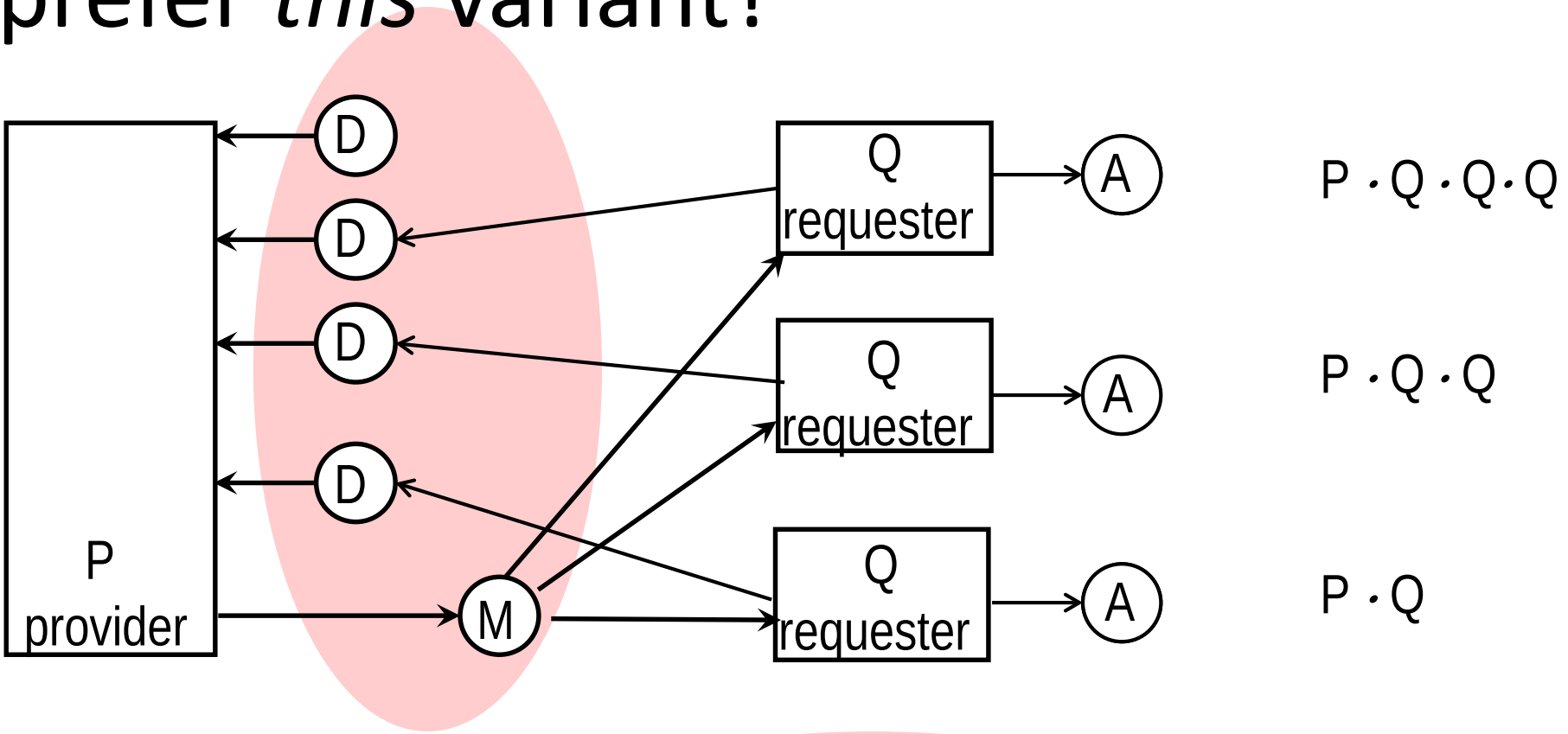
P · Q · Q

P · Q

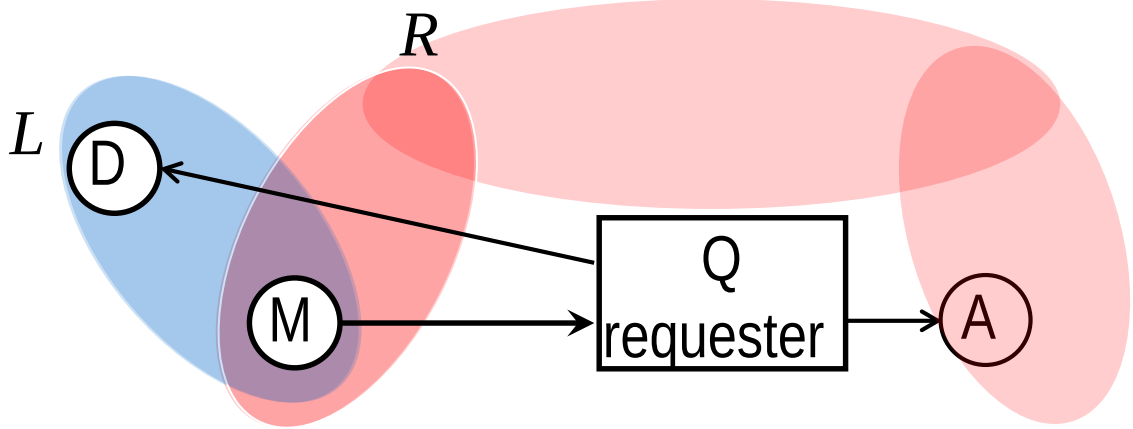
generic reques-ter Q :



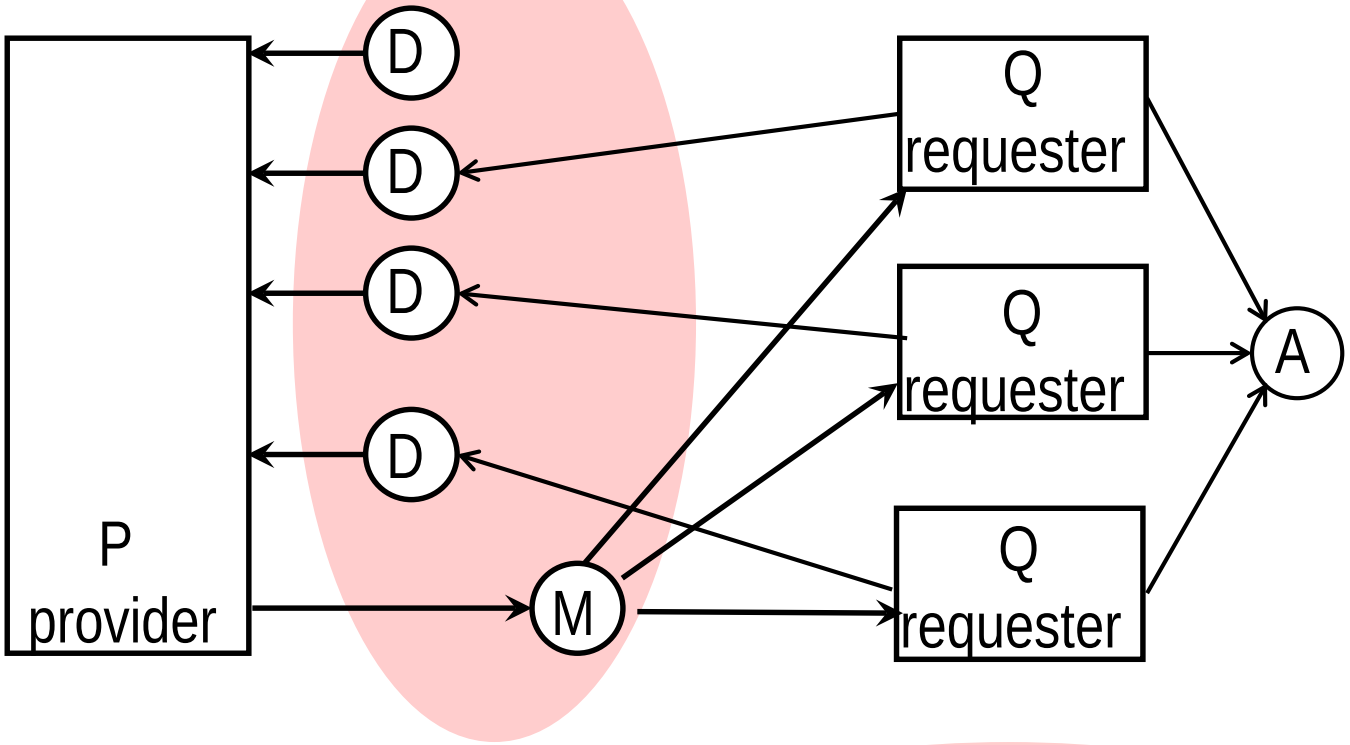
prefer *this* variant?



generic requester Q :



prefer *this* variant?

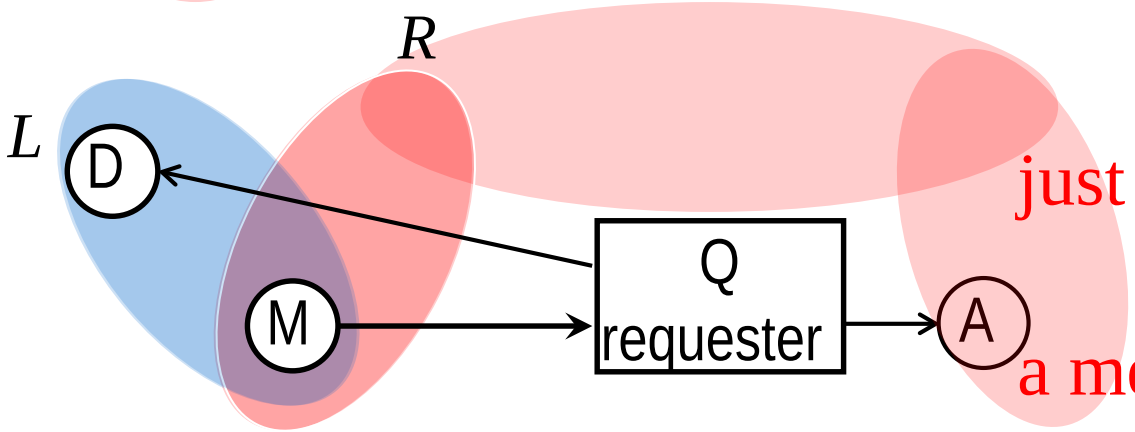


P · Q · Q · Q

P · Q · Q

P · Q

generic
reques
ter Q :



just make

a member of L

The algebraic structure of services

Given:

- a set $\mathfrak{S}$ of *services,*
- an associative operator $\oplus : \mathfrak{S} \times \mathfrak{S} \to \mathfrak{S}$,
- a set Q of *requirements* $\rho_1, \ldots, \rho_n \subseteq \mathfrak{S}$.

associative

This yields the algebraic structure

$$(\mathfrak{S}; \oplus, Q).$$

The algebraic structure of services

This yields the algebraic structure

$(\mathfrak{S}; \oplus, \mathbb{Q})$.

For $S, T \in \mathfrak{S}$, $\rho \in \mathbb{Q}$,

T is a ρ - *partner* of S,

iff $S \oplus T \models \rho$.

Let $\mathrm{sem}_\rho(S) =_{\mathrm{def}}$ the set of all ρ - partners of S.

the "classical" requirement ρ :

soundnss

derived notions

(w.r.t some ρ):

S may be *substituted by S' :*

$\mathrm{sem}_\rho(S) \subseteq \mathrm{sem}_\rho(S')$

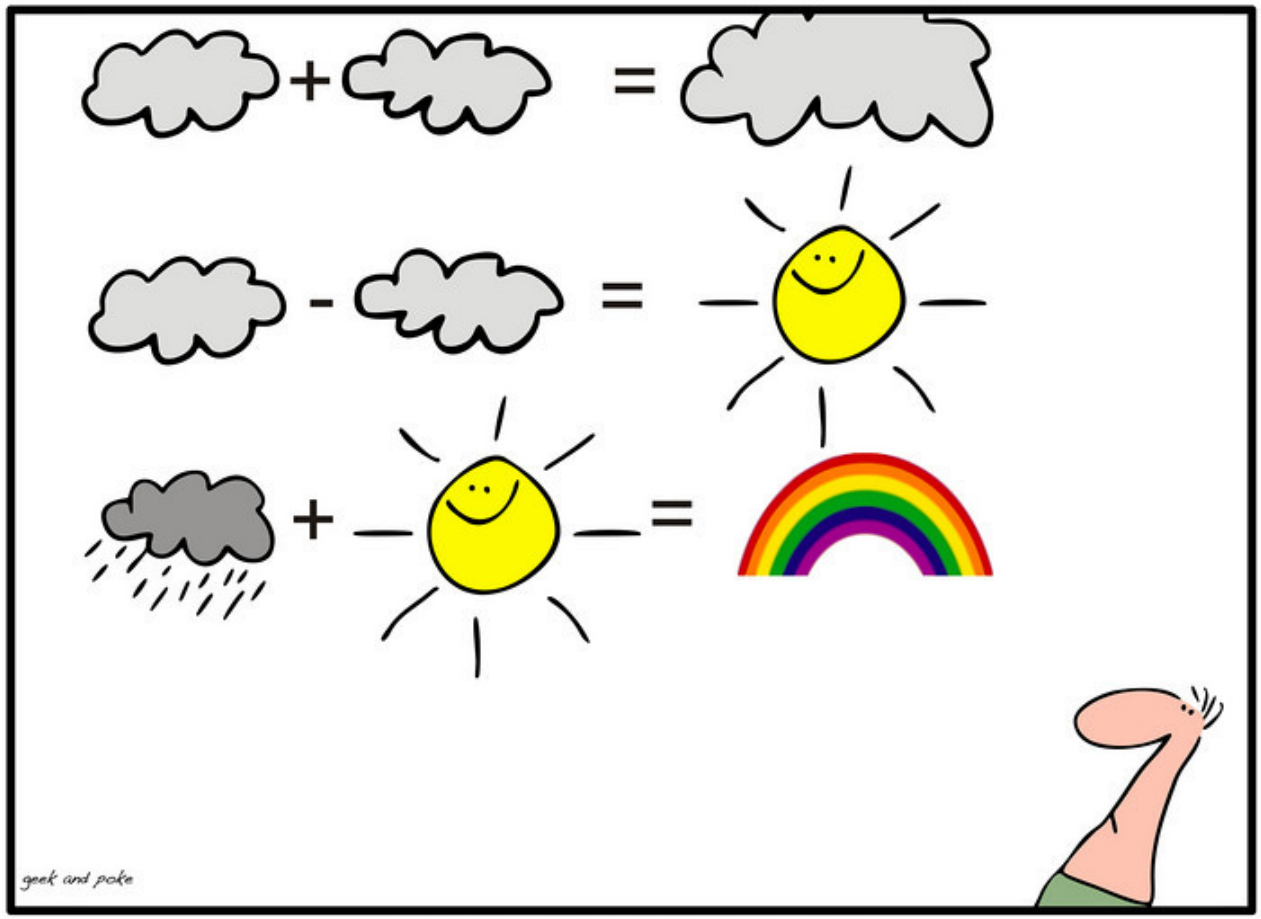
S and T *are equivalent:*

$\mathrm{sem}_\rho(S) = \mathrm{sem}_\rho(T)$

U adapts S and T*:*

$S \oplus U \oplus T \models \rho$

A generalization:
The algebraic structure of clouds

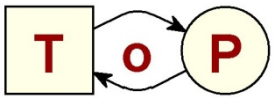


SUMMERSOC

Hersonissos, Wednesday, June 28, 2017. 9.30 – 10.30

Tutorial
Conceptual Foundations of SOC

1. **SOC exceeds classical Theoretical Informatics**
2. **Services are made to be composed!**
3. **Required: mechanisms to compose *many* services.**
4. **Composition must retain or guarantee *properties.***
5. **Composition is surprisingly expressive!**



Theory of Programming

Prof. Dr. W. Reisig

Wolfgang Reisig

Humboldt-Universität zu Berlin