Ludwig-Maximilians-Universität München
Lehrstuhl für Datenbanksysteme und Data Mining
Prof. Dr. Thomas Seidl

Knowledge Discovery and Data Mining 1

(Data Mining Algorithms 1)

Wintersemester 2019/20

Agenda

Agenda

Motivation

- Process models are generated eigther normative or descriptive
  - **Normative**: - invented by human

    - represent how a certain process is supposed to work
  - **Descriptive**: - created by process discovery algorithms based on log files

    - represent how a certain process is actually running

Process Discovery Algorithm „α-Miner“[1]

Idea: A simple algorithm to visualize processes

Input: Event log L over activities A

Output: marked petri net / Workflow net

1. Detect log-based ordering relations from event Log L
2. Create Footprint Table
3. Execute the algorithm of the α-Miner
4. Derive the WF-net

[1] van der Aalst, W M P and Weijters, A J M M and Maruster, L (2004). "Workflow Mining: Discovering process models from event logs", *IEEE Transactions on Knowledge and Data Engineering*, vol 16

Process Discovery Algorithm „α-Miner"

Let L be an event log over activities A, and let $a, b \in A$.

1. *Detect log-based ordering relations from event Log L*

   i. „(direct) following"-relation $\boldsymbol{a >_L b}$
   $\Leftrightarrow \exists\ trace\ \ \sigma = \langle t_1, t_2, t_3, \ldots, t_{n-1} \rangle\ and\ i \in \{1,2, \ldots, n-2\}$
   $\text{s. t. } \sigma \in L$ and $t_i = a$ and $t_{i+1} = b$ and $t_{i+1} = b$.

   ii. „potential parallelism" $\boldsymbol{a \parallel_L b}$
   $\Leftrightarrow a >_L b\ and\ b >_L a$

   iii. „sequential task"-relation $\boldsymbol{a \rightarrow_L b}$
   $\Leftrightarrow a >_L b\ and\ b \not>_L a$

   iv. „not followed"-relation $\boldsymbol{a \#_L b}$
   $\Leftrightarrow a \not>_L b\ and\ b \not>_L a$

$L = [\langle a, c, d \rangle^3, \langle a, d, c \rangle^2, \langle b, c, d \rangle^2, \langle b, d, c \rangle^4]$

2. Create Footprint Table:
 i) Find the directly followed tupels

	a	b	c	d
a				
b				
c				
d				

$>_L: \{(a, c), (a, d), (b, c), (b, d), (c, d), (d, c)\}$

Process Discovery Algorithm „α-Miner“

Let L be an event log over activities A, and let $a, b \in A$.

1. *Detect log-based ordering relations from event Log L*

   i. „(direct) following“-relation $\boldsymbol{a >_L b}$
   $\Leftrightarrow \exists\ trace\ \ \sigma = \langle t_1, t_2, t_3, \dots, t_{n-1} \rangle\ and\ i \in \{1,2,\dots,n-2\}$
   $\text{s. t. } \sigma \in L \text{ and } t_i = a \text{ and } t_{i+1} = b \text{ and } t_{i+1} = b.$

   ii. „potential parallelism“ $\boldsymbol{a\ \|_L\ b}$
   $\Leftrightarrow a >_L b\ and\ b >_L a$

   iii. „ sequential task“-relation $\boldsymbol{a \rightarrow_L b}$
   $\Leftrightarrow a >_L b\ and\ b \not>_L a$

   iv. „not followed“-relation $\boldsymbol{a \#_L b}$
   $\Leftrightarrow a \not>_L b\ and\ b \not>_L a$

$L\ =\ [\langle a,c,d\rangle^3, \langle a,d,c\rangle^2, \langle b,c,d\rangle^2, \langle b,d,c\rangle^4]$

2. Create Footprint Table:
 ii) Find the potential parallel tupels and mark them in the table

	a	b	c	d
a				
b				
c				$\boldsymbol{\|_L}$
d			$\boldsymbol{\|_L}$	

$>_L: \{(a,c),(a,d),(b,c),(b,d),(c,d),(d,c)\}$
$\|_L$: {(c,d), (d,c)}

Process Discovery Algorithm „α-Miner“

Let L be an event log over activities A, and let $a, b \in A$.

1. *Detect log-based ordering relations from event Log L*

   i. „(direct) following“-relation $\boldsymbol{a >_L b}$
   $\Leftrightarrow \exists\ trace\ \ \sigma = \langle t_1, t_2, t_3, \dots, t_{n-1} \rangle\ and\ i \in \{1,2, \dots, n-2\}$
   s. t. $\sigma \in L$ and $t_i = a$ and $t_{i+1} = b$ and $t_{i+1} = b$.

   ii. „potential parallelism“ $\boldsymbol{a \parallel_L b}$
   $\Leftrightarrow a >_L b\ and\ b >_L a$

   iii. „sequential task“-relation $\boldsymbol{a \rightarrow_L b}$
   $\Leftrightarrow a >_L b\ and\ b \not>_L a$

   iv. „not followed“-relation $\boldsymbol{a \#_L b}$
   $\Leftrightarrow a \not>_L b\ and\ b \not>_L a$

$L = [\langle a,c,d \rangle^3, \langle a,d,c \rangle^2, \langle b,c,d \rangle^2, \langle b,d,c \rangle^4]$

2. Create Footprint Table:
iii) Find the sequential task tupels
and mark them in the table

	a	b	c	d
a			$\rightarrow_L$	$\rightarrow_L$
b			$\rightarrow_L$	$\rightarrow_L$
c				$\parallel_L$
d			$\parallel_L$	

$>_L: \{(a,c), (a,d), (b,c), (b,d), (c,d), (d,c)\}$
$\parallel_L$: {(c,d), (d,c)}
$\rightarrow_L: \{(a,c), (a,d), (b,c), (b,d)\}$

Process Discovery Algorithm „α-Miner“

Let L be an event log over activities A, and let $a, b \in A$.

1. *Detect log-based ordering relations from event Log L*

   i. „(direct) following“-relation $\boldsymbol{a >_L b}$
   $\Leftrightarrow \exists\ trace\ \ \sigma = \langle t_1, t_2, t_3, \dots, t_{n-1}\rangle\ and\ i \in \{1,2,\dots,n-2\}$
   s. t. $\sigma \in L$ and $t_i = a$ and $t_{i+1} = b$ and $t_{i+1} = b$.

   ii. „potential parallelism“ $\boldsymbol{a\ \|_L\ b}$
   $\Leftrightarrow a >_L b\ and\ b >_L a$

   iii. „ sequential task“-relation $\boldsymbol{a \rightarrow_L b}$
   $\Leftrightarrow a >_L b\ and\ b \not>_L a$

   iv. „not followed“-relation $\boldsymbol{a \#_L b}$
   $\Leftrightarrow a \not>_L b\ and\ b \not>_L a$

$L = [\langle a, c, d\rangle^3, \langle a, d, c\rangle^2, \langle b, c, d\rangle^2, \langle b, d, c\rangle^4]$

2. Create Footprint Table:
iv) Find the not followed tupels and mark them in the table

	a	b	c	d
a	$\#_L$	$\#_L$	$\rightarrow_L$	$\rightarrow_L$
b	$\#_L$	$\#_L$	$\rightarrow_L$	$\rightarrow_L$
c			$\#_L$	$\|_L$
d			$\|_L$	$\#_L$

$>_L$: $\{(a,c),(a,d),(b,c),(b,d),(c,d),(d,c)\}$
$\|_L$: {(c,d), (d,c)}
$\rightarrow_L$: $\{(a,c),(a,d),(b,c),(b,d)\}$
$\#_L$: {(a,a),(a,b),(b,a),(b,b),(c,c),(d,d)}

Process Discovery Algorithm „α-Miner“

Let L be an event log over activities A, and let $a, b \in A$.

1. *Detect log-based ordering relations from event Log L*

   i. „(direct) following“-relation $\boldsymbol{a >_L b}$
   $\Leftrightarrow \exists\ trace\ \ \sigma = \langle t_1, t_2, t_3, \ldots, t_{n-1}\rangle\ and\ i \in \{1,2,\ldots,n-2\}$
   s. t. $\sigma \in L$ and $t_i = a$ and $t_{i+1} = b$ and $t_{i+1} = b$.

   ii. „potential parallelism“ $\boldsymbol{a \parallel_L b}$
   $\Leftrightarrow a >_L b\ and\ b >_L a$

   iii. „ sequential task“-relation $\boldsymbol{a \rightarrow_L b}$
   $\Leftrightarrow a >_L b\ and\ b \not>_L a$

   iv. „not followed“-relation $\boldsymbol{a \#_L b}$
   $\Leftrightarrow a \not>_L b\ and\ b \not>_L a$

$L\ = [\langle a,c,d\rangle^3, \langle a,d,c\rangle^2, \langle b,c,d\rangle^2, \langle b,d,c\rangle^4]$

2. Create Footprint Table:
(v) The remaining tupels represent a „directly before“ relation, marked as $\boldsymbol{\leftarrow_L}$ and mark them in the table

	a	b	c	d
a	$\#_L$	$\#_L$	$\rightarrow_L$	$\rightarrow_L$
b	$\#_L$	$\#_L$	$\rightarrow_L$	$\rightarrow_L$
c	$\boldsymbol{\leftarrow_L}$	$\boldsymbol{\leftarrow_L}$	$\#_L$	$\parallel_L$
d	$\boldsymbol{\leftarrow_L}$	$\boldsymbol{\leftarrow_L}$	$\parallel_L$	$\#_L$

$>_L: \{(a,c),(a,d),(b,c),(b,d),(c,d),(d,c)\}$
$\parallel_L$: {(c,d), (d,c)}
$\rightarrow_L: \{(a,c),(a,d),(b,c),(b,d)\}$
$\#_L$: {(a,a),(a,b),(b,a),(b,b),(c,c),(d,d)}

Process Discovery Algorithm „α-Miner“

3. Execute the algorithm of the α-Miner

   i) All activities that start any trace yield the set of **starting activities,** collected in $\boldsymbol{T_{in}}$.

   ii) All activities that end any trace yield the set of **output activities,** $\boldsymbol{T_{out}}$.

   …

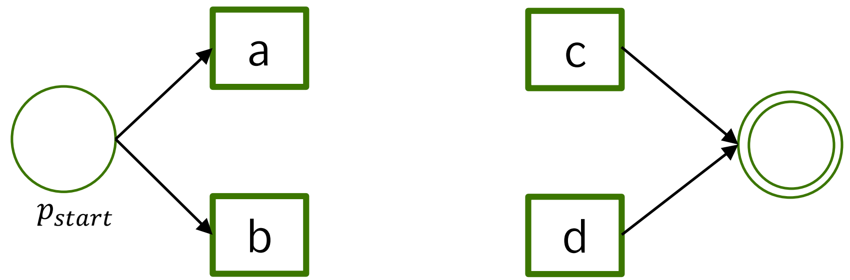
4. Derive the WF-net:

- The set of **transitions** is equal to A, so each activity represents a transition
- A starting place is created and connected to each node in $\boldsymbol{T_{in}}$.
- Also, a final place is created and each node in $\boldsymbol{T_{out}}$ is connected to it.

$L = [\langle a,c,d\rangle^3, \langle a,d,c\rangle^2, \langle b,c,d\rangle^2, \langle b,d,c\rangle^4]$

$T_{in} = \{a, b\}$

$T_{out} = \{c, d\}$



Process Discovery Algorithm „α-Miner“

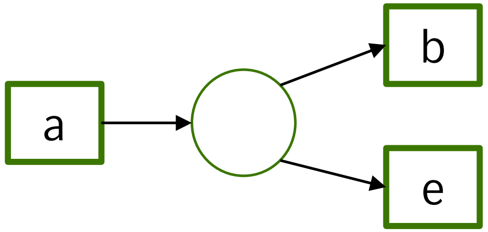
3. Execute the algorithm of the α-Miner …

   iii) Determine all pairs of sets A and B, such that

   - $\forall a_1, a_2 \in A: a_1 \# a_2$
   - $\forall b_1, b_2 \in B: b_1 \# b_2$
   - $\forall a_1 \in A, \forall b_1 \in B: a_1 \rightarrow b_1$
   - Select only the "maximal pairs":
     e.g. $(\{a\},\{c\}), (\{a\},\{d\}), (\{a\},\{c,d\}) \Longrightarrow (\{a\},\{c,d\})$

4. A place is added in between A and B and connected accordingly

e.g.$A = \{a\}, B = \{b, e\}$

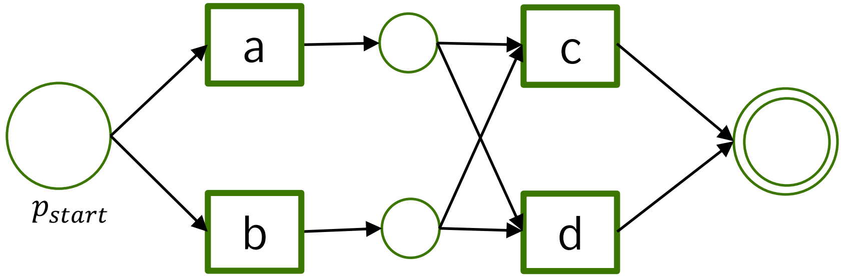


Heuristics-Miner is our first algorithm to capture concurrent process behavior.

	a	b	c	d
a	$\#_L$	$\#_L$	$\rightarrow_L$	$\rightarrow_L$
b	$\#_L$	$\#_L$	$\rightarrow_L$	$\rightarrow_L$
c	$\leftarrow_L$	$\leftarrow_L$	$\#_L$	$\parallel_L$
d	$\leftarrow_L$	$\leftarrow_L$	$\parallel_L$	$\#_L$

valid set of „maximal pairs“:

$(\{a\},\{c,d\}), (\{b\},\{c,d\})$



Process Discovery Algorithm „Heuristics-Miner“ [2]

Idea: α-Miner has several flaws (1-loops, 2-loops, no weighting).

Heuristics-Miner uses dependency as the condition to connect activities.

Input: Event log L

Output: Causal net, *here we stop at the dependency graph*

[2] Weijters, A. J. M. M., Wil MP van Der Aalst, and AK Alves De Medeiros. "Process mining with the heuristics miner-algorithm." *Technische Universiteit Eindhoven, Tech. Rep. WP* 166 (2006): 1-34.

Process Discovery Algorithm „Heuristics-Miner“

Let L be an event log over activities A, and let $a, b \in A$.

1. Create table displaying frequency of „directly follows“ relation $>_{\boldsymbol{L}}$

$$L = [\langle a,e\rangle^5, \langle a,b,c,e\rangle^{10}, \langle a,c,b,e\rangle^{10}, \langle a,b,e\rangle^{10}, \langle a,d,d,e\rangle^2, \langle a,d,d,d,e\rangle^1]$$

$>_L$	a	b	c	d	e
a	0	11	11	13	5
b	0	0	10	0	11
c	0	10	0	0	11
d	0	0	0	4	13
e	0	0	0	0	0

Process Discovery Algorithm „Heuristics-Miner“

2. Create a table showing the value of „dependency measures“ of all pairs of activities over L

$$|a \Rightarrow_L b| = \begin{cases} \dfrac{|a >_L b| - |b >_L a|}{|a >_L b| + |b >_L a| + 1} & \text{, if a} \neq b \\ \dfrac{|a > a|}{|a > a| + 1} & , if\ a = b \end{cases}$$

$|a \Rightarrow_L b| \in]-1,1[$

$|a \Rightarrow_L b| = 0$, if $|a >_L b| = |b >_L a|$

$|a \Rightarrow_L b| \rightarrow 1$, if a follows almost always after b

$>_L$	a	b	c	d	e
a	0	11	11	13	5
b	0	0	10	0	11
c	0	10	0	0	11
d	0	0	0	4	13
e	0	0	0	0	0

$\lvert\Rightarrow_L\rvert$	a	b	c	d	e
a					
b					
c					
d					
e					

Process Discovery Algorithm „Heuristics-Miner“

2. Create a table showing the value of „dependency measures“ of all pairs of activities over L

$$|a \Rightarrow_L b| = \begin{cases} \dfrac{|a >_L b| - |b >_L a|}{|a >_L b| + |b >_L a| + 1} & \text{, if } a \neq b \\ \dfrac{|a > a|}{|a > a| + 1} & , if\ a = b \end{cases}$$

$|a \Rightarrow_L b| \in]-1,1[$

$|a \Rightarrow_L b| = 0$, if $|a >_L b| = |b >_L a|$

$|a \Rightarrow_L b| \rightarrow 1$, if a follows almost always after b

Lower triangular matrix is the negative and transposed of the upper triangular matrix.

$>_L$	a	b	c	d	e
a	0	11	11	13	5
b	0	0	10	0	11
c	0	10	0	0	11
d	0	0	0	4	13
e	0	0	0	0	0

$\|\Rightarrow_L\|$	a	b	c	d	e
a	0	0.92	0.92	0.93	0.83
b	-0.92	0	0	0	0.92
c	-0.92	0	0	0	0.92
d	-0.93	0	0	0.80	0.93
e	-0.83	-0.92	-0.92	-0.93	0

$$|a \Rightarrow_L b| = \frac{11 - 0}{11 + 0 + 1} = 0.92$$

$$|b \Rightarrow_L c| = \frac{10-10}{10+10+1} = 0$$

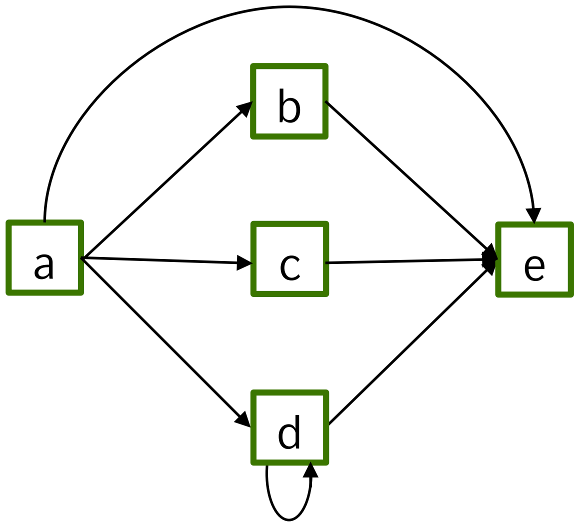
Process Discovery Algorithm „Heuristics-Miner“

3. i) Select **two thresholds** to reduce noise ($\tau_{>_L}$) and infrequent traces ($\tau_{\Rightarrow_L}$)
   ii) Create the dependency graph DG: an arc between x and y is only included if $|x <_L y| \geq \tau_{>_L} \wedge |x \Rightarrow_L y| \geq \tau_{\Rightarrow_L}$

$>_L$	a	b	c	d	e
a	0	11	11	13	5
b	0	0	10	0	11
c	0	10	0	0	11
d	0	0	0	4	13
e	0	0	0	0	0

$\lvert\Rightarrow_L\rvert$	a	b	c	d	e
a	0	0.92	0.92	0.93	0.83
b	~~-0.92~~ 0	0	0	0	0.92
c	~~-0.92~~ 0	0	0	0	0.92
d	~~-0.93~~ 0	0	0	0.80	0.93
e	~~-0.83~~ 0	~~-0.92~~ 0	~~-0.92~~ 0	~~-0.93~~ 0	0

Ex. 1:
Setting $\tau_{>_L} = 2$ and $\tau_{\Rightarrow_L} = 0.7$ yields to the following dependency graph:



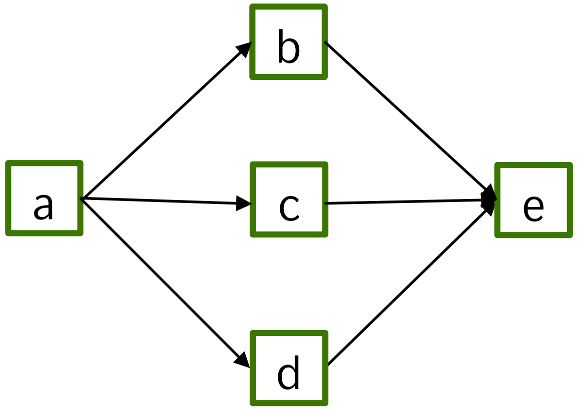
Process Discovery Algorithm „Heuristics-Miner“

3. i) Select **two thresholds** to reduce noise ($\tau_{>_L}$) and infrequent traces ($\tau_{\Rightarrow_L}$)
   ii) Create the dependency graph DG: an arc between x and y is only included if $|x <_L y| \geq \tau_{>_L} \wedge |x \Rightarrow_L y| \geq \tau_{\Rightarrow_L}$

$>_L$	a	b	c	d	e
a	0	11	11	13	5
b	0	0	10	0	11
c	0	10	0	0	11
d	0	0	0	40	13
e	0	0	0	0	0

$\lvert\Rightarrow_L\rvert$	a	b	c	d	e
a	0	0.92	0.92	0.93	~~0.83~~ 0
b	~~-0.92~~ 0	0	0	0	0.92
c	~~-0.92~~ 0	0	0	0	0.92
d	~~-0.93~~ 0	0	0	~~0.80~~ 0	0.93
e	~~-0.83~~ 0	~~-0.92~~ 0	~~-0.92~~ 0	~~-0.93~~ 0	0

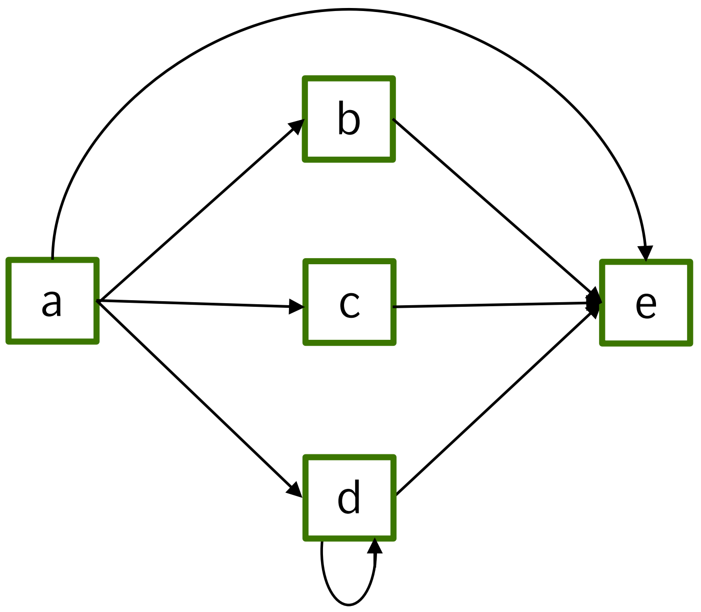
Ex. 2:
Setting $\tau_{>_L} = 5$ and $\tau_{\Rightarrow_L} = 0.9$ yields to the following dependency graph:



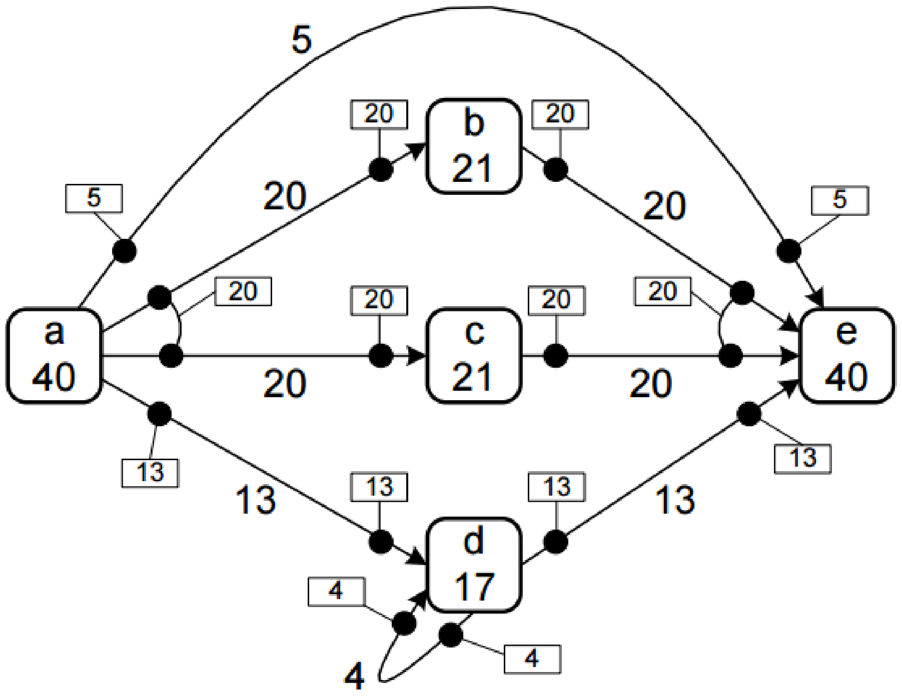
Process Discovery Algorithm „Heuristics-Miner“

4. Last step – *not in this lecture*:

   dependency graph → causal net



→



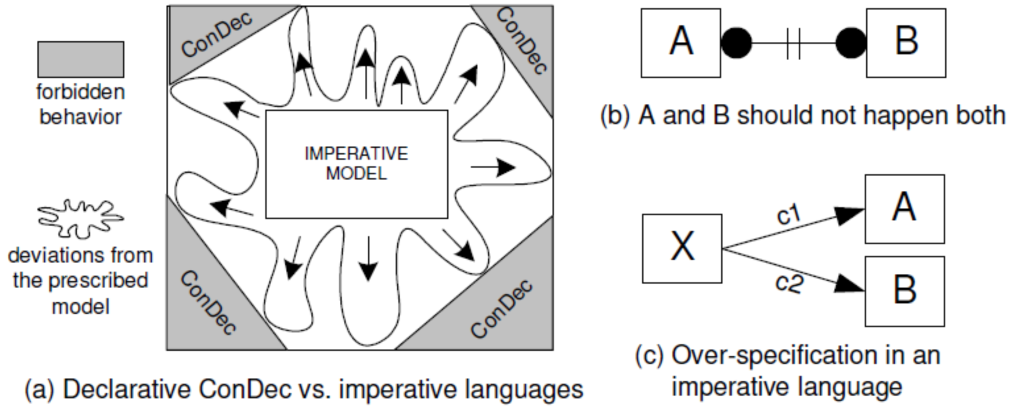
Process Discovery Algorithm – Some Others

- **„Inductive-Miner (IM)“** [3]:

  It uses the directly-follows graph that corresponds to the „direct follows“ relation ($>_L$) used by the α-Miner and creates a Process Tree Q.

- **„Declare“** [4]**:**

  It is a constrained based declarative approach.



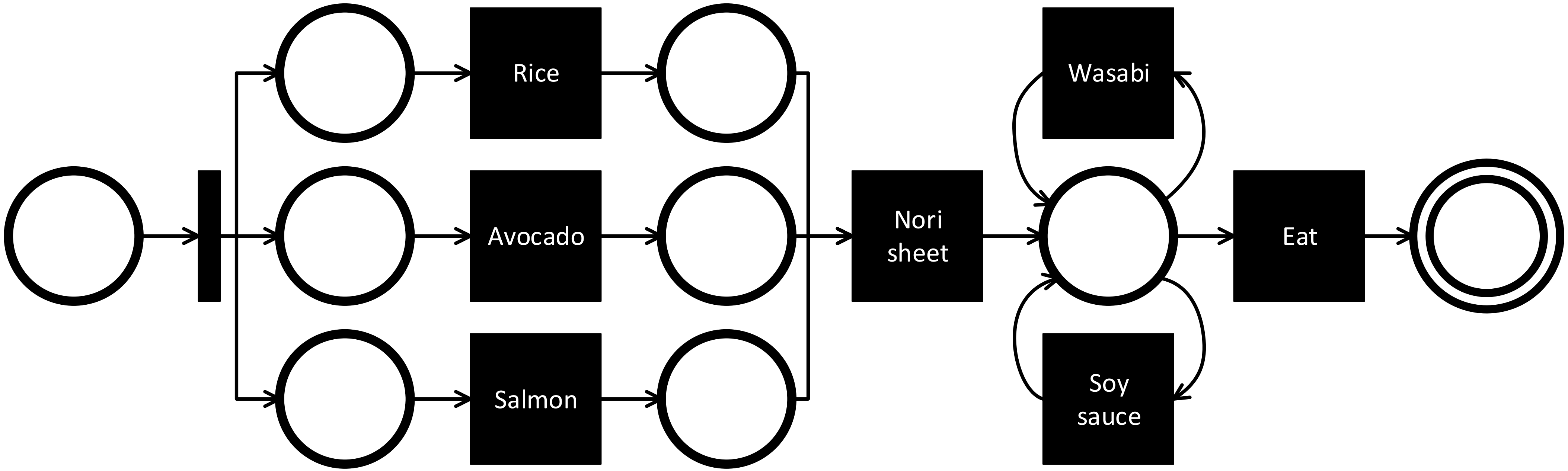
Imperative vs. Declarative approaches

[3] S.J.J. Leemans, D. Fahland, andW.M.P. van der Aalst. Discovering Block-structured Process Models from Event Logs: A Constructive Approach. In J.M. Colom and J. Desel, editors, *Applications and Theory of Petri Nets 2013*, volume 7927 of *Lecture Notes in Computer Science*, pages 311–329. Springer, Berlin, 2013.
[4]Pesic, Maja, Helen Schonenberg, and Wil MP Van der Aalst. "Declare: Full support for loosely-structured processes." *11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007)*. IEEE, 2007.

Agenda

Motivation

- Given an event log and a process model, decide for each case wether it conforms to the model or not. If not, give the issues.



<Salmon, Rice, Avocado, Nori, Eat>
<Rice, Salmon, Wasabi>
<Avocado, Rice, Soy sauce, Nori, Eat>

conform?

non-conform?

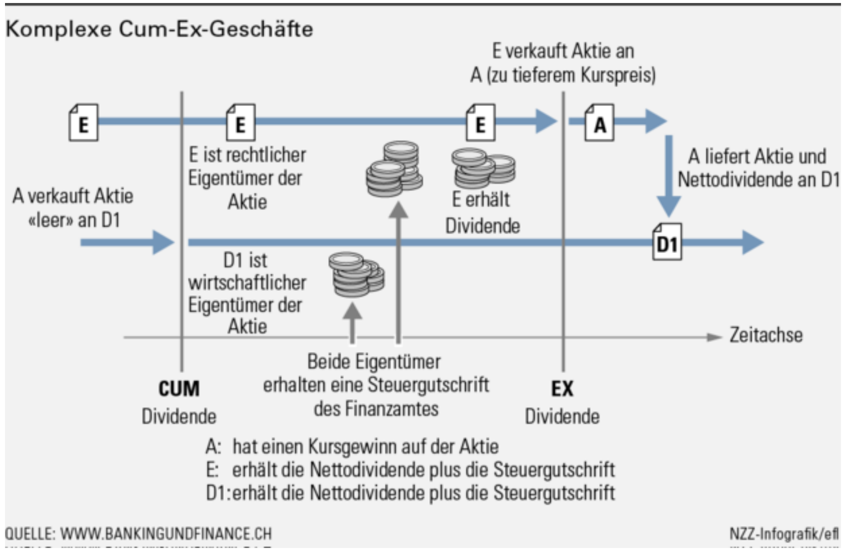
Goal: Fraud detection

- Alteration of medical treatment, usually for higher compensations ("upcoding"). Cheap medication billed as costly medication. Medication is non-conform to the treatment plan, e.g. flu vaccination after broken leg.

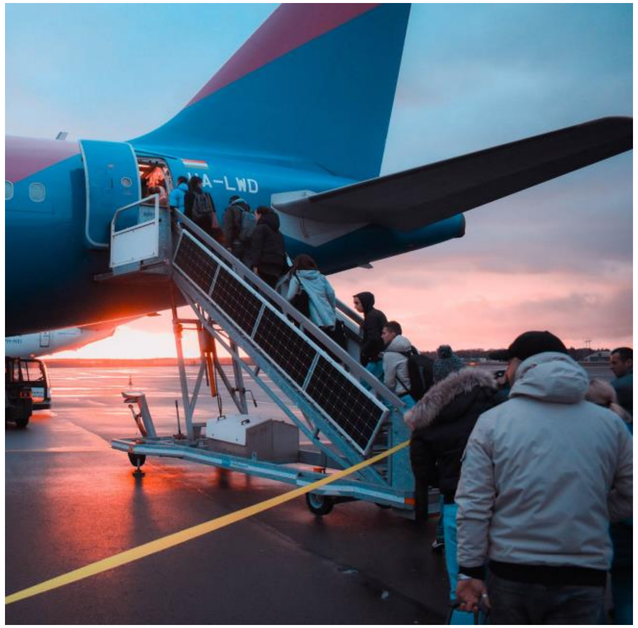
- Duplicate execution of actions.
  Billing twice for same service or good

		Key Figures	SalesOdrAmt_D
Sales Organization	Sales Order	Billing Document	
Dom. Sales Org US	388	90000324	78 EUR
		90000339	78 EUR
	389	#	19 EUR
	390	90000336	233 EUR

- Embezzlement, theft or misuse of company assets.
  Usage of company truck at suspicious times for private actions (evenings, vacation,…),
  or faked payments using complex and unusual cashflows.



Goal: Workflow improvements



- Root-cause detection
  Quality check failed for some products. Search for shared historic activities (e.g. same supplier, preprocessed by same employee or machine, similar environmental conditions).

- Standardization of deviations
  Customers are processed faster at a certain counter. How has the employee deviated the process? E.g. Families with children board first at the airport.

- Customer aggregation
  Some customers look for furniture in a popular shop. The order of furniture presentation influences their habbits. Where to offer the small items like tealights? Which customer types map to which market traversal paths?

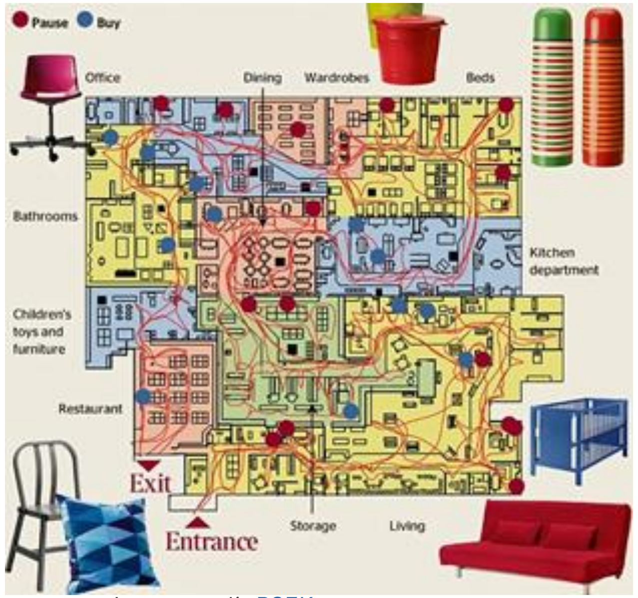


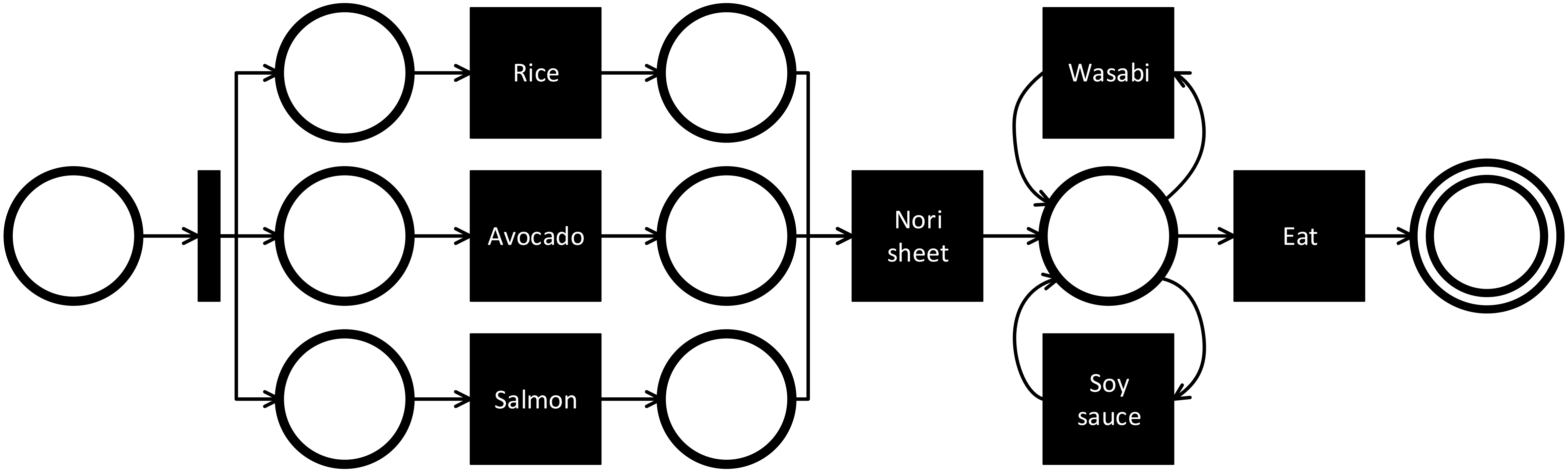
Image credit PSFK

Automata Theory: Decide Language Membership

- Idea:
  - Put a token into the start position.
  - For each event, fire the transition with the same label in the Petri net.
  - If the Petri net accepts the sequence, the trace passed the conformance checking.
  - Otherwise, a rejected trace has zero fitness.

[1] A.K. Alves de Medeiros,W.M.P. van der Aalst, and A.J.M.M.Weijters. Quantifying Process Equivalence Based on Observed Behavior. *Data and Knowledge Engineering*, **64**(1):55–74, 2008.

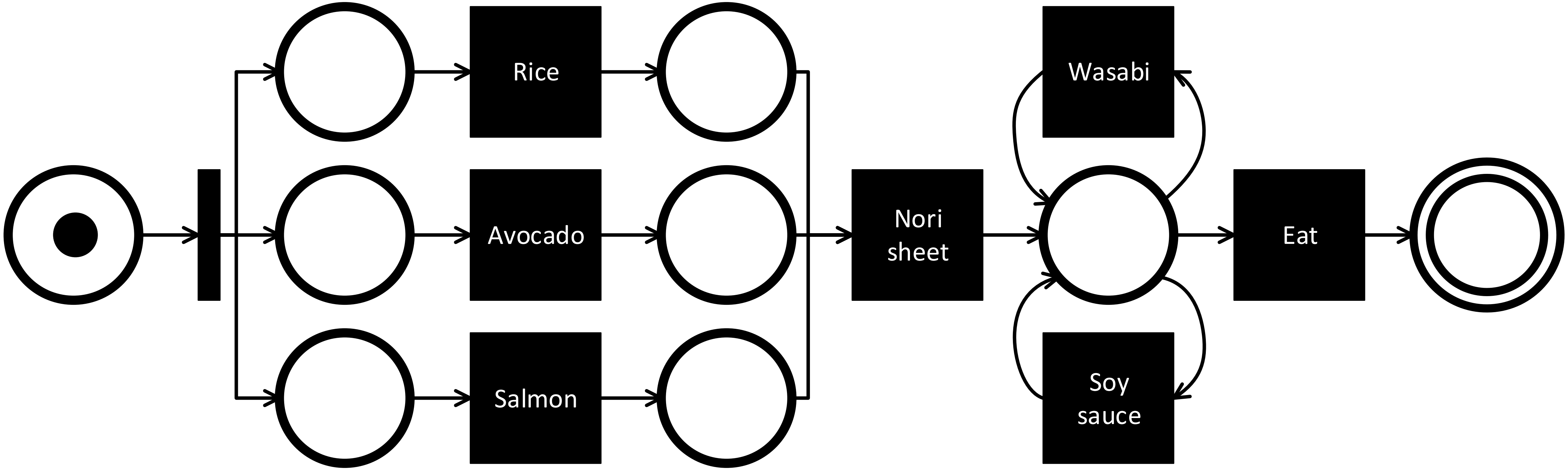
Petri Net Membership Test



Checking: <Avocado, Rice, Salmon, Nori, Eat>

(p)roduced : 0 (c)onsumed : 0

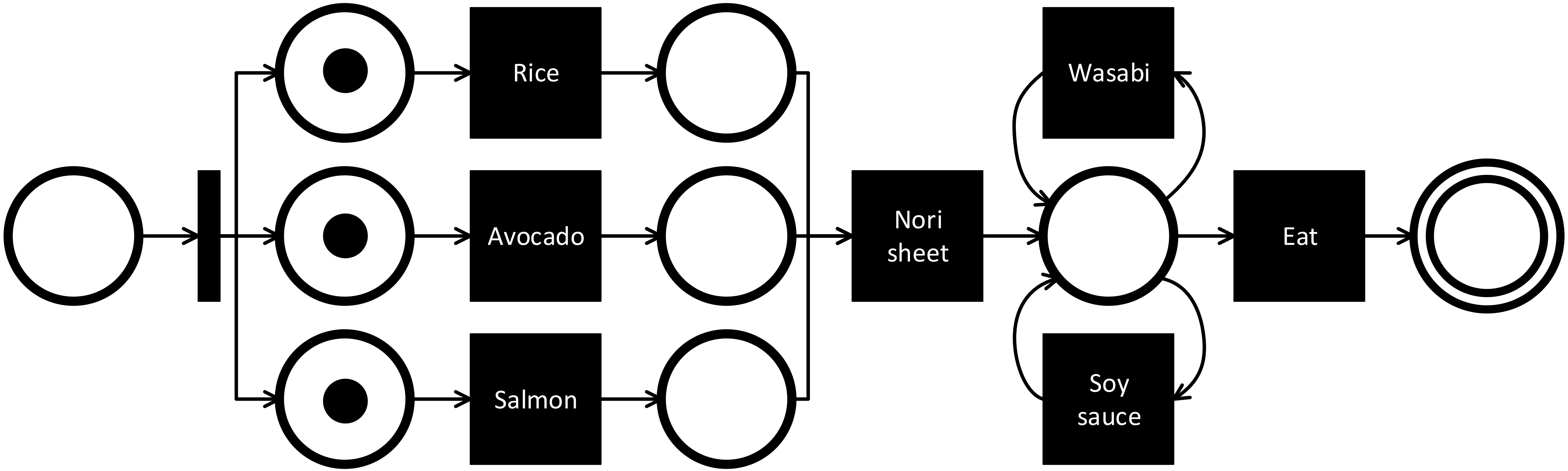
Petri Net Membership Test



Checking: <Avocado, Rice, Salmon, Nori, Eat>

(p)roduced : 1 (c)onsumed : 0

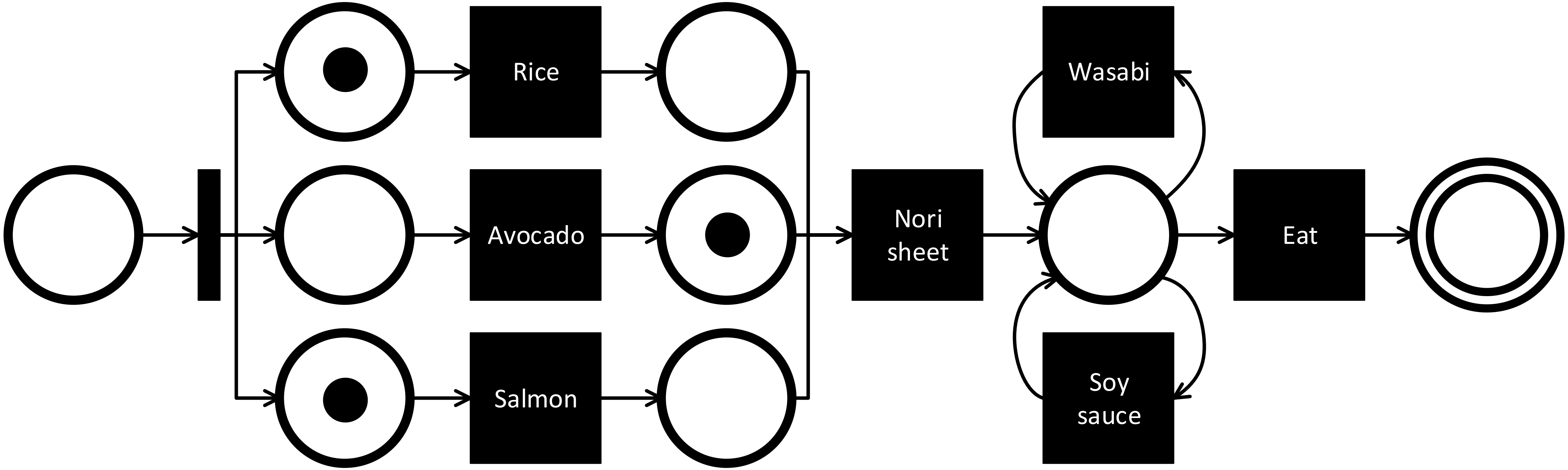
Petri Net Membership Test



Checking: <<u>Avocado</u>, Rice, Salmon, Nori, Eat>

(p)roduced : 4 (c)onsumed : 1

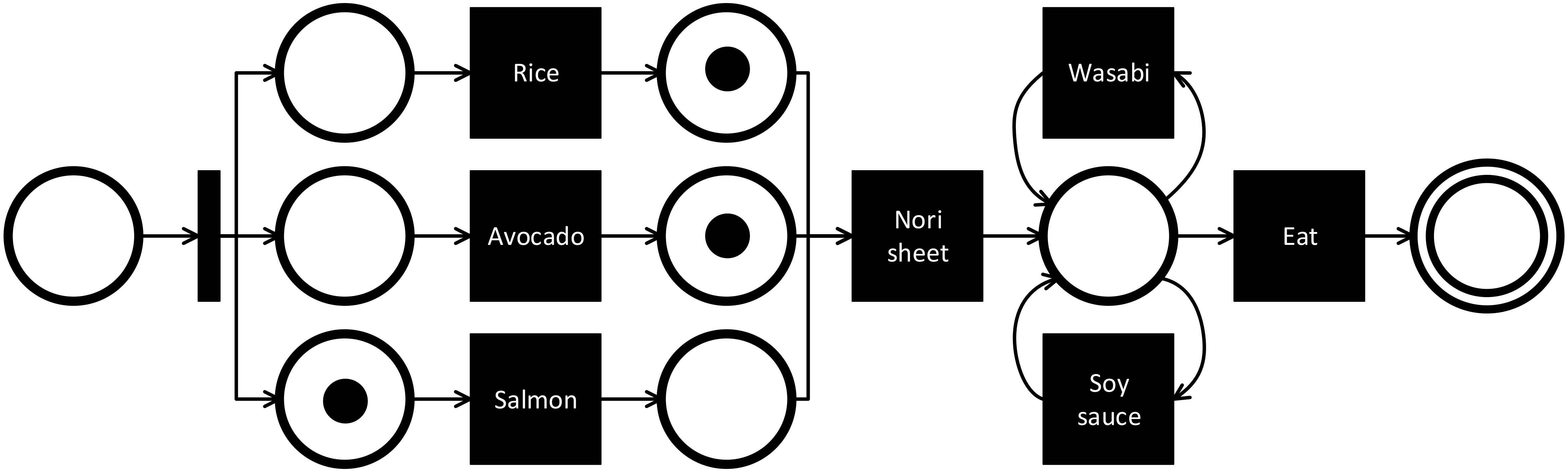
Petri Net Membership Test



Checking: <Avocado, Rice, Salmon, Nori, Eat>

(p)roduced : 5 (c)onsumed : 2

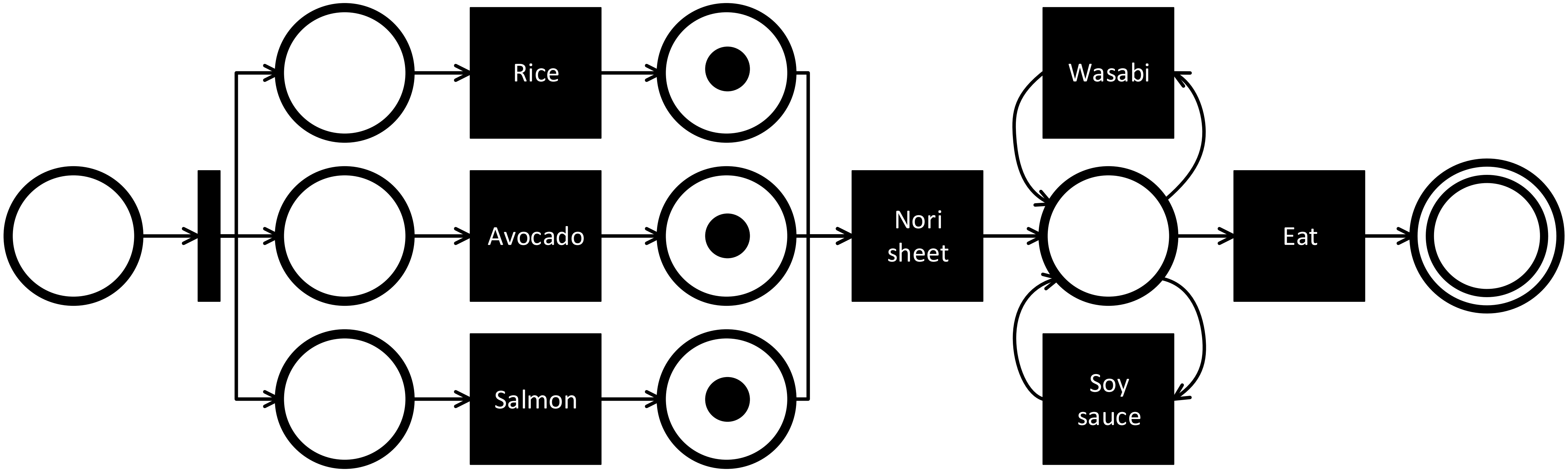
Petri Net Membership Test



Checking: <Avocado, Rice, Salmon, Nori, Eat>

(p)roduced : 6 (c)onsumed : 3

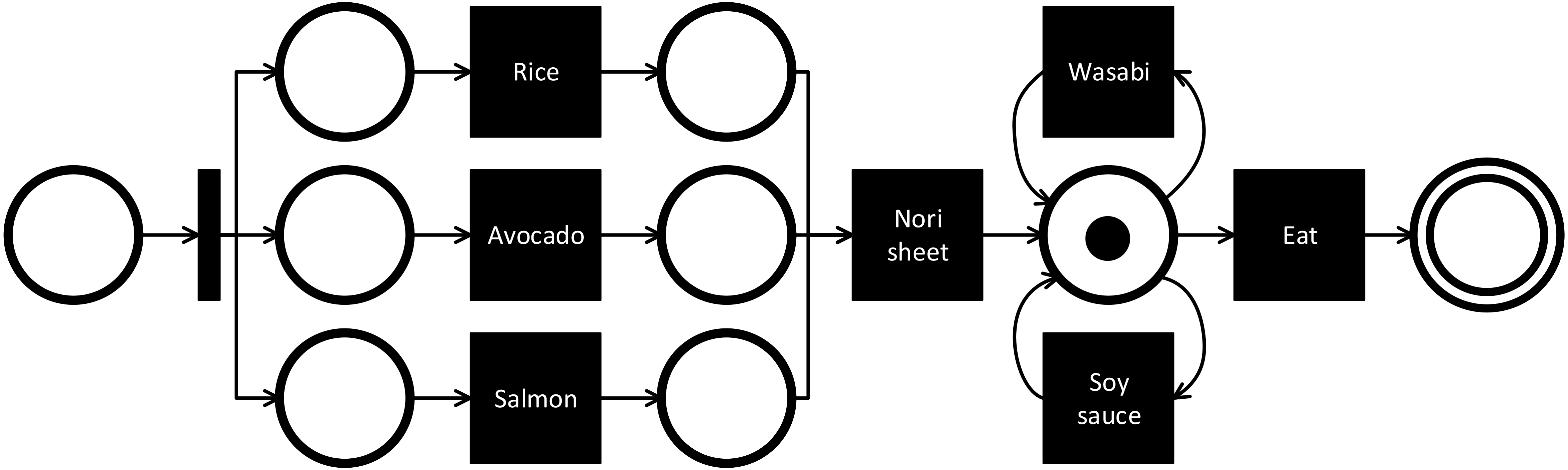
Petri Net Membership Test



Checking: <Avocado, Rice, Salmon, Nori, Eat>

(p)roduced : 7 (c)onsumed : 4

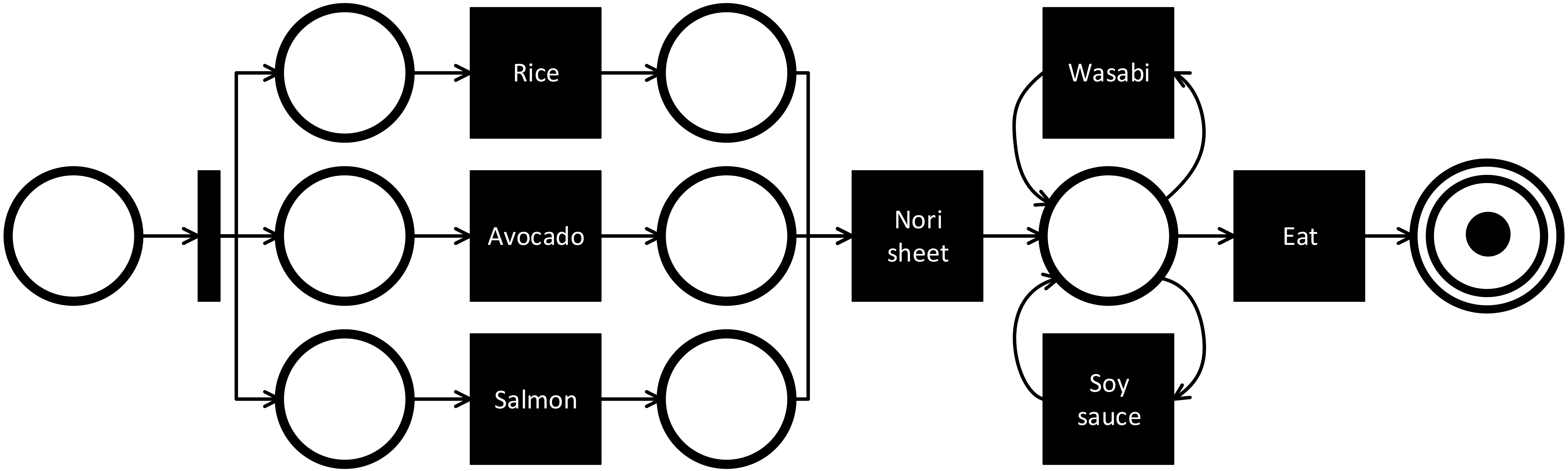
Petri Net Membership Test



Checking: <Avocado, Rice, Salmon, Nori, Eat>

(p)roduced : 8 (c)onsumed : 7

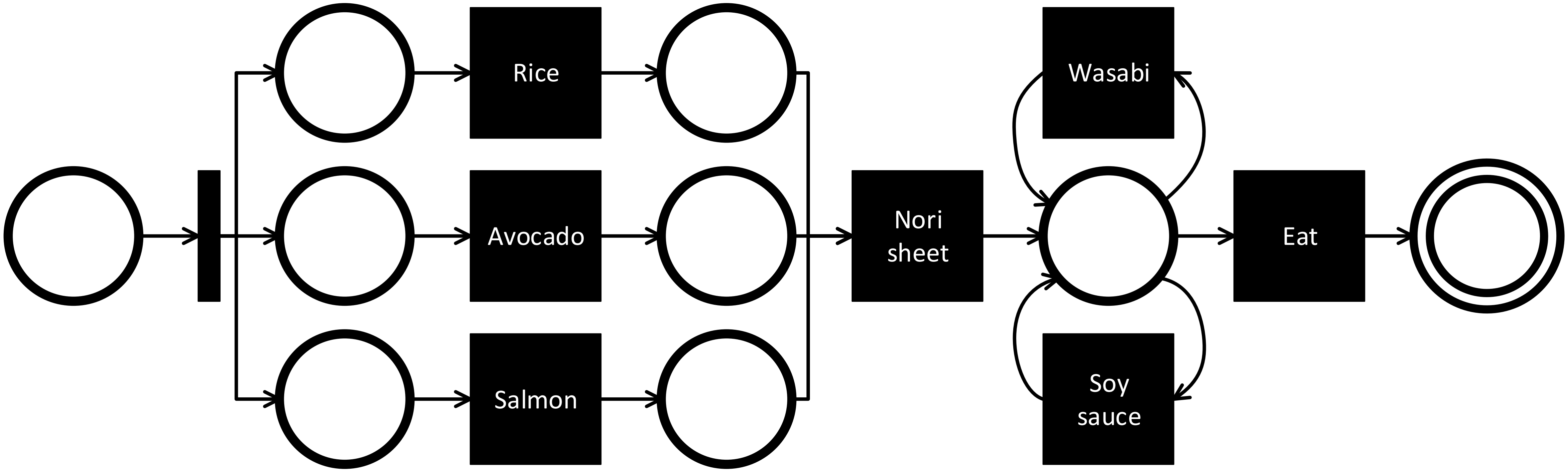
Petri Net Membership Test



Checking: <Avocado, Rice, Salmon, Nori, Eat>

(p)roduced : 9 (c)onsumed : 8

Petri Net Membership Test



Checking: <Avocado, Rice, Salmon, Nori, Eat>

(p)roduced : 9 (c)onsumed : 9

Petri Net Membership Test

The fitness of a case with trace σ on WF-net M is defined as:

$$fitness(\sigma, M) = \frac{1}{2}\left(1 - \frac{m}{c}\right) + \frac{1}{2}\left(1 - \frac{r}{p}\right)$$

Considering the example:

Checking: $\sigma =$ <Avocado, Rice, Salmon, Nori, Eat>

(p)roduced : 9 (c)onsumed : 9

$$fitness(\sigma, M) = \frac{1}{2}\left(1 - \frac{0}{9}\right) + \frac{1}{2}\left(1 - \frac{0}{9}\right) = 1$$

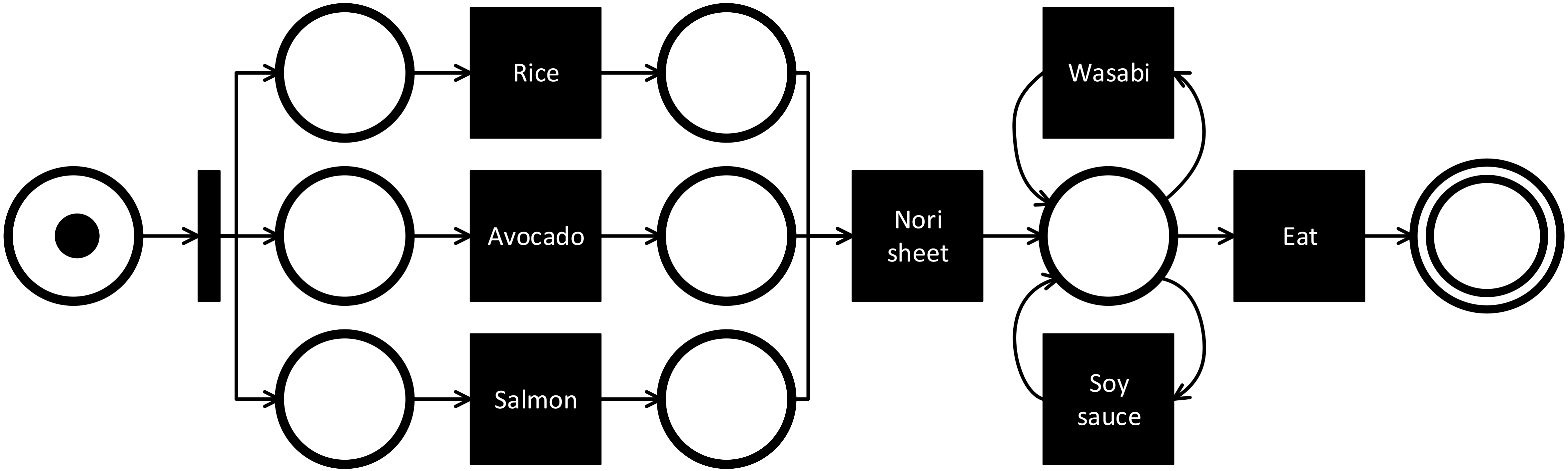
Token Replay[1]

- Problem with pure Automata approach:
  - We cannot decide between almost fit and critically deviating traces (binary classifier).
  - In practical applications we often need some flexibility to execute the processes.

- Modified Idea:
  - Put a token into the start position.
  - For each event, try to fire the corresponding transition in the net.
  - If not possible, create a virtual new token after the transition.
  - In the end, determine the fitness based on the tokens left in the model and the virtually added ones.

[1] A.K. Alves de Medeiros,W.M.P. van der Aalst, and A.J.M.M.Weijters. Quantifying Process Equivalence Based on Observed Behavior. *Data and Knowledge Engineering*, **64**(1):55–74, 2008.

Token Replay Example

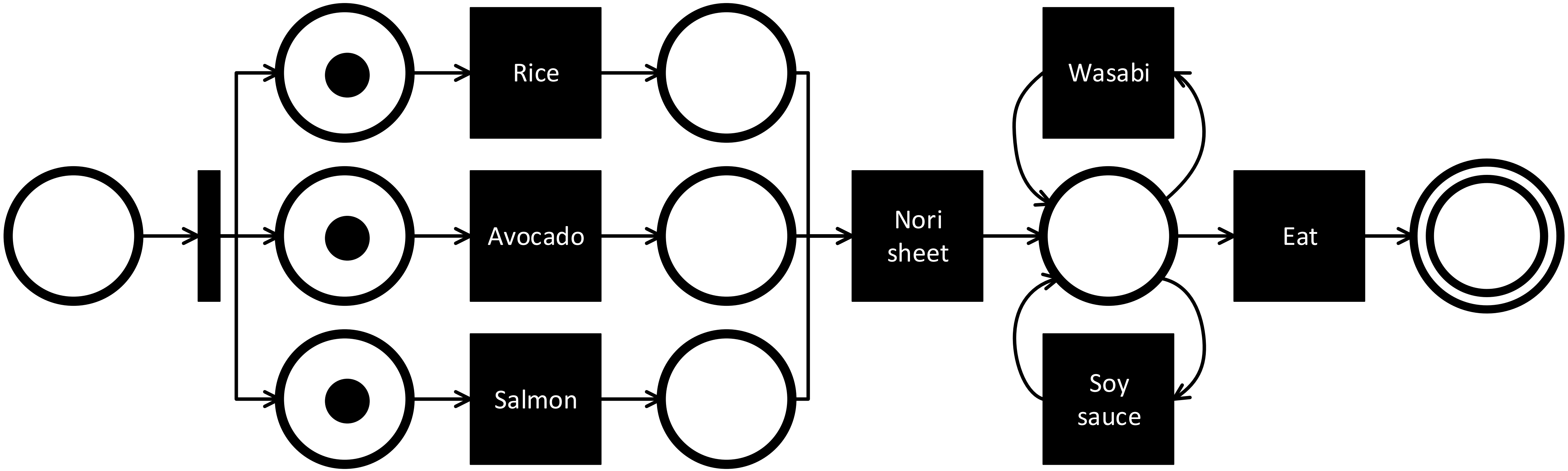


Checking: <Rice, Salmon, Wasabi>

(p)roduced : 1 (c)onsumed : 0

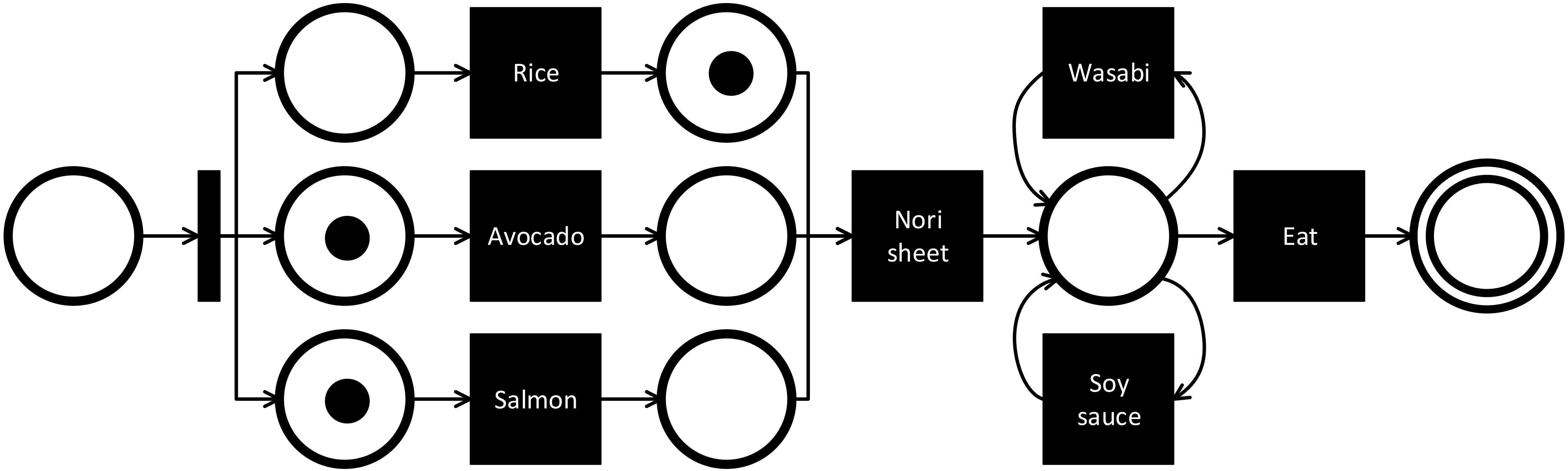
(m)issing : 0 (r)emaining : 0

Token Replay Example



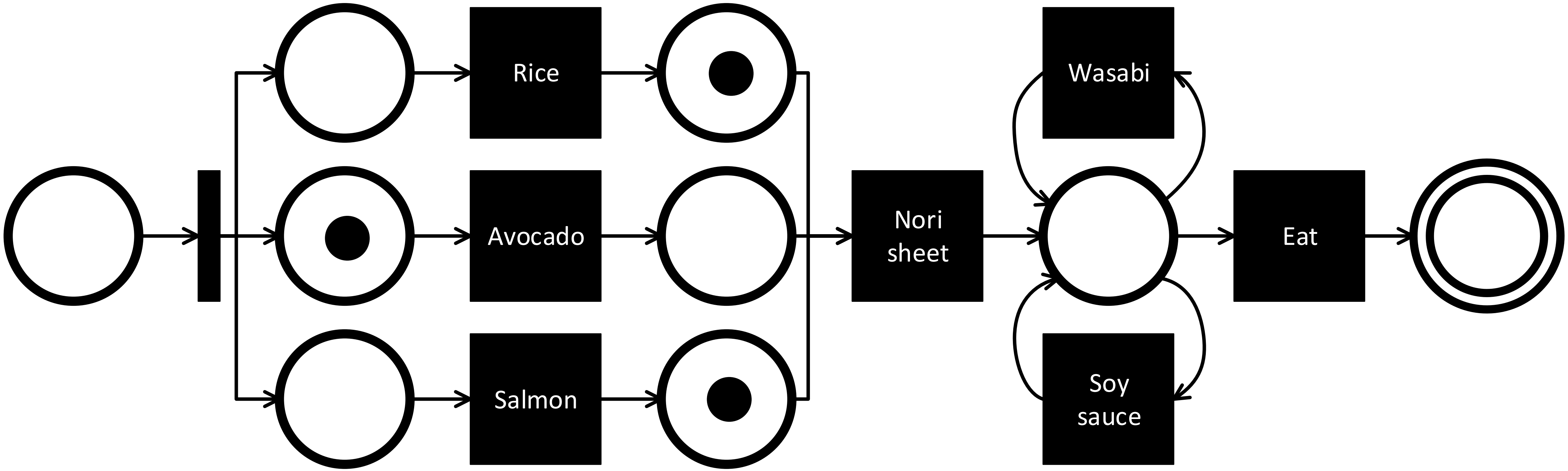
Checking:	<Rice, Salmon, Wasabi>		
(p)roduced :	4	(c)onsumed :	1
(m)issing :	0	(r)emaining :	0

Token Replay Example



Checking:	<Rice, Salmon, Wasabi>		
(p)roduced :	5	(c)onsumed :	2
(m)issing :	0	(r)emaining :	0

Token Replay Example

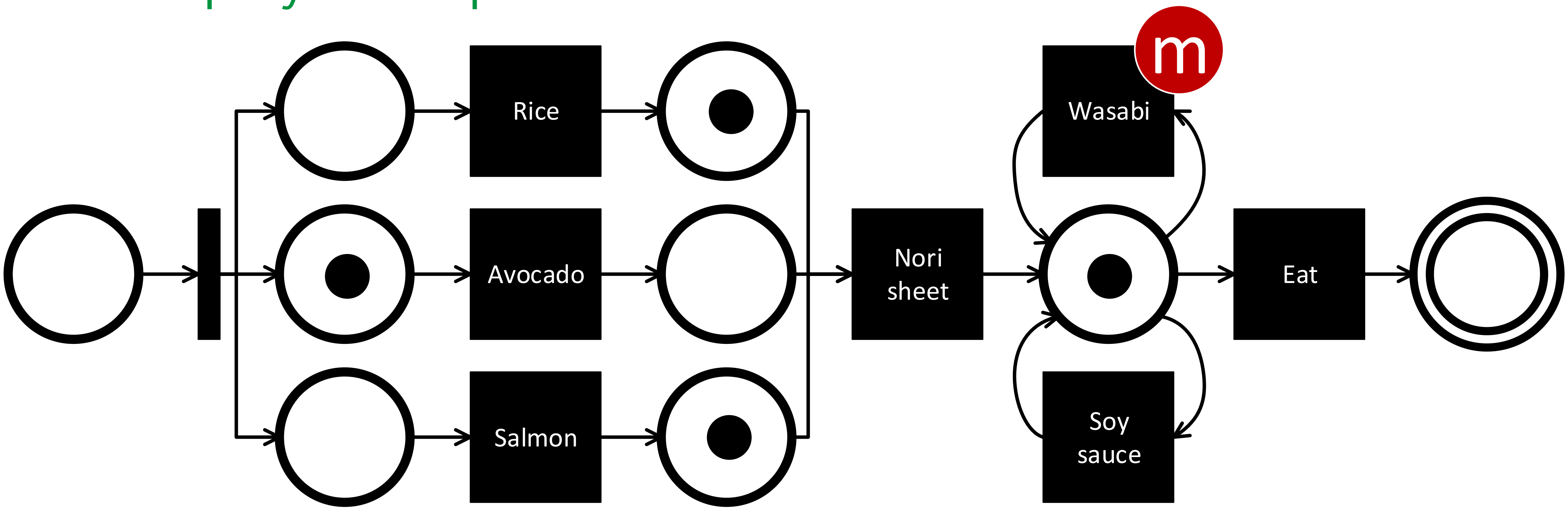


Checking: <Rice, Salmon, Wasabi>

(p)roduced : 6 (c)onsumed : 3

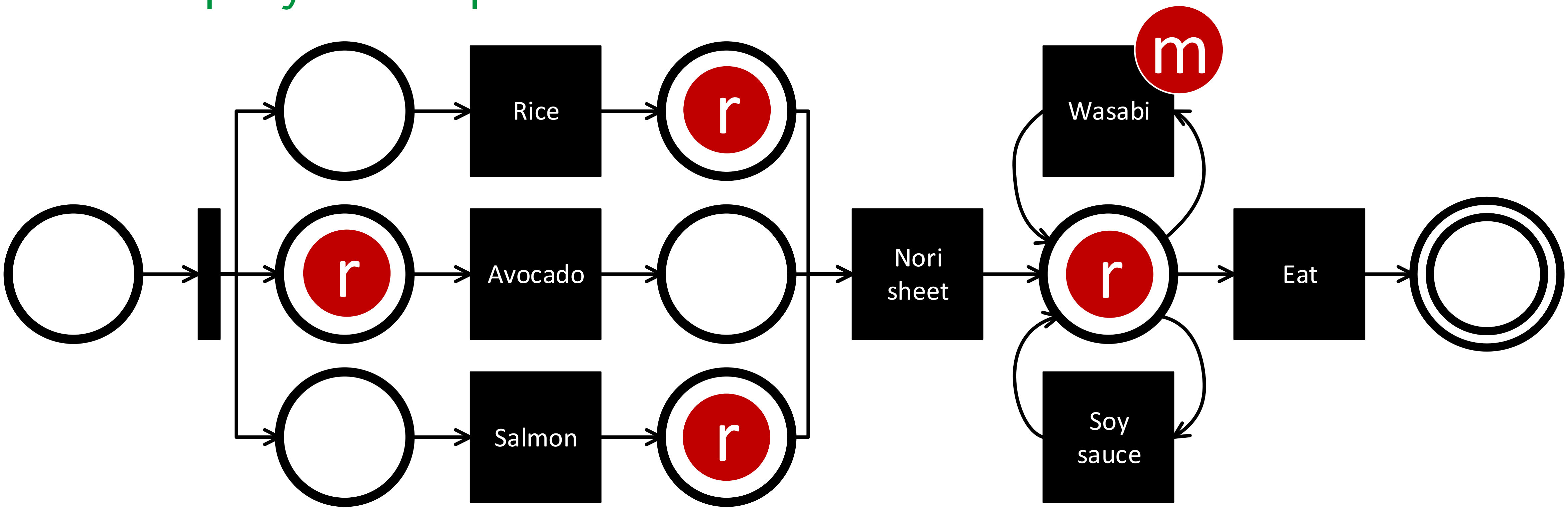
(m)issing : 0 (r)emaining : 0

Token Replay Example



Checking:	<Rice, Salmon, Wasabi>		
(p)roduced :	7	(c)onsumed :	4
(m)issing :	1	(r)emaining :	0

Token Replay Example



Checking: <Rice, Salmon, Wasabi>

(p)roduced : 7 (c)onsumed : 4

(m)issing : 1 (r)emaining : 4

Token Replay Example

The fitness of a case with trace σ on WF-net M is defined as:

$$fitness(\sigma, M) = \frac{1}{2}\left(1 - \frac{m}{c}\right) + \frac{1}{2}\left(1 - \frac{r}{p}\right)$$

Considering the example:

Checking: $\sigma =$ <Rice, Salmon, Wasabi>

(p)roduced :	7	(c)onsumed :	4
(m)issing :	1	(r)emaining :	4

$$fitness(\sigma, M) = \frac{1}{2}\left(1 - \frac{1}{4}\right) + \frac{1}{2}\left(1 - \frac{4}{7}\right) = 0{,}375$$

Token Replay: Discussion

- Allows a continuous fitness score in the interval $[0,1]$.
- Intuitive and easy to implement.
- For critical deviating behavior, model gets flooded with tokens. Earlier deviations mask later deviations.
  → all behavior afterwards gets accepted, fitness values too low
- Depending on a Petri net representation of the process.

Alignments[2]

- To overcome drawbacks of Token Replay, it might be better to map observed behavior on modelled behavior.

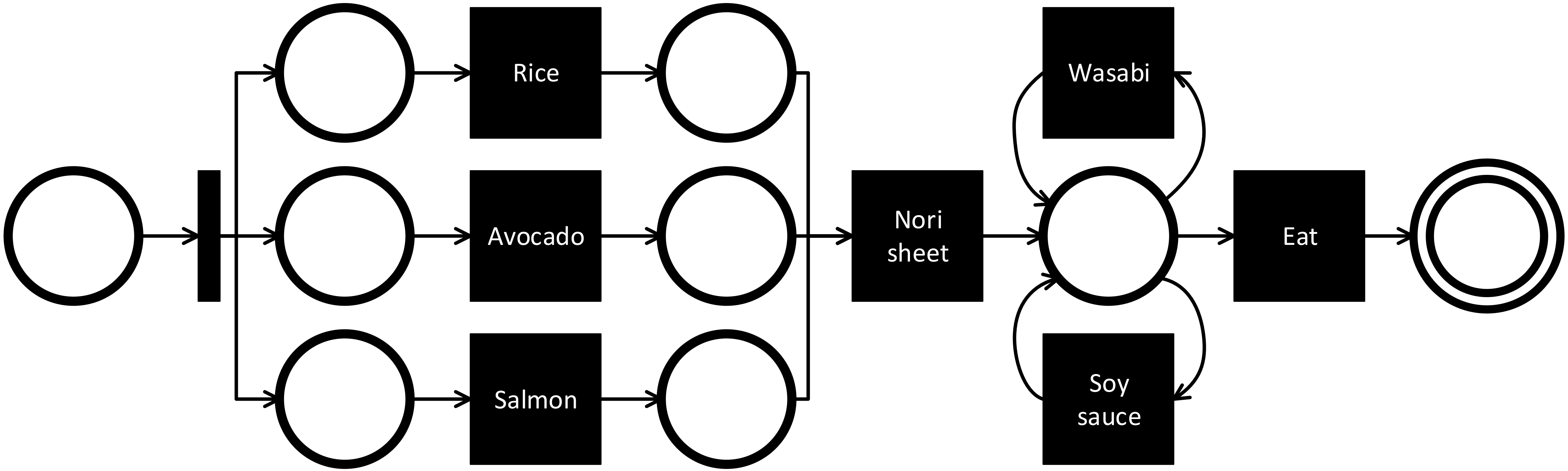
- Idea:
  - Consider all mappings between a model and a trace.
  - Simulate moves in the model and in the trace.
  - Optimize for most synchronuous moves
    (fire transition a and read a in the trace in parallel).
  - Finally, compare the optimal alignment with the worst alignment possible to determine the fitness.

moves in the log	a	b	c	d	>>	g	h
moves in the model	a	b	c	d	f	>>	h

>> is an asynchronous move

[2] W.M.P. van der Aalst, A. Adriansyah, and B. van Dongen. Replaying History on Process Models for Conformance Checking and Performance Analysis. *WIREs Data Mining and Knowledge Discovery*, **2**(2):182–192, 2012.

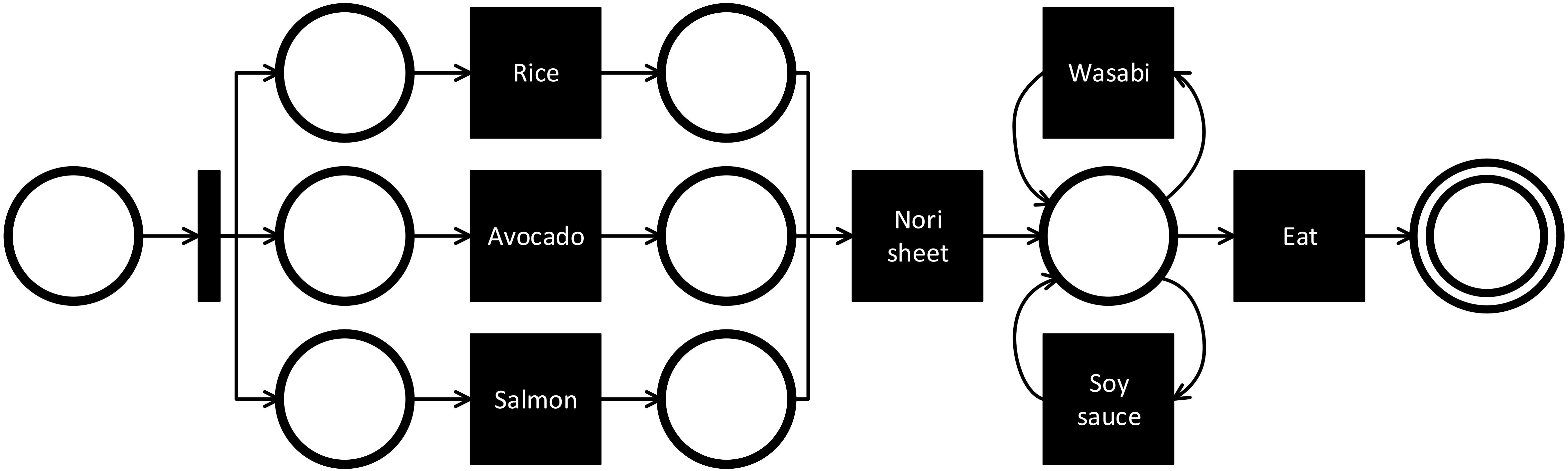
Alignments



- Worst possible alignment for <Rice, Salmon, Wasabi>:

Rice	Salmon	Eat	>>	>>	>>	>>	>>
>>	>>	>>	Rice	Avocad o	Salmon	Nori sheet	Eat

Alignments



- Optimal alignment for <Rice, Salmon, Salmon, Wasabi>:

Rice	>>	Salmon	Salmon	Wasabi	>>
Rice	Avocad o	Salmon	>>	Wasabi	Eat

Alignments

- Optimal alignment for <Rice, Salmon, Salmon, Wasabi>:

Rice	>>	Salmon	Salmon	Wasabi	>>
Rice	Avocado	Salmon	>>	Wasabi	Eat

- Optimal alignments do not require to be unique:

Rice	>>	Salmon	Salmon	Wasabi	>>
Rice	Avocado	>>	Salmon	Wasabi	Eat

Rice	Salmon	>>	Salmon	Wasabi	>>
Rice	Salmon	Avocado	>>	Wasabi	Eat

>>	Rice	Salmon	Salmon	Wasabi	>>
Avocado	Rice	>>	Salmon	Wasabi	Eat

>>	Rice	Salmon	Salmon	Wasabi	>>
Avocado	Rice	Salmon	>>	Wasabi	Eat

- However, the distance between log and model equal for all optimal alignments.

Alignments

- Optimal alignment for <Rice, Salmon, Salmon, Wasabi>:

$\lambda_{opt}^{M}(\sigma) =$

Rice	>>	Salmon	Salmon	Wasabi	>>
Rice	Avocado	Salmon	>>	Wasabi	Eat

$\delta\left(\lambda_{opt}^{M}(\sigma)\right) = 3$

- Worst alignment:

$\lambda_{worst}^{M}(\sigma) =$

Rice	Salmon	Eat	>>	>>	>>	>>	>>
>>	>>	>>	Rice	Avocado	Salmon	Nori sheet	Eat

$\delta\left(\lambda_{worst}^{M}(\sigma)\right) = 8$

- The fitness is defined as

$$fitness(\sigma, M) = 1 - \frac{\delta\left(\lambda_{opt}^{M}(\sigma)\right)}{\delta\left(\lambda_{worst}^{M}(\sigma)\right)} = 0.625$$

Alignments Discussion

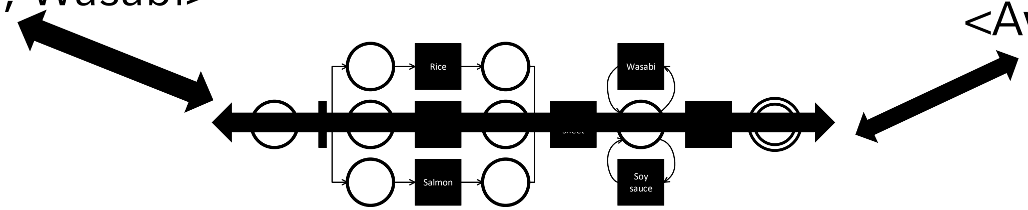
- Alignments easier to understand: Instead of tokens in Petri-nets, we talk about skipped and inserted events.
- Higher accuracy, since Token Replay suffers from token flooding.
- Fitness values for Alignments tends to be to low, while Token Replay often yields higher values.
- More flexibility due to modifications of the costs δ. E.g. activity "avocado" might be cheaper to drop than dropping the activity "rice".
- Not depending on Petri-nets only.
- However, very computational expensive.

Applications for Conformance Scores

- We only talked about conformance checking for fraud detection and workflow diagnostics.
- Fitness values determined by conformance checking provide us with a definition of distance between model and trace.
- The unstructured trace space, which is not a native vector space, becomes semi-metric.
  - The distance is not defined between traces, but uses models as reference points.
  - As the distance is not computed directly, but depends on a secondary structure, it is called geodetic.

<Rice, Salmon, Salmon, Wasabi>

<Avocado, Rice, Salmon, Nori, Eat>



  - Using this distance, clustering and outlier detection become possible:
    - Richter, F., Wahl, F., Sydorova, A., & Seidl, T. LWDA (2019). k-process: Model-Conformance-based Clustering of Process Instances.
    - Richter, F., Zellner, L., Sontheim, J., & Seidl, T. (2019, October). Model-Aware Clustering of Non-conforming Traces. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"* (pp. 193-200). Springer, Cham.
  - We can also lift this approach to a log-to-log level, defining distances between two process logs for clustering and outlier detection (k-means, DBSCAN,…):
    - Richter, F., Zellner, L., Azaiz, I., Winkel, D., & Seidl, T. (2019, September). LIProMa: Label-Independent Process Matching. In *International Conference on Business Process Management* (pp. 186-198). Springer, Cham.

Temporal Conformance Checking

- Until now: Does the order of events conform to a given model? Often it is interesting if events are also executed at the "right" time.
- Even for conform traces, an activity can be executed too early or too late.
- In the following, the execution order was correct and according to model, there is no problem:

$$\mathit{cook\ rice} \xrightarrow{6h13m12s} \mathit{prepare\ avocado} \xrightarrow{0h2m43s} \mathit{salmon} \xrightarrow{0h4m7s} \mathit{Combine\ and\ roll\ Nori\ sheet}$$

The last event failed due to dry and hard rice.

- Recent research on this at DBS:
  - Richter, Florian, and Thomas Seidl. "TESSERACT: time-drifts in event streams using series of evolving rolling averages of completion times." *International Conference on Business Process Management*. Springer, Cham, 2017.
  - Richter, Florian, and Thomas Seidl. "Looking into the TESSERACT: Time-drifts in event streams using series of evolving rolling averages of completion times." *Information Systems* 84 (2019): 265-282.
  - Sontheim, J., Richter, F., & Seidl, T. LWDA (2019). Temporal Deviations on Event Sequences.

Agenda

Perspectives - Motivational Example

Average daily outside temperature in °C

	Log 1	Log 2	RLE-based (Log 1)
Day 1	14.2	14.2	1*14.2
Day 2	14.4	14.4	4*14.4
Day 3	14.4	14.4	1*14.3
Day 4	14.4	-21.3	1*14.2
Day 5	14.4	14.4	
Day 6	14.3	14.3	
Day 7	14.2	14.2	

Detecting
anomalous behavior in
temperature data
by changing perspectives

Log 1:
E.g. Mean and standard deviation can be computed
$\Rightarrow$ still *seems* normal

Log 2:
Point anomaly is obvious

Log 3 (Saves entries of Log 1 in a **R**un-**L**ength **E**ncoding manner):
Exposes entries of Log 1 as a possible **collective anomaly**

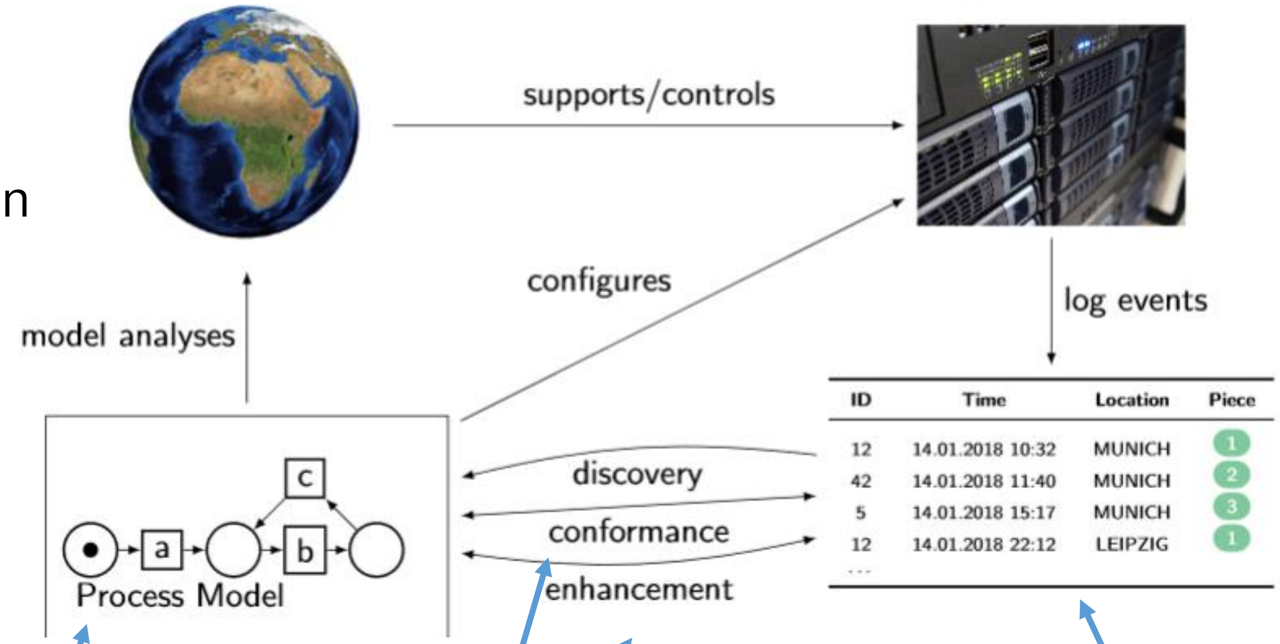
Motivation - Perspectives

- Analysis can be done by using different perspectives

=> Event logs provide much more information
E.g.: Timestamps, resources, transactions, costs etc.

- Thus far: **Control-flow perspective**

- Moreover:
  - **Time perspective**
  - **Case perspective**
  - **Organizational perspective**



Model can show multiple perspectives

Extension and conformance checks in every perspective possible

Events have attributes concerning various perspectives

Motivation - Perspectives

Time perspective

- Focus on timing and frequency of events
- Goals: Discover bottlenecks, monitor utilization of resources, remaining time prediction

Case perspective

- Focus on case properties
- Properties can be case attributes, event attributes, a path taken, performance information
- Goals: Mining decisions (e.g. a specific path) based on the characteristics of the case shows which data is relevant and should be included in the model

Organizational perspective

- Focus on information about resources
- Resources can be people, systems, roles, departments
- Goals: Classify actors in terms of roles, show social network

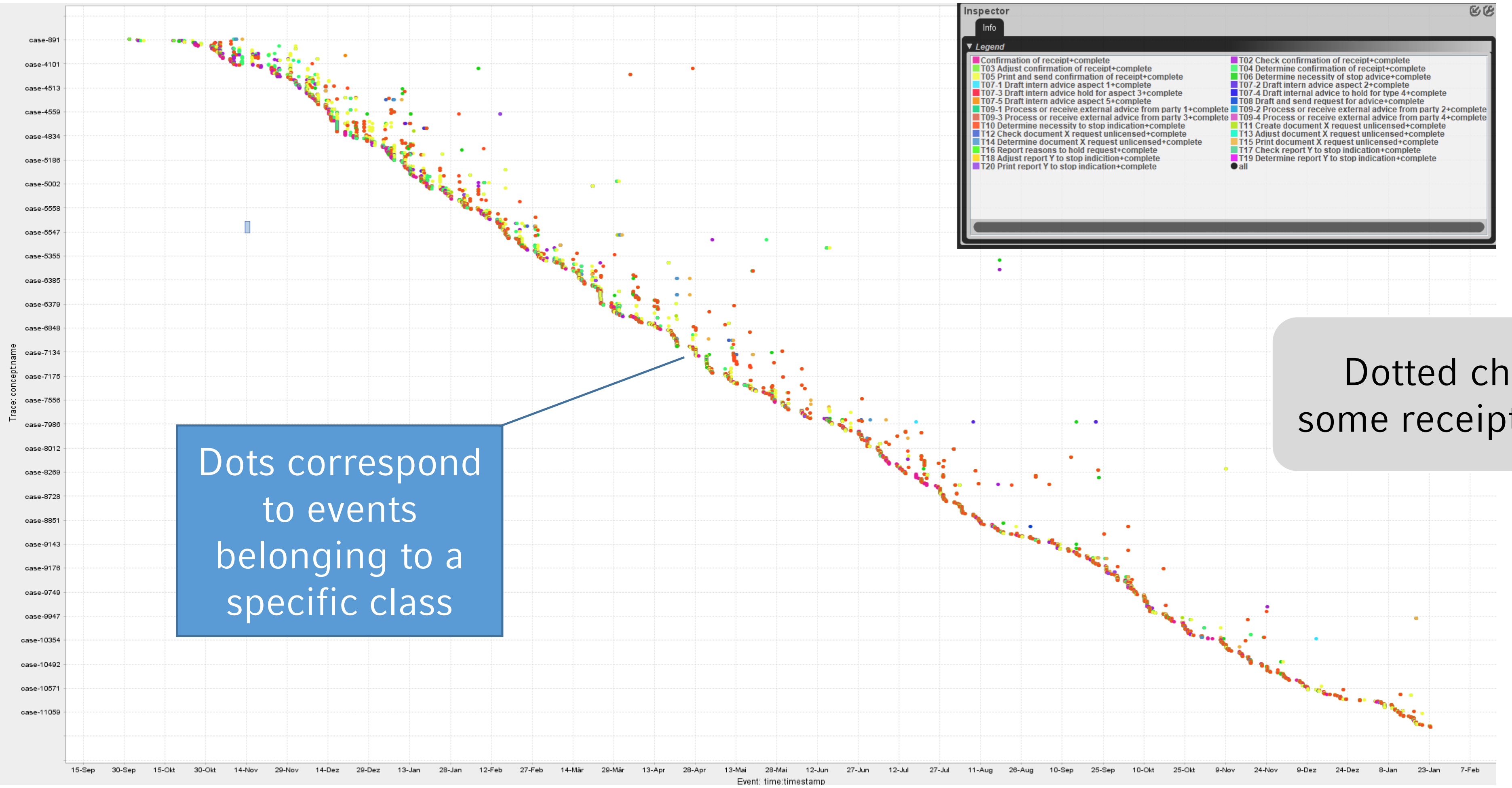
Exemplarily introducing temporal mining now

Temporal Visualization – Dotted Chart Analysis

- How to get a general overview: **Dotted Chart Analysis**

Legend mapping event colors to event descriptors

Classifier (here: *Case ID*) sorted by timestamp of first event

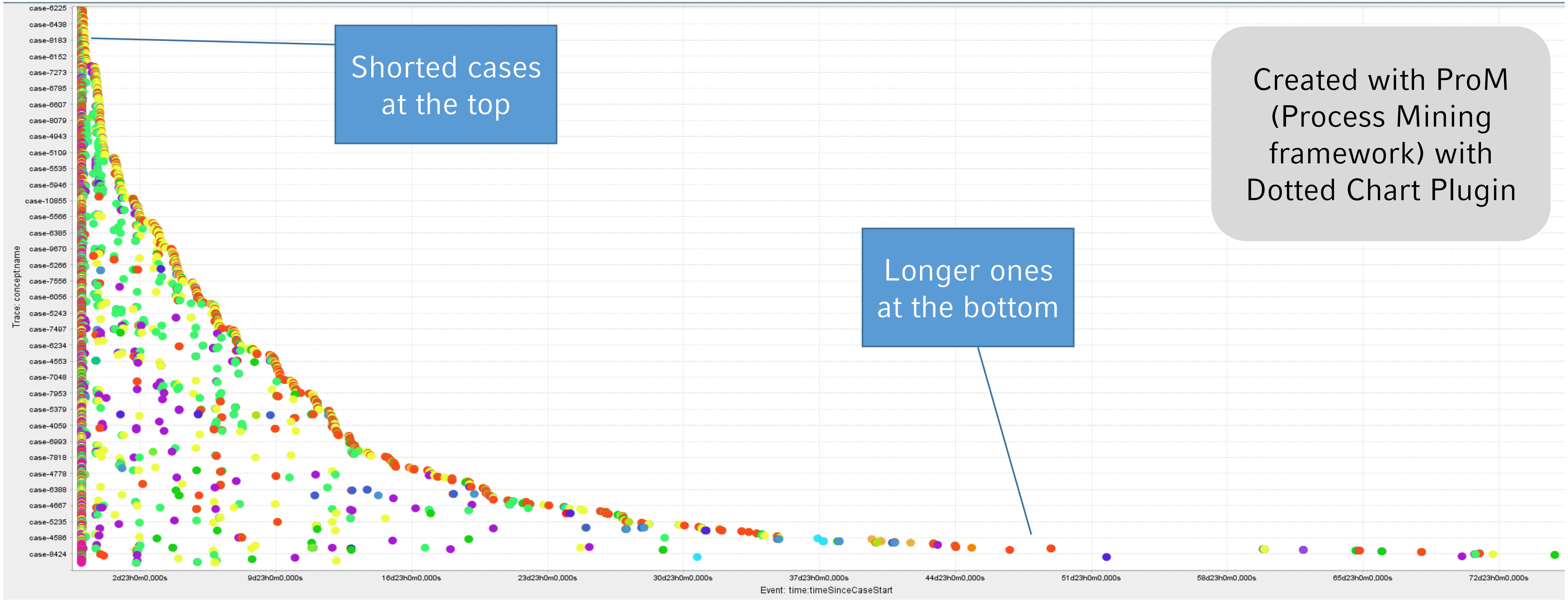


Dots correspond to events belonging to a specific class

Dotted chart for some receipt process

Time (absolute, relative or logical)

Temporal Visualization – Dotted Chart Analysis



Time since case started
sorted by duration of a case

Temporal Visualization – Dotted Chart Analysis

Time since week started.
Indicates that only few events were executed by night and at weekends.

→ Most events on weekdays between 9am and 4pm

Temporal Mining

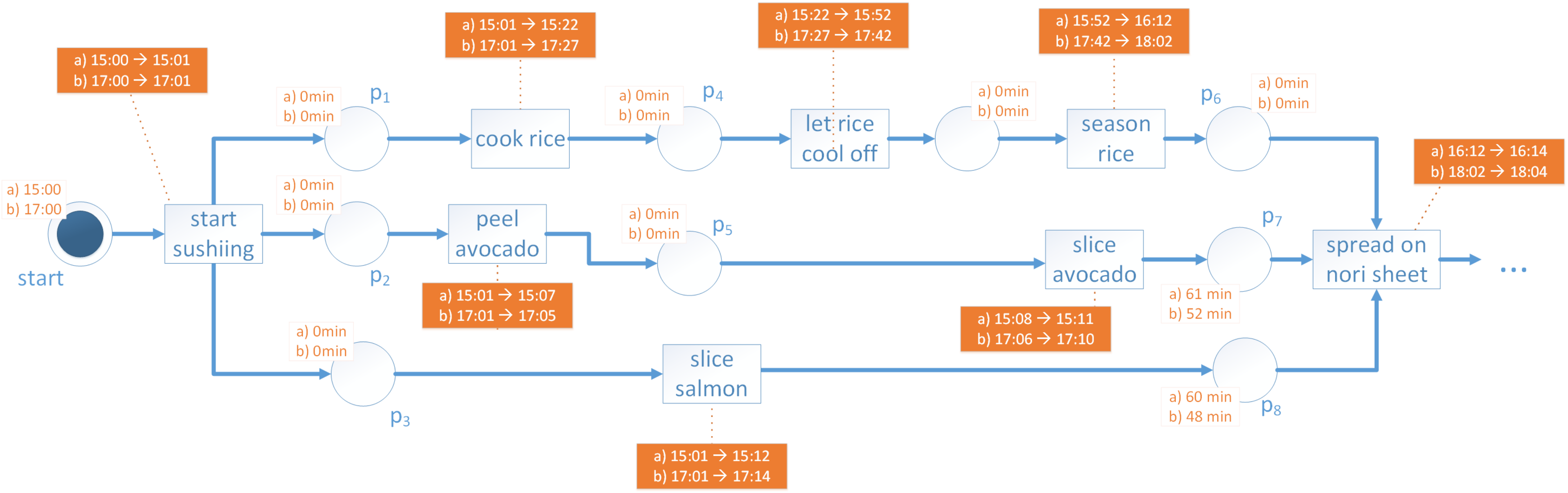
Presence of timestamps enables

- **discovery of bottlenecks**
  - Limitation of capacity of a specific resource
- **monitoring of resource utilization**
  - *Which resources are occupied by which activity the most?*
- **prediction of remaining processing times of running cases**
  - Based on computations made on discovered cases so far
- etc.

Token replay can be extended to replay event logs with timestamps included (*time-based replay*).

This can help to extract aforementioned information.

Temporal Mining – Time-based replay



Replay of first part of our sushi process for two cases starting at 3pm i.e. 5pm

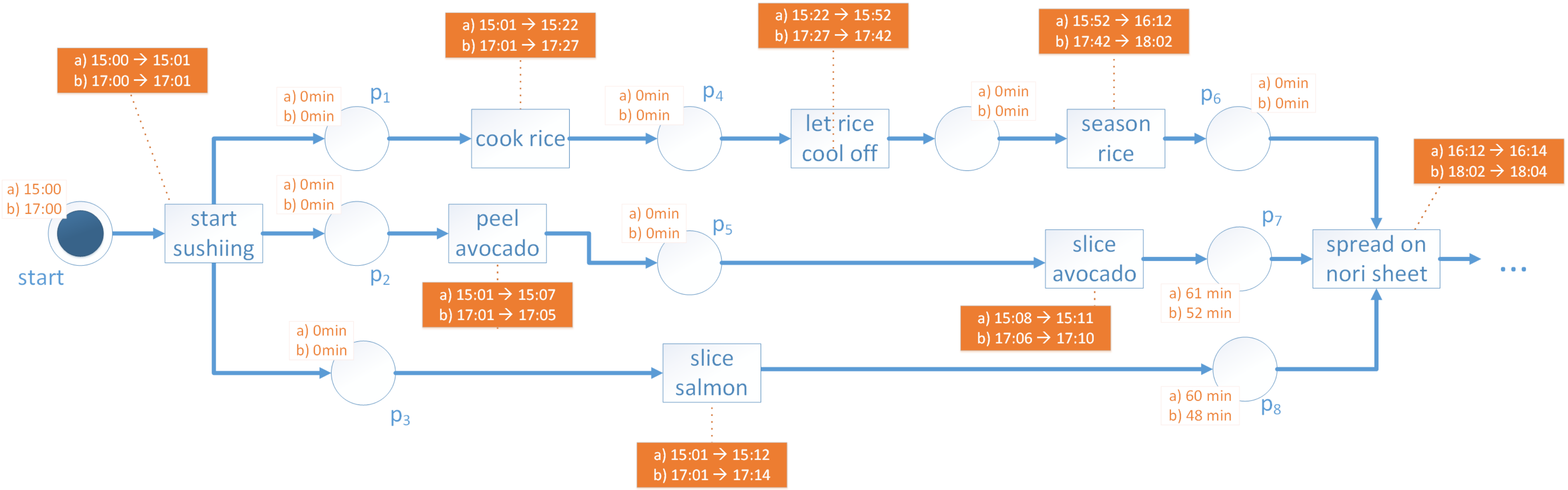
Timed replay for 2 cases showing durations at transitions and waiting times at places

Temporal Mining – Time-based replay

Replay of first part of our sushi process for two cases starting at 3pm i.e. 5pm

Record collection of token visits
→ derive multi set of durations for each place

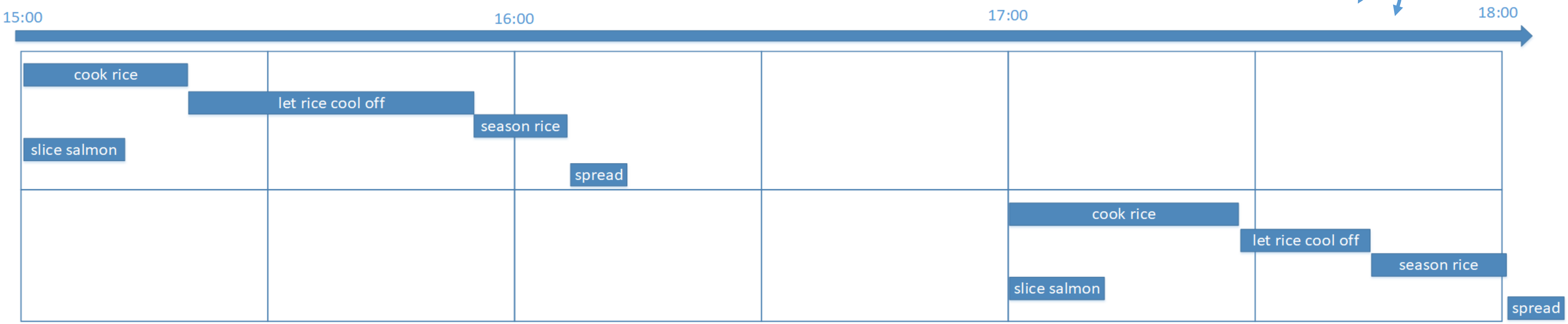
Partial sushiing process seems to have a bottleneck at $\langle cook\ rice,\ \ season\ rice\rangle$



Temporal Mining – Time-based replay

Possibility to
- Fit distribution
- Compute statistics such as
  - mean,
  - standard deviation,
  - minimum,
  - maximum
  - etc.

⇒ Visualization of waiting times
⇒ Visualization of service times
⇒ Bottleneck detection and analysis



Timeline resembling a Gantt-Chart (excerpt of time-based replay)

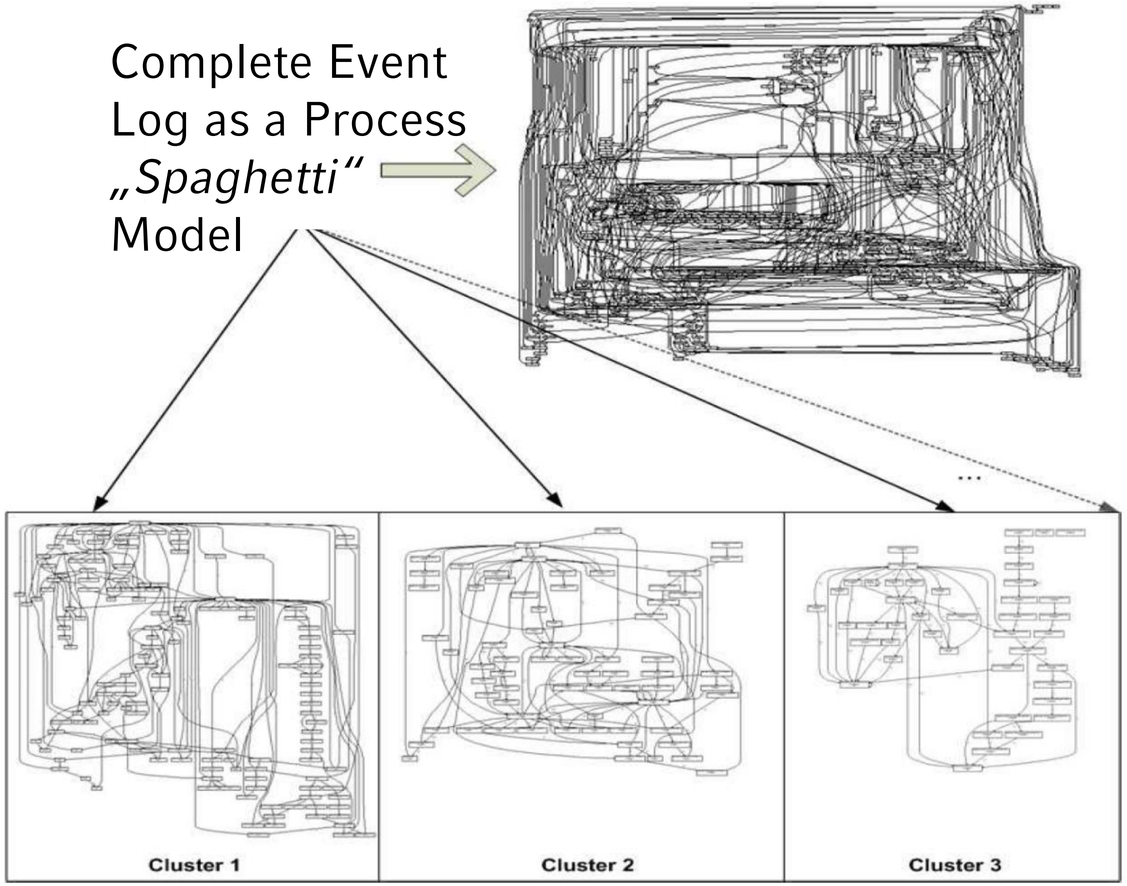
Trace Clustering - Motivation

High *diversity:*

Single cases differ significantly from one another

→ possibly very complex models

Complete Event Log as a Process *„Spaghetti"* Model



Our sushiing process already can be very complex depending on the granularity of visualization

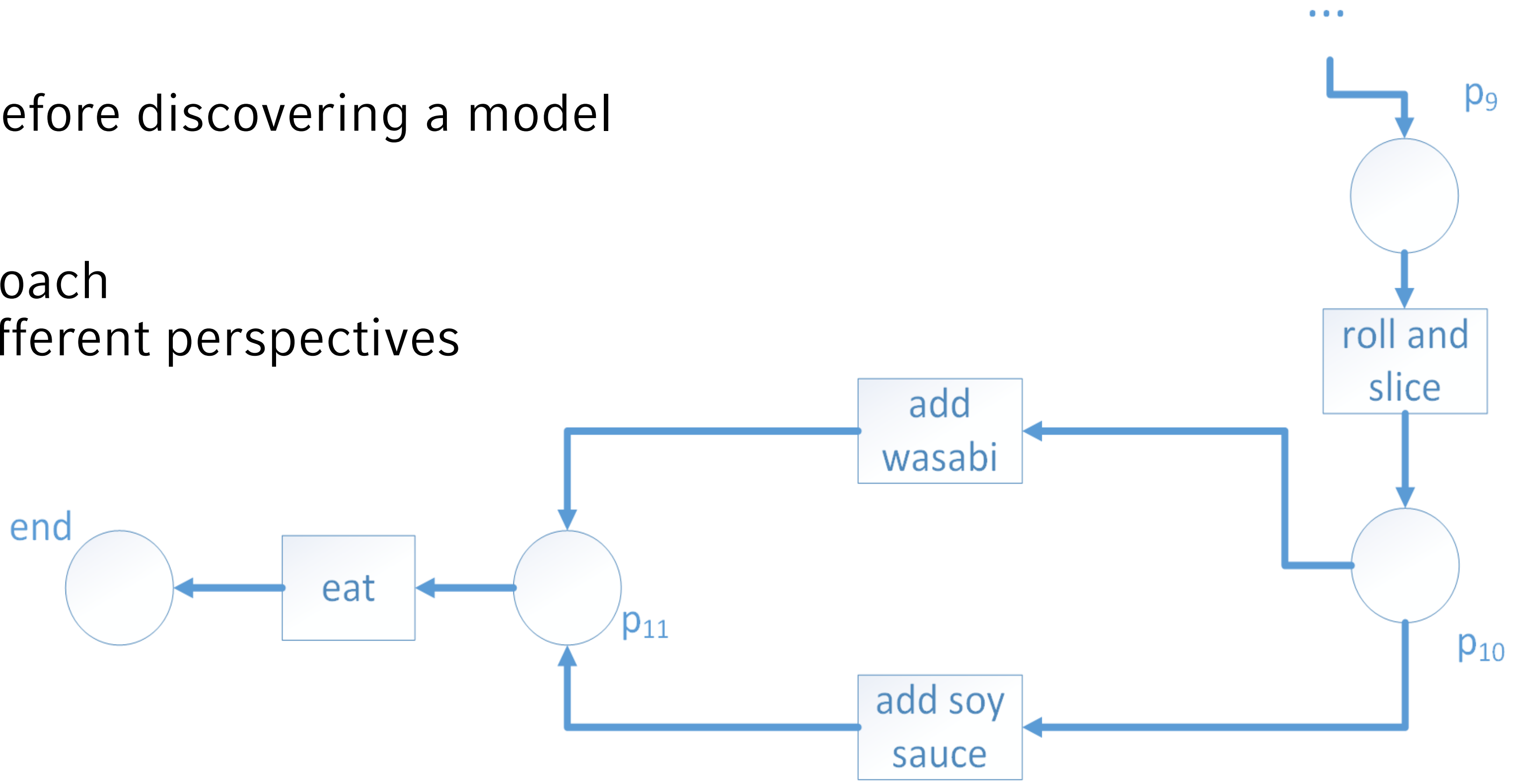
Trace Clustering - Motivation

Assumption:

Process variants hidden within the event log

⇒ Cluster traces before discovering a model

⇒ Clustering approach
also based on different perspectives



Example: Second part of our sushiing process

Trace Clustering - Example

- How to determine a similarity value between our data points (here: *cases*)?
- Clustering on points in vector space is well-known

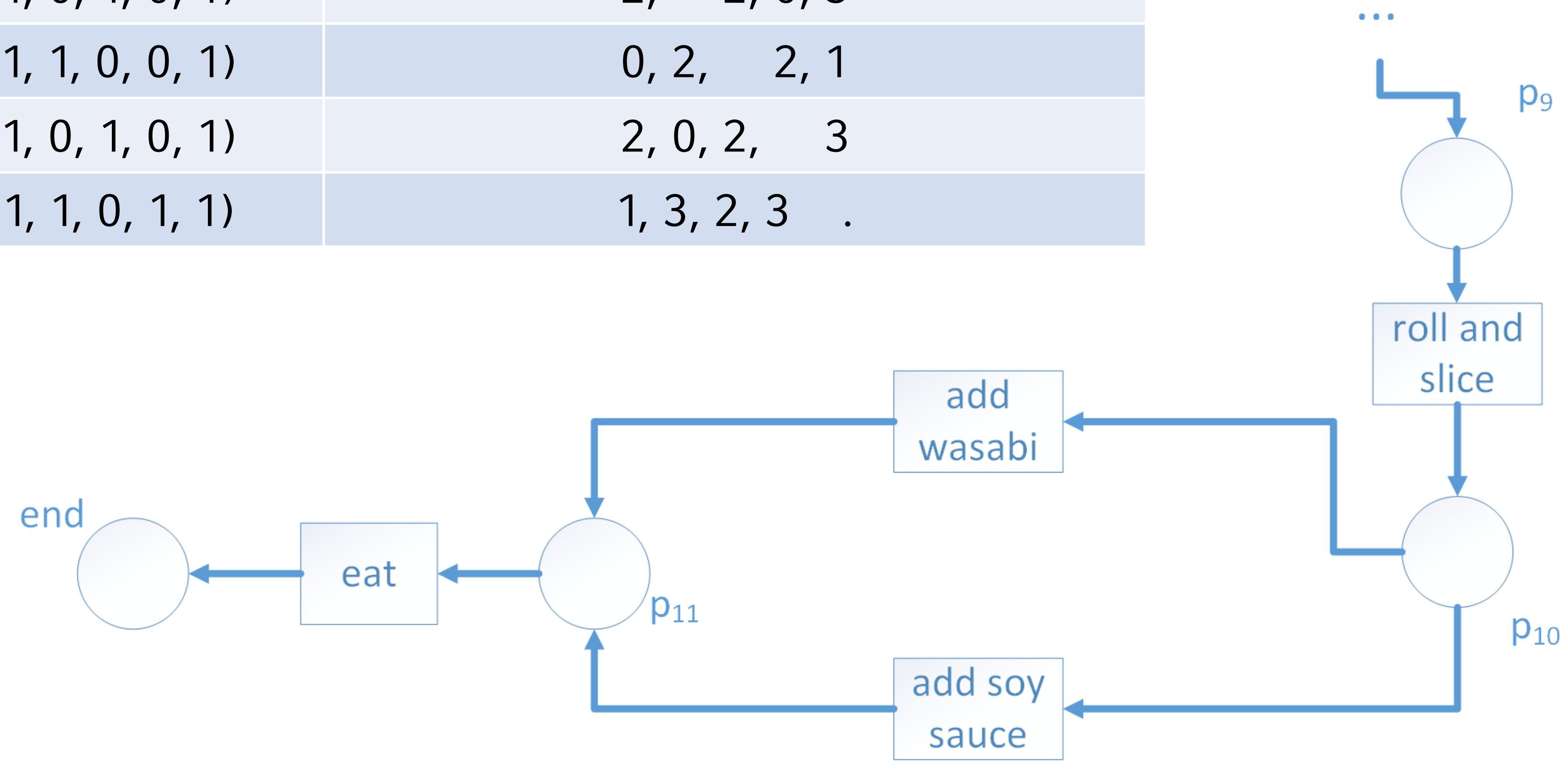
=> Embedding of *cases* into vector space necessary → **Profiles**

Case ID	Roll and slice	Add wasabi	Add soy sauce	Prepare stir-fried rice	Eat
1	1	1	0	0	1
2	1	0	1	0	1
3	1	1	0	0	1
4	1	0	1	0	1
5	1	1	0	1	1

Add up the number of activity execution for each case

Trace Clustering - Example

Case ID	Vector	Manhattan distance to other vectors
1	(1, 1, 0, 0, 1)	2, 0, 2, 1
2	(1, 0, 1, 0, 1)	2, 2, 0, 3
3	(1, 1, 0, 0, 1)	0, 2, 2, 1
4	(1, 0, 1, 0, 1)	2, 0, 2, 3
5	(1, 1, 0, 1, 1)	1, 3, 2, 3 .



⇒ **E.g. cluster with agglomerative approach**

Trace Clustering – Methods

Aforementioned profile is called *Activity Profile (Activity Histogram)*

- Defines one item (feature) per type of activity
- An activity item is measured by counting all events of a trace which have that activities name
- Of course, **various other profiles possible** as well

In General:

Profile: Set of items with measurements
Item: Assigns numeric value to each trace

⇒ A Profile can be considered a function f which maps a trace *t* to a vector $(i_1, i_2, \dots, i_n)$ with *n* items:

$$f(t) \rightarrow (i_1, i_2, \dots, i_n),$$

→ **✓ Embedding into vector space**

⇒ Various clustering methods can be applied now

Trace Clustering – Methods

More examples:

<u>Transition profile:</u>

Items: Direct following relations in a trace

Measure: How often an event A has been followed by an event B

Goal: Measure behavior of traces (capturing the context) cf. **n-grams**

<u>Performance profile:</u>

Items: Size of a trace
regarding timestamps: case duration, (min, max, mean) time difference between events etc.

Measure: Depends on predefined items e.g. size is measured by number of events

Goal: Measure performance of a trace (→ also part of Temporal Mining)

Additional Mining Tasks - Roundup

- Processes can be analyzed by using different **perspectives**
  - time
  - case
  - organizational

  perspective

- Get an overview by applying **Dotted Chart Analysis**

- **Temporal Mining** useful to
  - detect bottlenecks
  - monitor resource utilization
  - predict remaining processing time

- **Trace Clustering** helps to distinguish between process variants dependent on different perspectives (*profiles*)

Resources

- ProM Framework

- Wil van der Aalst. 2016. *Process Mining: Data Science in Action* (2nd. ed.). Springer Publishing Company, Incorporated

- Song, Minseok & Günther, Christian & Aalst, Wil. (2008). Trace Clustering in Process Mining. Lecture Notes in Business Information Processing. 17. 109-120. 10.1007/978-3-642-00328-8_11.

- R.P., Jagadeesh Chandra Bose & Aalst, Wil. (2009). Context Aware Trace Clustering: Towards Improving Process Mining Results. SDM. 10.1137/1.9781611972795.35.