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

Knowledge Discovery and Data Mining 1

(Data Mining Algorithms 1)

Chapter 5 – Process Mining

Wintersemester 2022/23

Agenda

1. Introduction
2. Basics
3. Supervised Methods
4. Unsupervised Methods
5. Process Mining
   - Introduction
   - Process Models
   - Log Files
   - Process Discovery
   - Conformance Checking

Processes occur everywhere

Laws	Business	Life sciences	Medicine	IT / OS	Manufacturing	Process engineering
						
legal procedures	negotiations	metabolism	anamnesis	programs in execution	machining	chemical processes
criminal trials	orders	growth	diagnosis	system processes	assembly	thermal treatments
civil litigations	logistics	regulation	intervention	user processes	packaging	material technology
		immune response	evaluation		(event-based)	(mainly continuous)

Processes – notion and meaning

Statistics point of view

- All **data are generated** by processes
  - Statistical learning – learn the distribution (parametric or non-parametric density functions)
  - Deterministic processes – outcomes are determined in advance
  - Stochastic processes – outcomes carry uncertainties

Progress point of view

- Processes represent **changes over time**
- Etymology --- Latin "processus" (progress), from "procedere" (to procede)
- Ancient greek Heraclitos: "panta rhei" (πάντα ῥεῖ), "everything is in flow"
- Abstract time (sequences) or metric time (wall clock, calendar)

Wikipedia

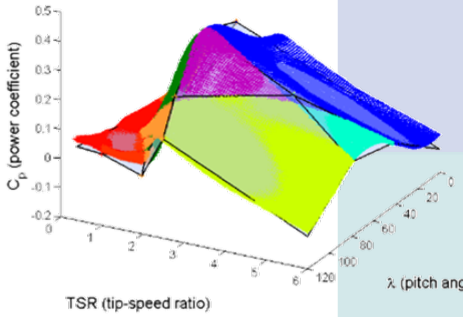
- „A **process** is a set of activities that interact to achieve a result.“
  - From https://en.wikipedia.org/wiki/Process_(disambiguation)
  - Some more meanings in https://de.wikipedia.org/wiki/Prozess_(Begriffsklärung)

Continuous processes vs. event-based processes

Continuous processes

Flow Models

- Partial Differential Equations
- (non-) linear functions
  e.g., hinging hyperplanes
- **Control theory**



[Ivanescu, Kranen, Seidl: SSDBM 2012]

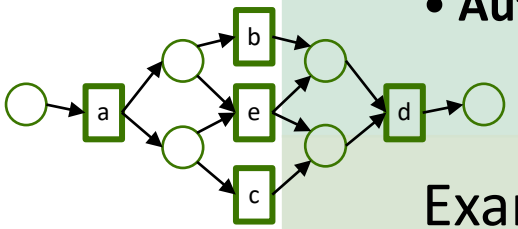
Examples

- Technical combustion
- Chemical processes

Event-based processes

Discrete Models

- Cases / traces of events
- Transition systems, Petrinets
- **Automata theory**



Examples

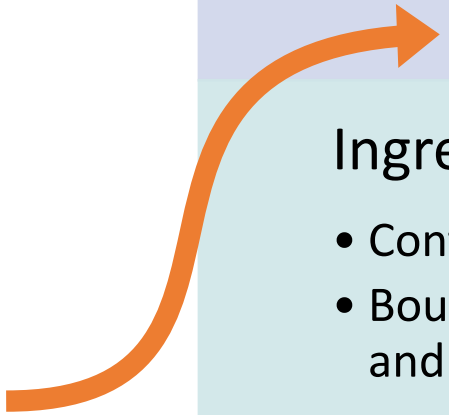
- Assembly in manufacturing
- Customer journeys
- Logistics

Example elevator control – both perspectives play a role

Smooth start and stop

Control theory

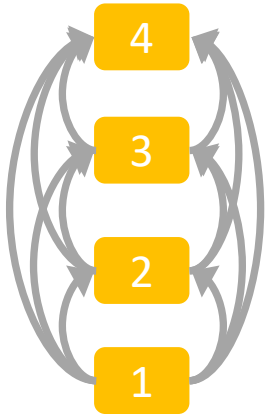
- Jerk-free acceleration
- Partial differential equations
- Mechanical Eng. view



Ingredients

- Continuous 2nd derivatives
- Boundary conditions for start and stop positions

Ride selection



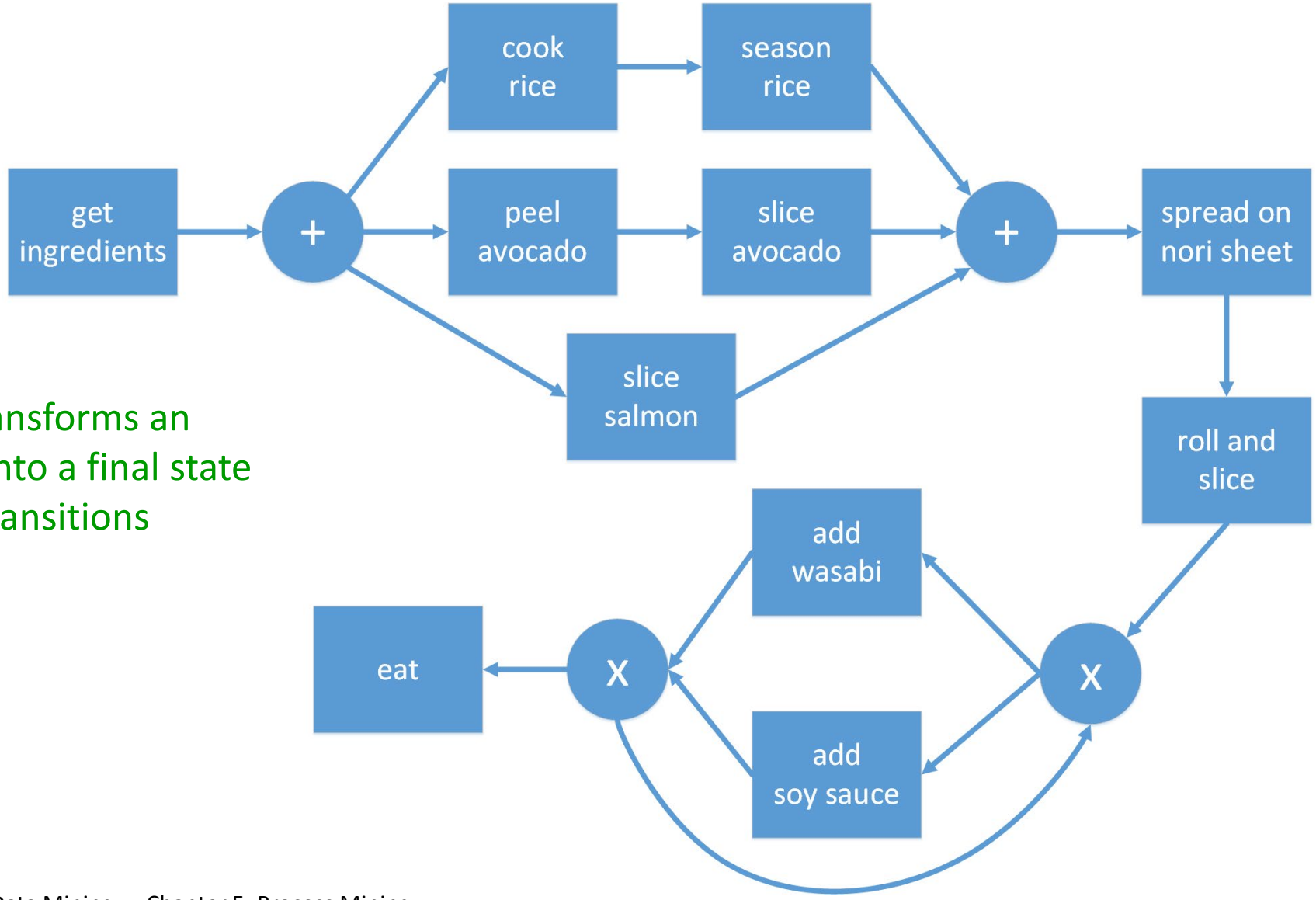
Automata theory

- Users select target floors
- State transition system
- Computer Science view

Ingredients

- States represent floors
- Alphabet for floor selection
- Transitions represent rides

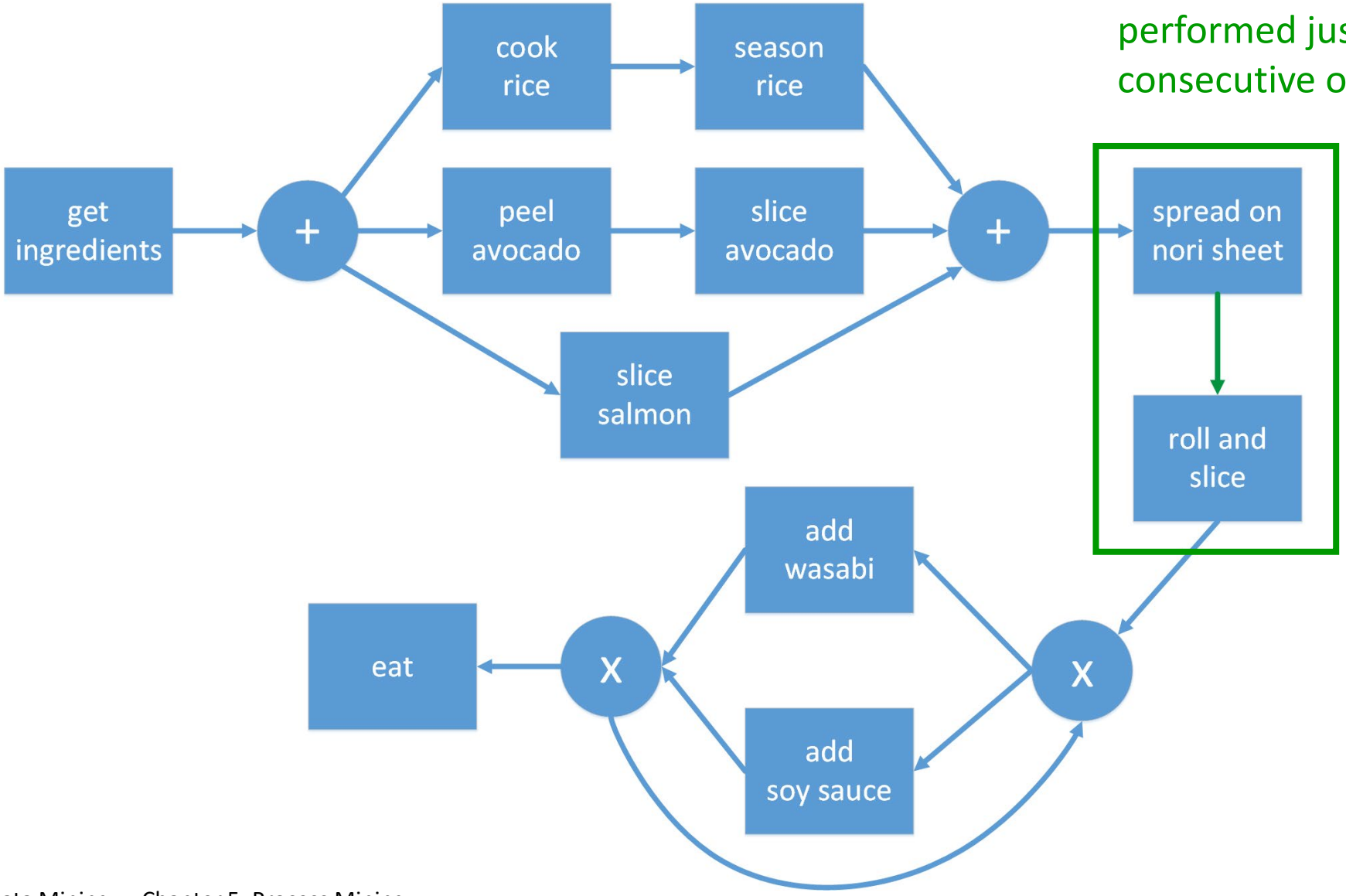
Example: Sushi Process



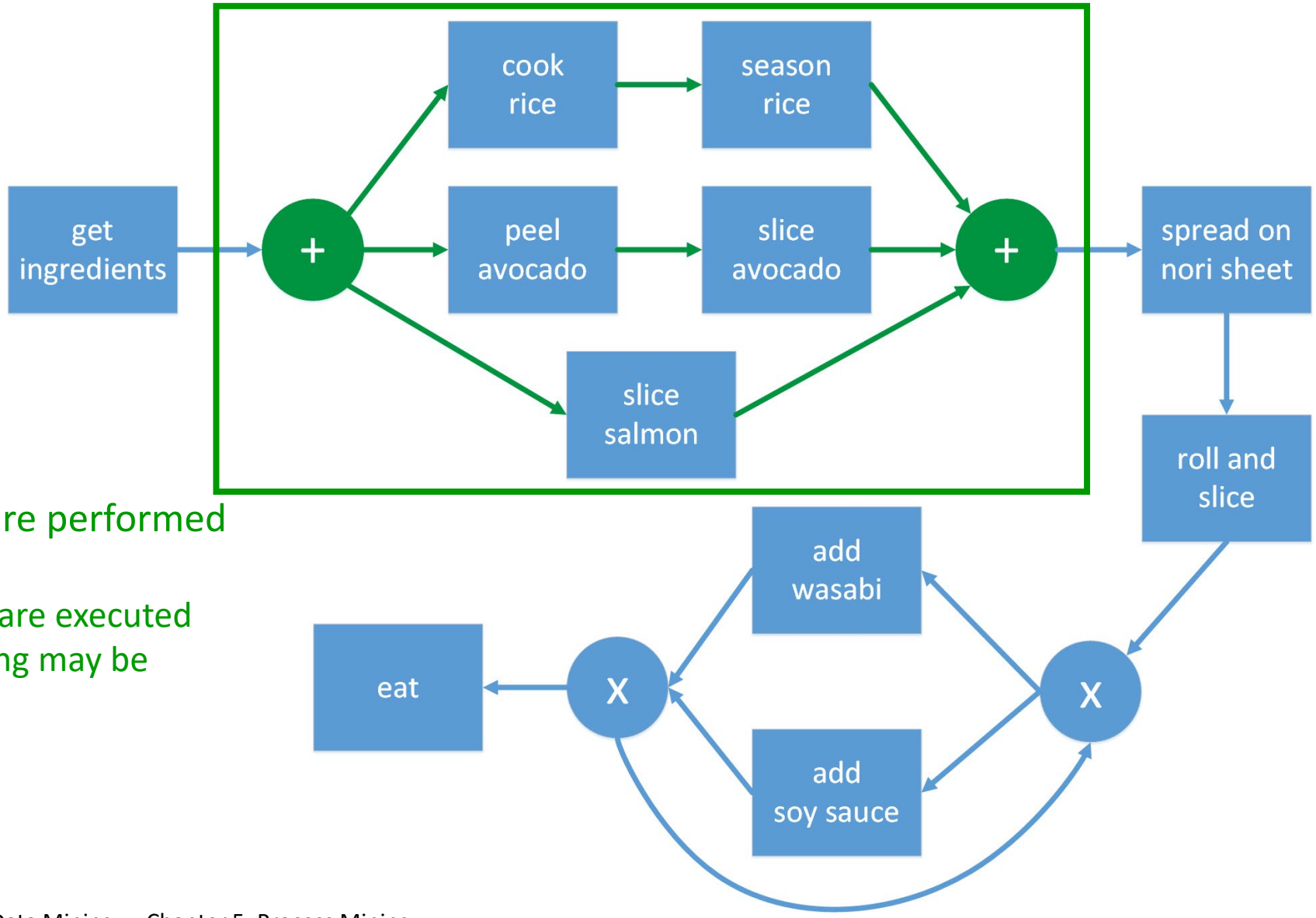
A process transforms an initial state into a final state via several transitions

Process Properties: Sequence

Most actions are performed just in consecutive order



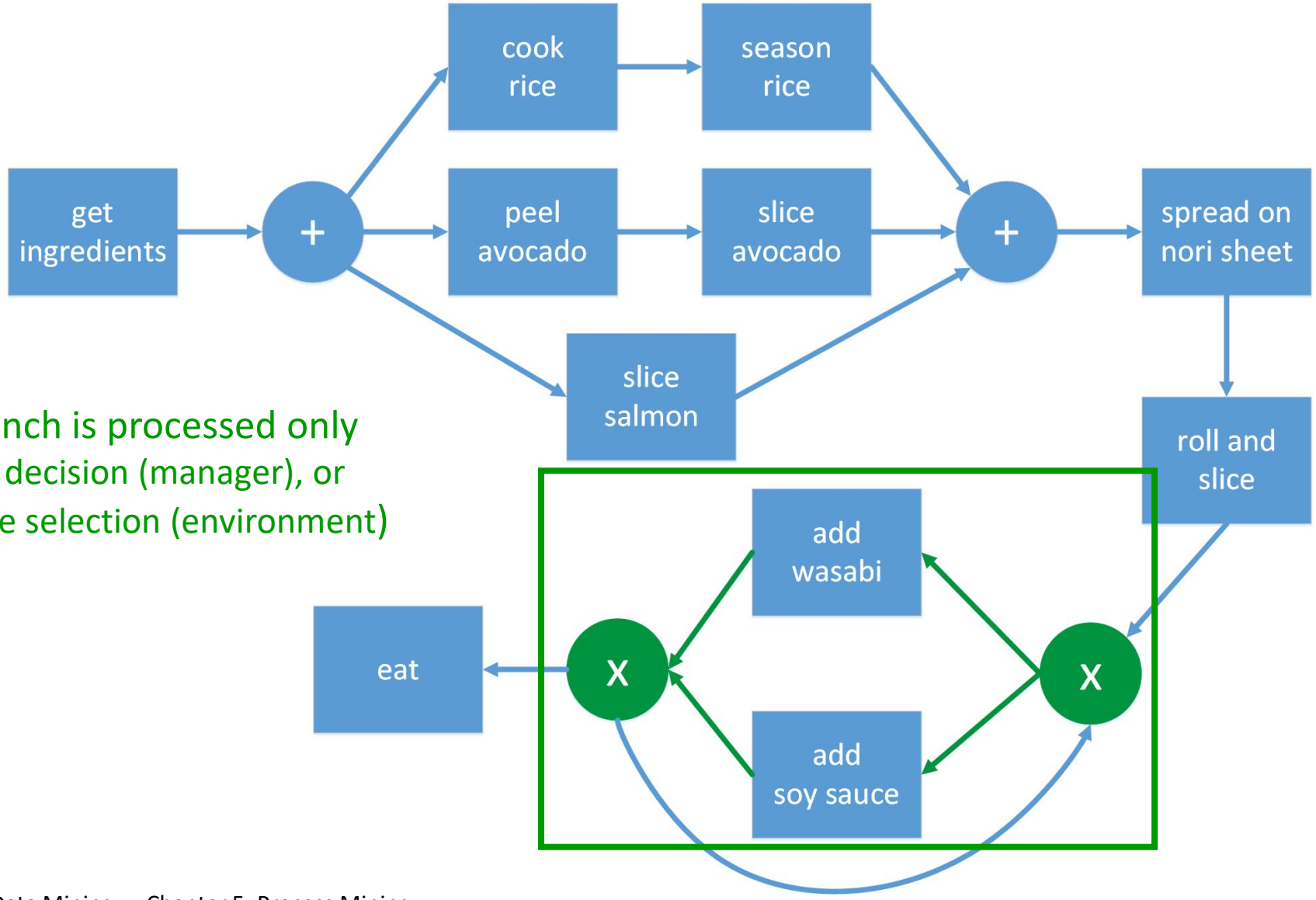
Process Properties: Concurrency



Some actions are performed in parallel

- All branches are executed
- The processing may be interleaved

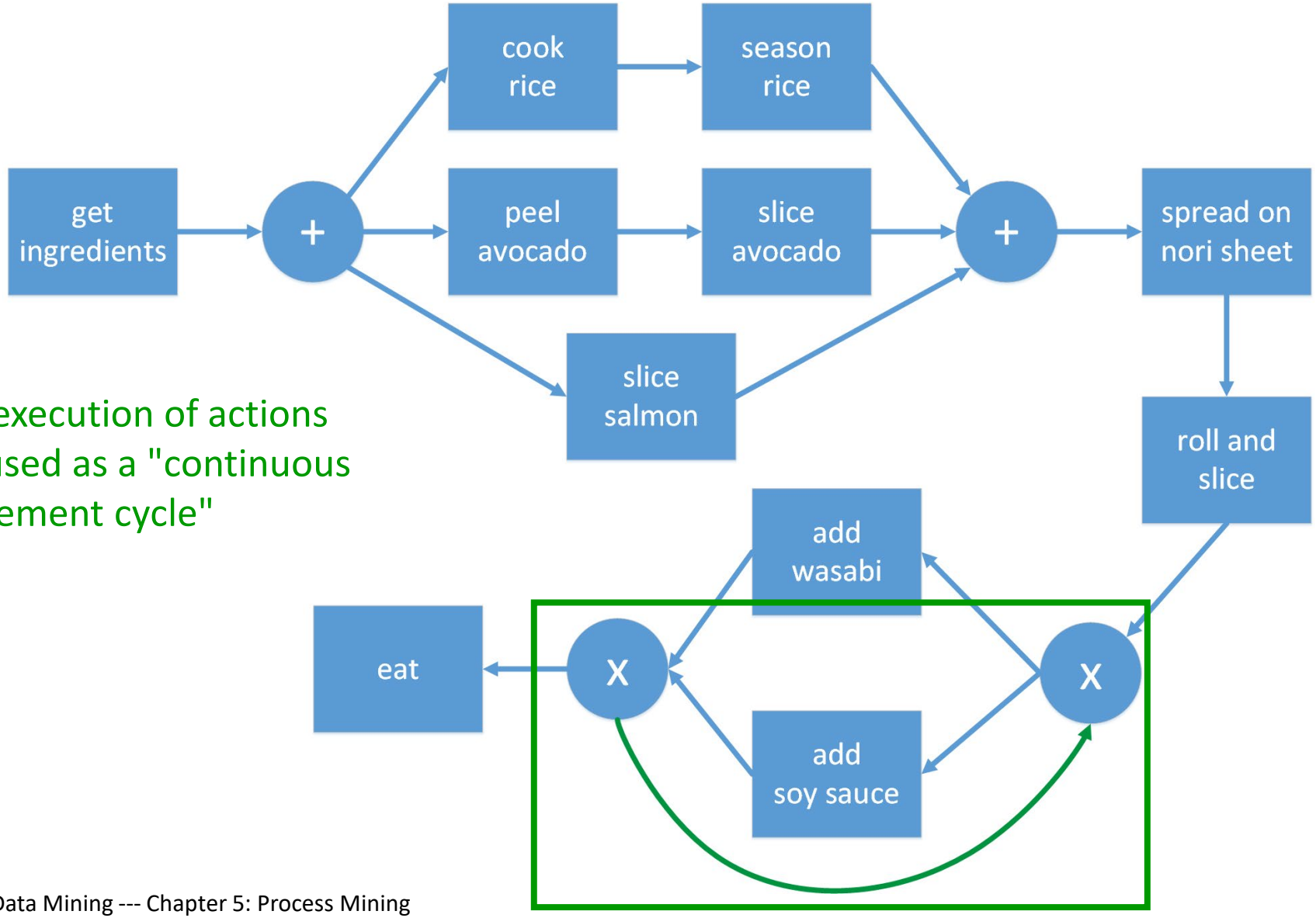
Process Properties: Choice



A single branch is processed only

- by active decision (manager), or
- by passive selection (environment)

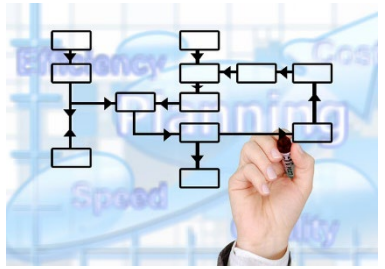
Process Properties: Loop



Repeated execution of actions

- Often used as a "continuous improvement cycle"

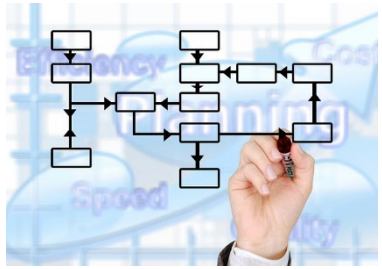
Log files keep track of events



time	case	activity
2018-6-6-6:29	732	a
2018-6-6-6:32	744	a
2018-6-6-6:33	732	b
2018-6-6-6:34	728	a
2018-6-6-6:35	732	d
2018-6-6-6:37	744	b
2018-6-6-6:38	728	c
2018-6-6-6:39	751	a
2018-6-6-6:42	744	d
2018-6-6-6:43	732	d
2018-6-6-6:44	744	e
2018-6-6-6:45	751	c
2018-6-6-6:47	732	e
2018-6-6-6:48	744	g
2018-6-6-6:59	751	d
2018-6-6-7:02	751	e
2018-6-6-7:03	728	e
2018-6-6-7:04	768	a
2018-6-6-7:05	751	h
2018-6-6-7:07	768	c
2018-6-6-7:08	728	h
2018-6-6-7:09	732	g
2018-6-6-7:12	768	d
2018-6-6-7:13	779	a
2018-6-6-7:14	768	e
2018-6-6-7:15	779	b
2018-6-6-7:17	768	h
2018-6-6-7:18	779	d

- Event log file = protocol of events which happened
- Core information per event
  - **Activity** – What happened in that step?
  - **Timestamp** – When did it happen?
  - **Case** – Which case is affected?
    - Processes are executed for several instances
    - Examples: patients, customers, applications, mechanical parts
- Additional information per event
  - **Context** – e.g., machine, tools, location, operator
  - **Resources** consumption – e.g., energy, work force, duration

Traces represent process executions for individual cases

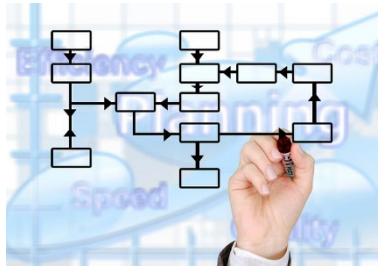


time	case	activity
2018-6-6-6:29	732	a
2018-6-6-6:32	744	a
2018-6-6-6:33	732	b
2018-6-6-6:34	728	a
2018-6-6-6:35	732	d
2018-6-6-6:37	744	b
2018-6-6-6:38	728	c
2018-6-6-6:39	751	a
2018-6-6-6:42	744	d
2018-6-6-6:43	732	d
2018-6-6-6:44	744	e
2018-6-6-6:45	751	c
2018-6-6-6:47	732	e
2018-6-6-6:48	744	g
2018-6-6-6:59	751	d
2018-6-6-7:02	751	e
2018-6-6-7:03	728	e
2018-6-6-7:04	768	a
2018-6-6-7:05	751	h
2018-6-6-7:07	768	c
2018-6-6-7:08	728	h
2018-6-6-7:09	732	g
2018-6-6-7:12	768	d
2018-6-6-7:13	779	a
2018-6-6-7:14	768	e
2018-6-6-7:15	779	b
2018-6-6-7:17	768	h
2018-6-6-7:18	779	d

#	trace
455	acdeh
191	abdeg
177	adceh
144	abdeh
111	acdeg
82	adceg
56	adbeh
47	acdefdbeh
38	adbeg
33	acdefbdeh
14	acdefbdeg
11	acdefdbeg
9	adcefcdeh
8	adcefdbeh
5	adcefbdeg
3	acdefbdefdbeg
2	adcefdbeg
2	adcefbdefbdeg
1	adcefdbefbdeh
1	adbefbdefdbeg
1	adcefdbefcdefdbeg
1391	

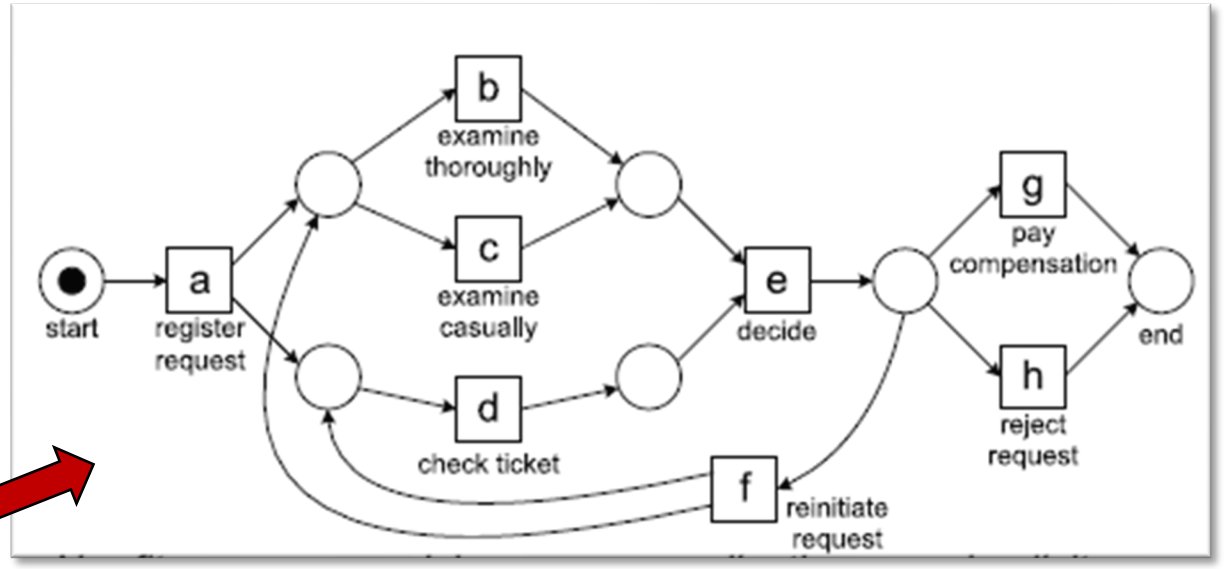
- Extraction of traces
  - Group log file entries by cases
  - I.e., for each case, extract the respective events to form its trace
- Examples
  - Some traces: a-b-d-d-e-g (732), a-c-e-h (728), a-c-d-e-h (768)
  - The number column indicates the frequency of traces (from a bigger example)
- Comparison to market basket analysis
  - Transactions have been grouped by customers already
  - Order of items in transactions is not available

Process discovery – „play in“ of process models



time	case	activity
2018-6-6-6:29	732	a
2018-6-6-6:32	744	a
2018-6-6-6:33	732	b
2018-6-6-6:34	728	a
2018-6-6-6:35	732	d
2018-6-6-6:37	744	b
2018-6-6-6:38	728	c
2018-6-6-6:39	751	a
2018-6-6-6:42	744	d
2018-6-6-6:43	732	d
2018-6-6-6:44	744	e
2018-6-6-6:45	751	c
2018-6-6-6:47	732	e
2018-6-6-6:48	744	g
2018-6-6-6:59	751	d
2018-6-6-7:02	751	e
2018-6-6-7:03	728	e
2018-6-6-7:04	768	a
2018-6-6-7:05	751	h
2018-6-6-7:07	768	c
2018-6-6-7:08	728	h
2018-6-6-7:09	732	g
2018-6-6-7:12	768	d
2018-6-6-7:13	779	a
2018-6-6-7:14	768	e
2018-6-6-7:15	779	b
2018-6-6-7:17	768	h
2018-6-6-7:18	779	d

#	trace
455	acdeh
191	abdeg
177	adceh
144	abdeh
111	acdeg
82	adceg
56	adbeh
47	acdefdbeh
38	adbeg
33	acdefbdeh
14	acdefbdeg
11	acdefdbeg
9	adcefcdeh
8	adcefdbeh
5	adcefbdeg
3	acdefbdefdbeg
2	adcefdbeg
2	adcefbdefbdeg
1	adcefdbefbdeh
1	adbefbdefdbeg
1	adcefdbefcdefdbeg
1391	



Process models: many are transition systems

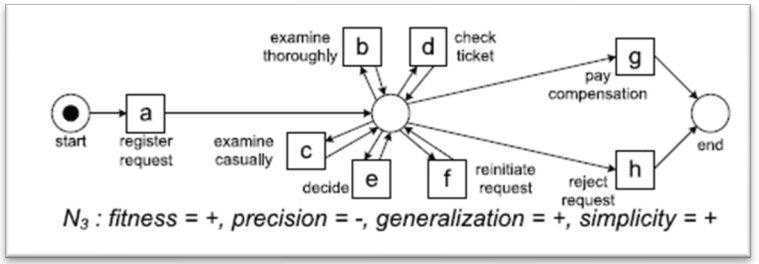
- Petrinets, BPM-nets, Workflow-nets
- Reachability graphs, causal networks

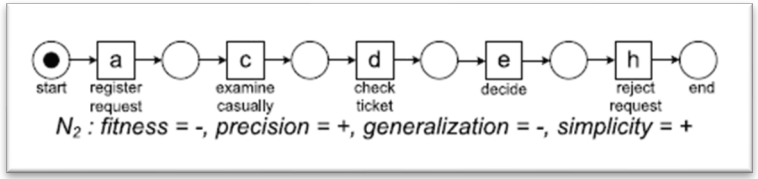
Discovery task: Extract process model from log entries which

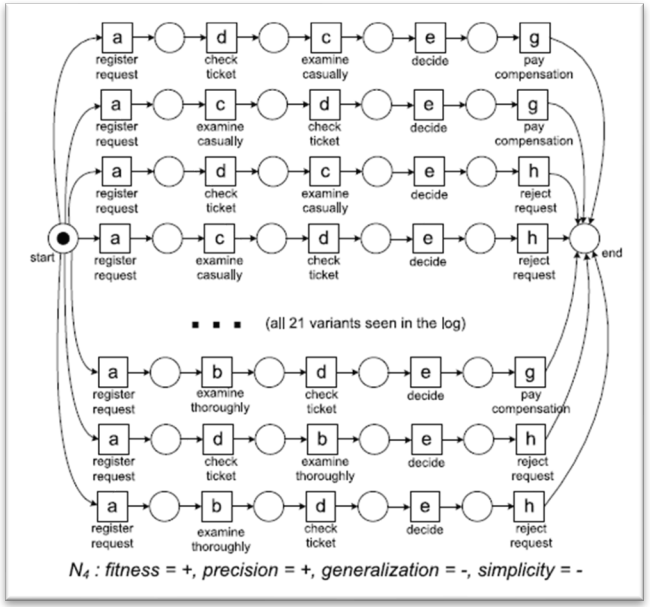
- … is able to replay the log ⇒ $Fitness$
- … simplifies as far as possible ⇒ $Simplicity$
- … does not overfit the log ⇒ $Generalization$
- … does not underfit the log ⇒ $Precision$

Process discovery – tune generalization of model

#	trace
455	acdeh
191	abdeg
177	adceh
144	abdeh
111	acdeg
82	adceg
56	adbeh
47	acdefdbeh
38	adbeg
33	acdefbdeh
14	acdefbdeg
11	acdefdbeg
9	adcefcdeh
8	adcefdbeh
5	adcefbdeg
3	acdefbdefdbeg
2	adcefdbeg
2	adcefbdefbdeg
1	adcefdbefbdeh
1	adbefbdefdbeg
1	adcefdbefcdefdbeg
1391	







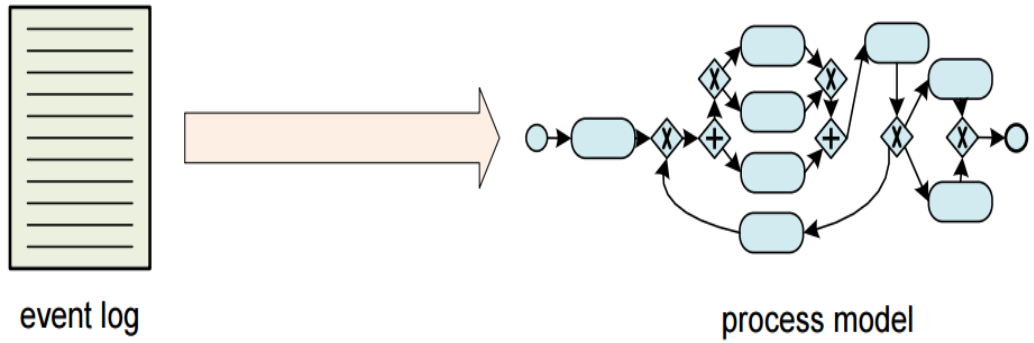
Over-generalization

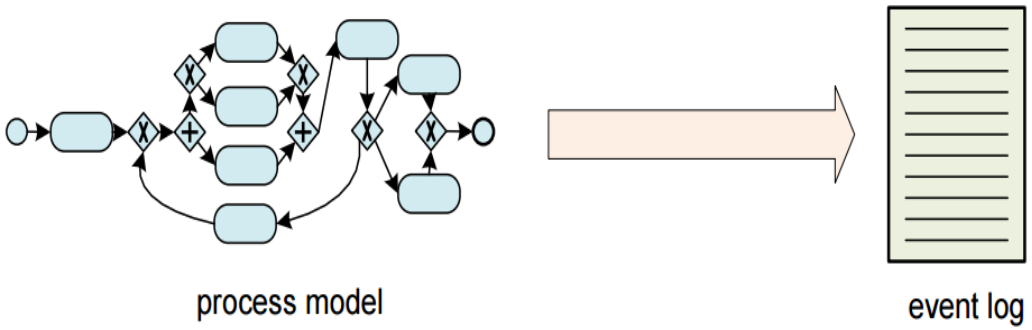
Desired fit

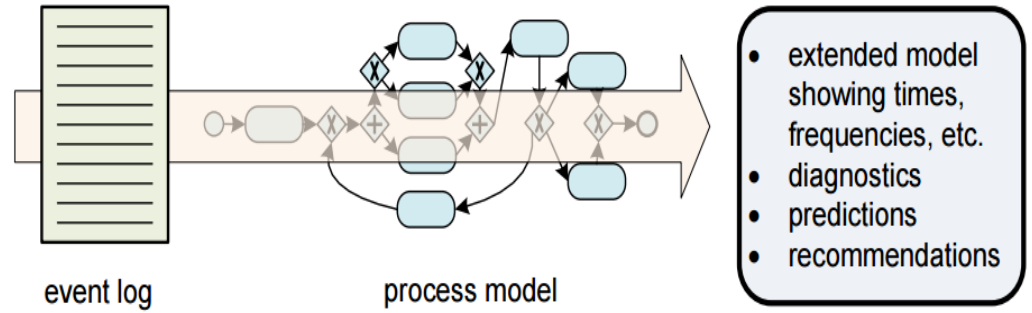
Overfitting

Process mining: different tasks

- Process Discovery
- Conformance Check
- Process Enhancement

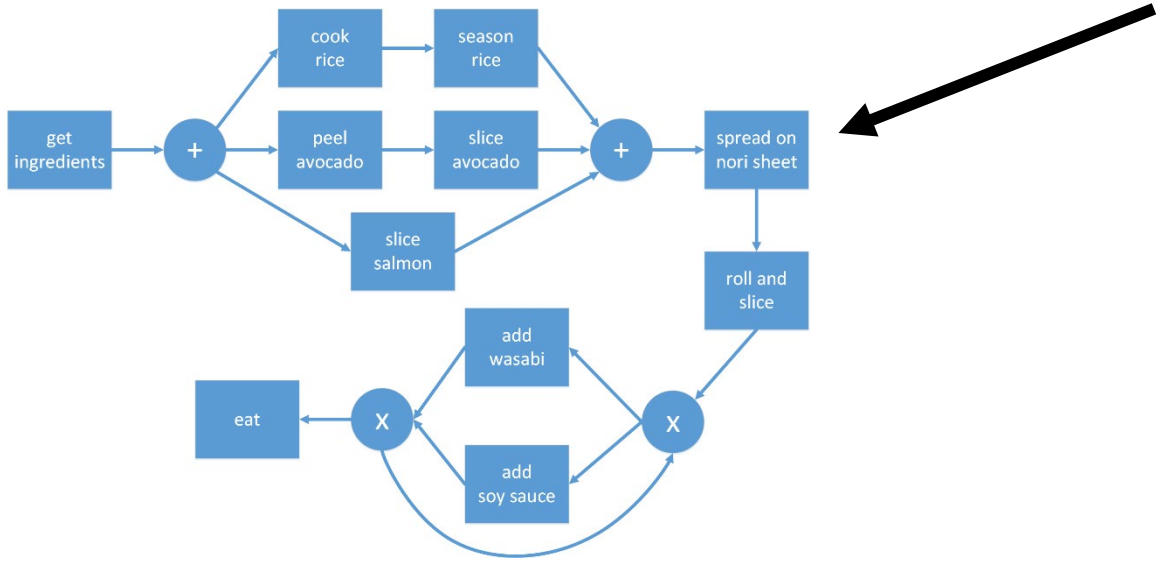






Process Mining Task: Discovery

- Given an event log, find a process model which
  - must be able to replay the log ⇒ $Fitness$
  - simplifies as far as possible ⇒ $Simplicity$
  - does not overfit the log ⇒ $Generalization$
  - does not underfit the log ⇒ $Precision$

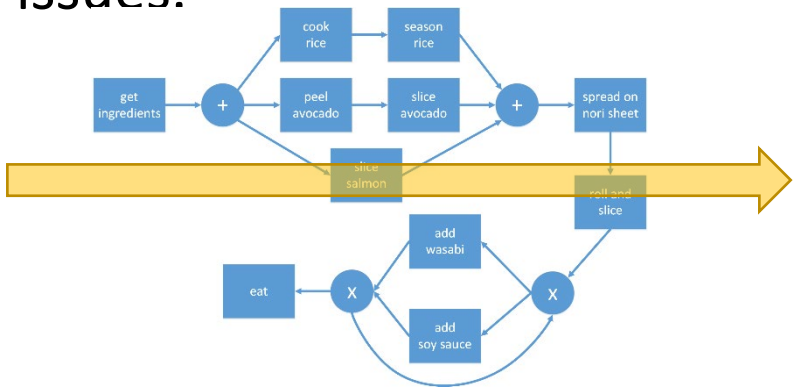


case id	activity	timestamp
	…	
Sushi 113	get ingredients	09:31
Sushi 239	slice salmon	09:35
Sushi 239	spread on nori sheet	09:42
Sushi 248	eat	09:43
Sushi 249	get ingredients	09:47
Sushi 113	cook rice	09:51
Sushi 239	roll and slice	09:51
Sushi 113	peel avocado	09:53
Sushi 239	add soy sauce	09:54
Sushi 239	add soy sauce	09:55
Sushi 239	eat	09:57
	…	

Process Mining Task: Conformance Checking

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

cook rice, add wasabi,
roll and slice, eat



conform

non-conform

- A case instance can perform better than others. Then reveal the beneficial deviations to improve the general workflow.
- If the case performs worse, identify the root cause to avoid misbehavior.

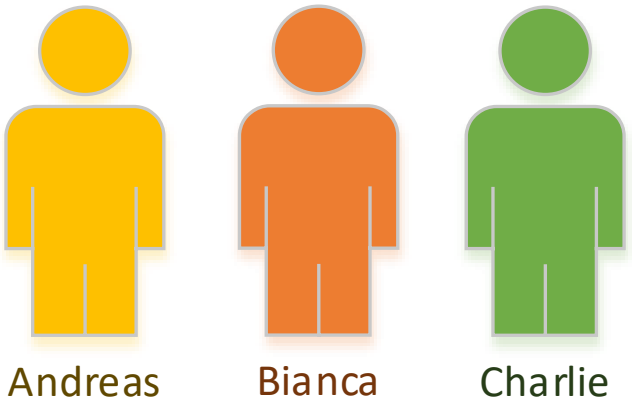
Housebreaking

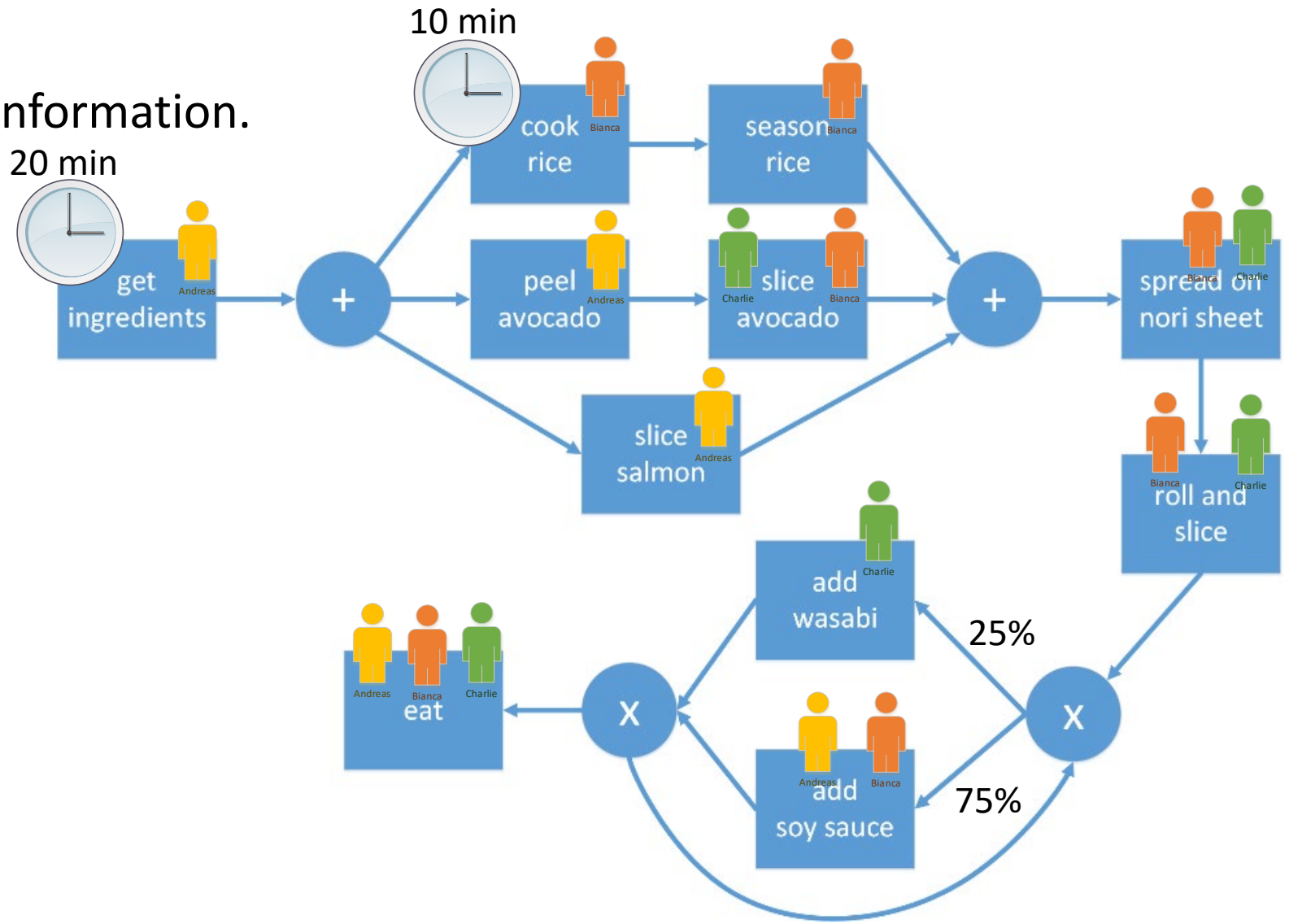
Trails

Tool choice

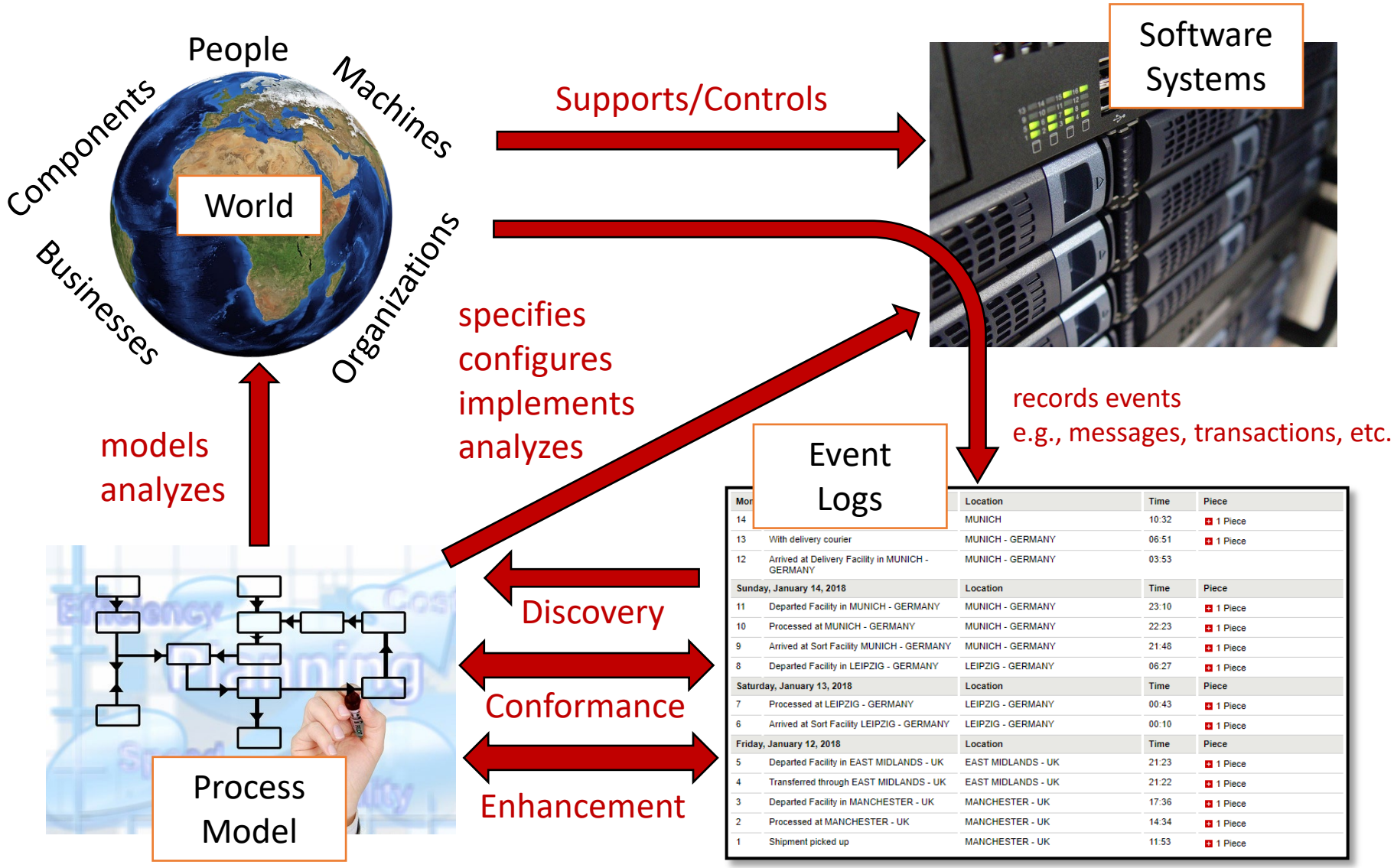
Process Mining Task: Enhancement

- Given a process model, augment with additional information.
  - Temporal information
  - Social networks
  - Organisational roles
  - Decision rules





Information Flow of Event Data



Process Mining Tools (community)

- ProM
  - Web page: promtools.org
  - First version in 2010
  - Java-based
  - Provides many algorithms in a GUI

- Pm4py
  - Web page: pm4py.org
  - First version in 2019
  - Python-based
  - Several algorithms available

Agenda

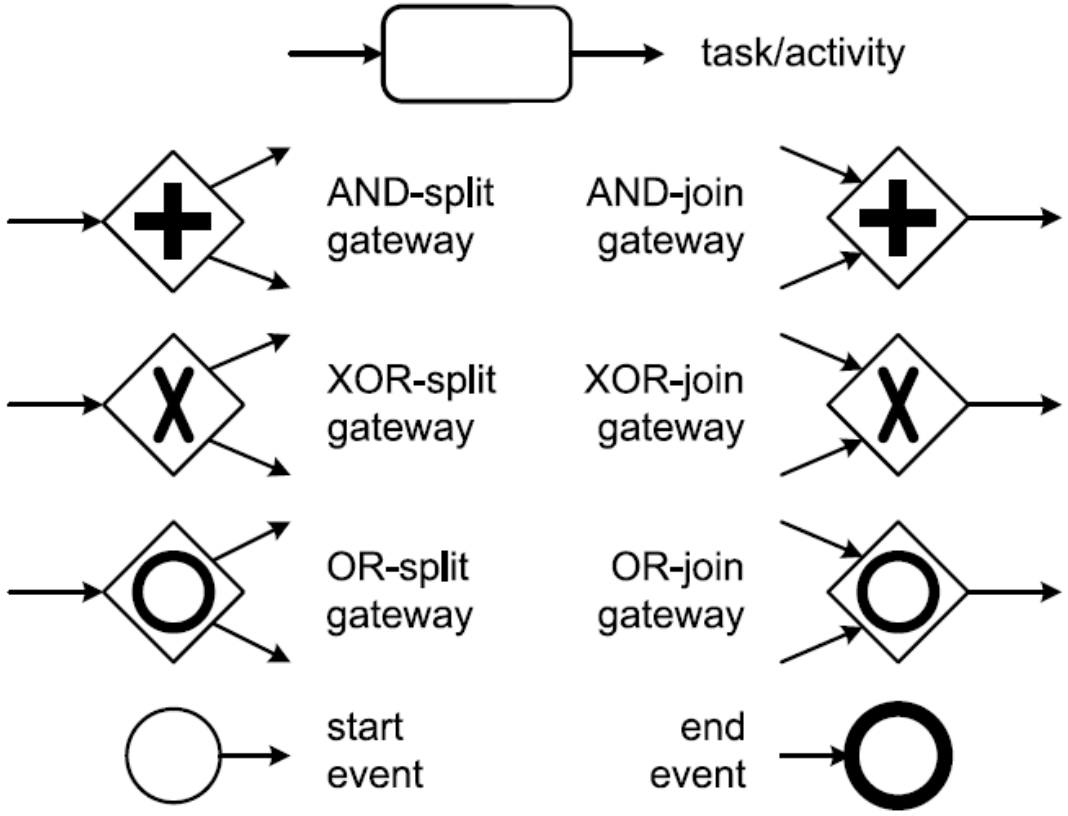
1. Introduction
2. Basics
3. Supervised Methods
4. Unsupervised Methods
5. Process Mining
   - Introduction
   - Process Models
   - Log Files
   - Process Discovery
   - Conformance Checking

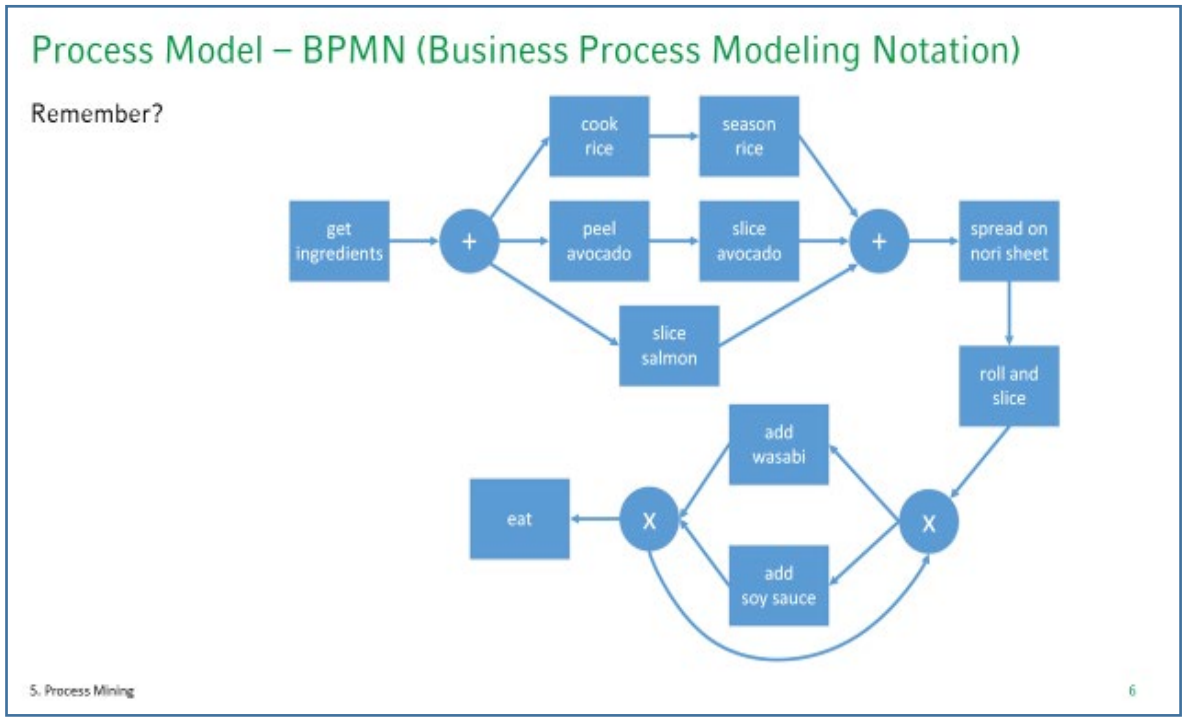
Process Models – Motivation

- Purposes to use Process Models
  - Predetermine operational processes in the form of guidelines
    - Descriptive vs. prescriptive model
  - Visualization of processes
  - Process reasoning
  - Analysis of given processes
    - Starting point for initial implementation and re-design
    - Distribution of responsibilities
    - Planning and controlling
    - Compliance checking
    - Performance prediction via simulation
    - …
- Benefits of Process Models
  - Insights by changing perspectives and highlights
  - Specification / documentation for certifications or legal contract purposes
  - Verification of executions to reveal problems
  - Performance analysis to identify issues like bottlenecks
  - Simulation (digital twin) to experiment virtually with varying settings

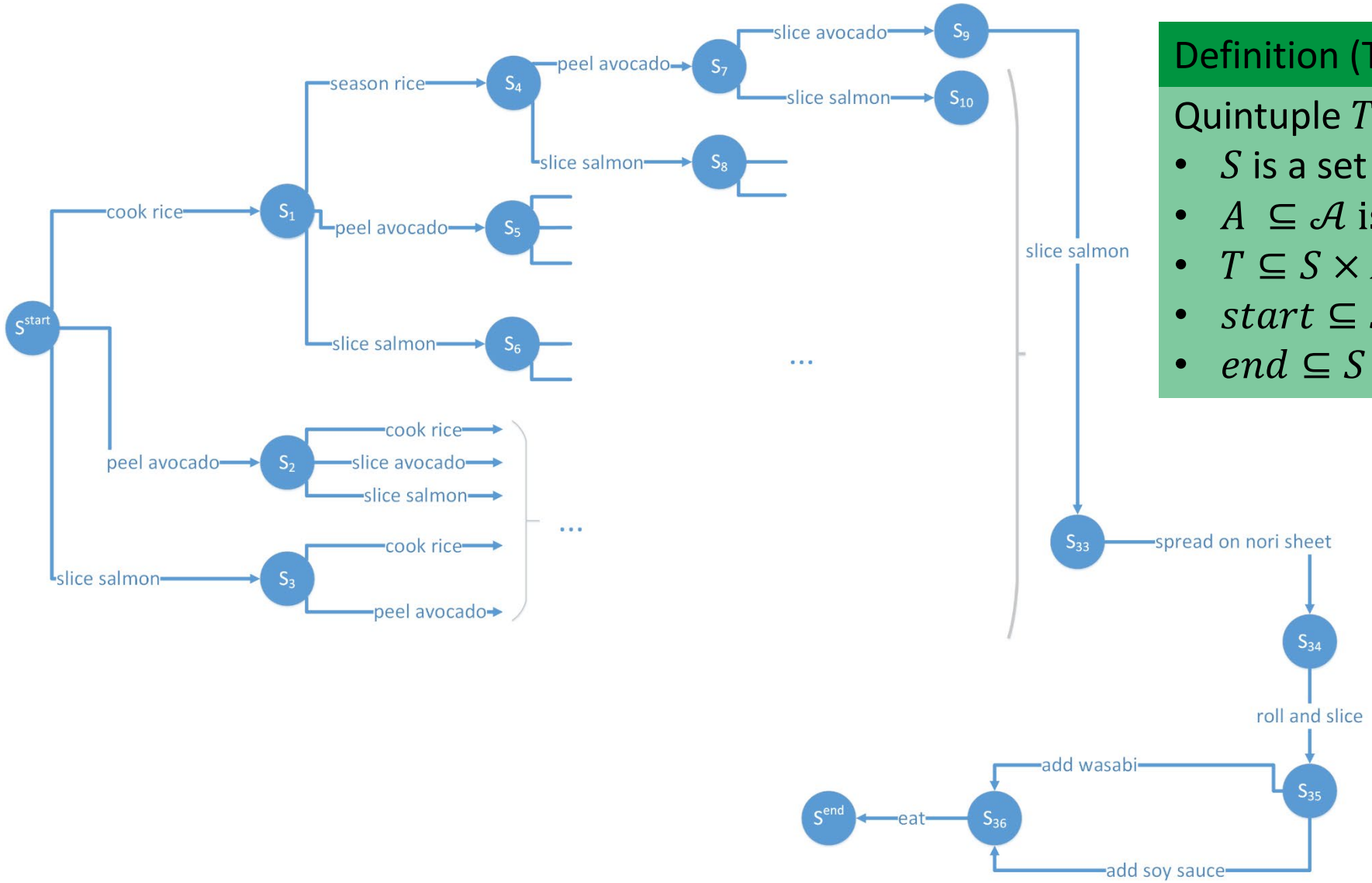
Process Model – BPMN (Business Process Modeling Notation)

Exemplary subset of elements contained in BPMN





Process Model – Transition System

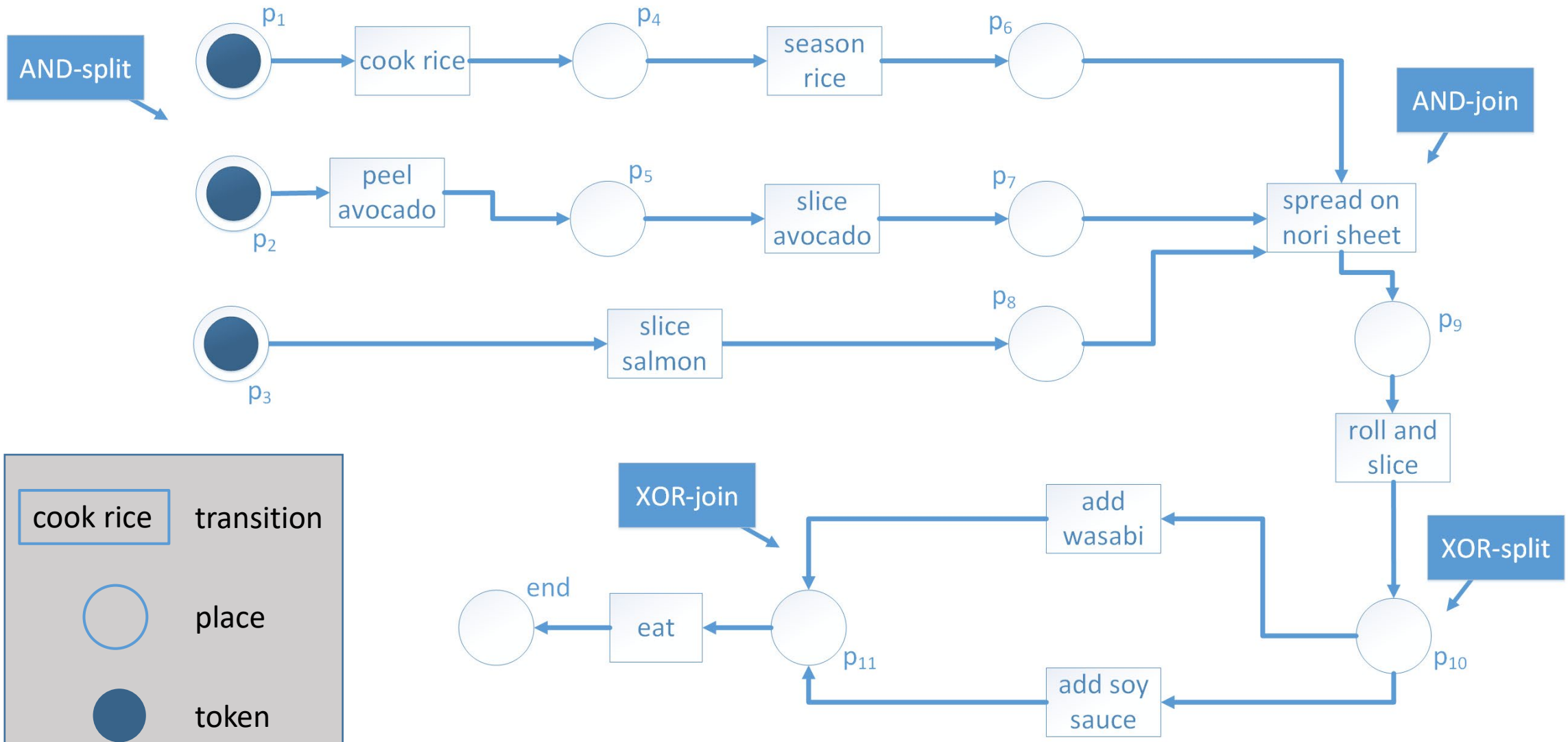


Definition (Transition system)

Quintuple $T = (S, A, T, start, end)$ where

- S is a set of *states*
- $A \subseteq \mathcal{A}$ is a set of *activities*
- $T \subseteq S \times A \times S$ is a set of *transitions*
- $start \subseteq S$ is the set of *initial states*
- $end \subseteq S$ is the set of *final states*

Process Model – Petri Nets



Process Model – Petri Nets

- As already seen the Petri net is a bipartite graph

Definition (Petri net)

A Petri net $N = (P, T, F, b)$ consists of
- Two disjoint sets ($P \cap T = \emptyset$) representing places P and transitions T
- A flow relation $F \subseteq (T \times P) \cup (P \times T)$
- A start place $b \in P$ (without loss of generality: a start set $B \subseteq P$)

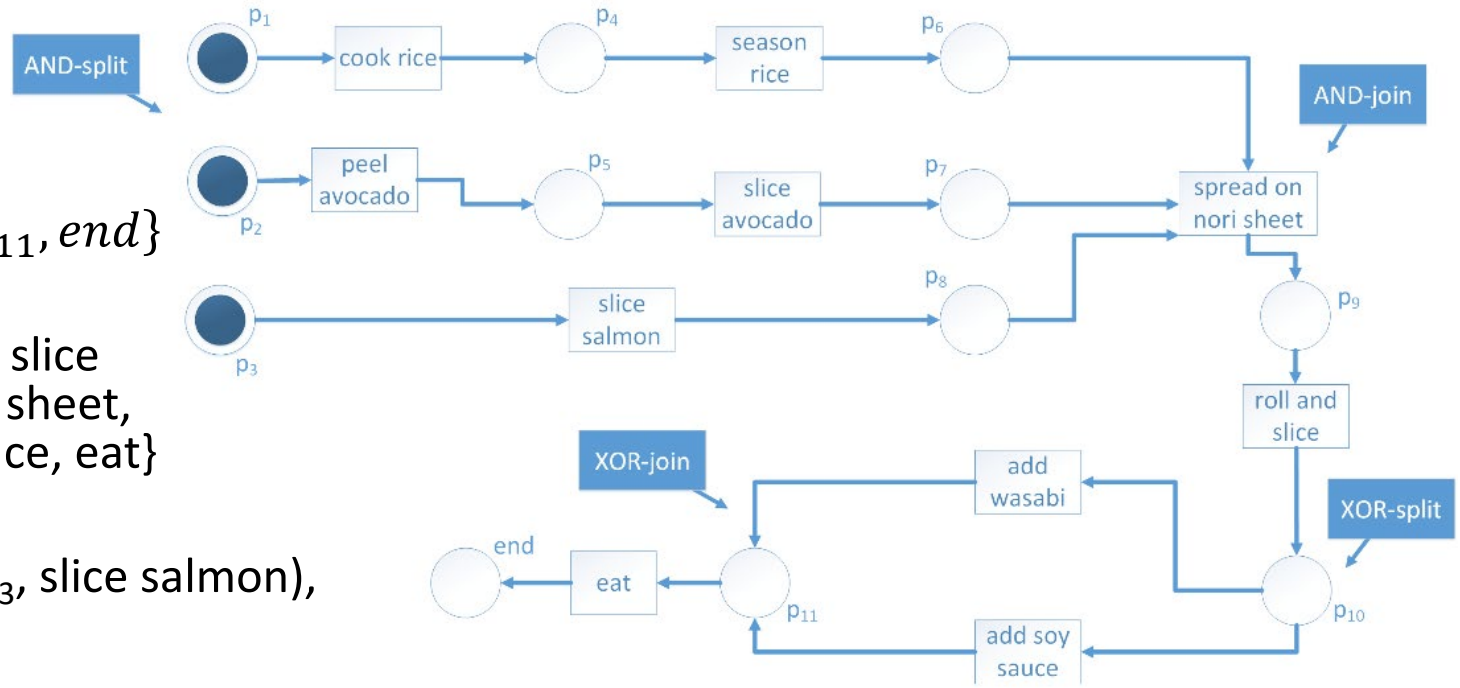
In general, we assume
- a single start place $b \in P$ (without preceding transitions), and
- a single end place $e \in P$ (without subsequent transitions)

- **Previous example as a Petri net**

$P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}, end\}$

$T =$ {cook rice, season rice, peel avocado, slice avocado, slice salmon, spread on nori sheet, roll and slice, add wasabi, add soy sauce, eat}

$F =$ {(p_1, cook rice), (p_2, peel avocado), (p_3, slice salmon), (cook rice, p_4), (peel avocado, p_5), …}



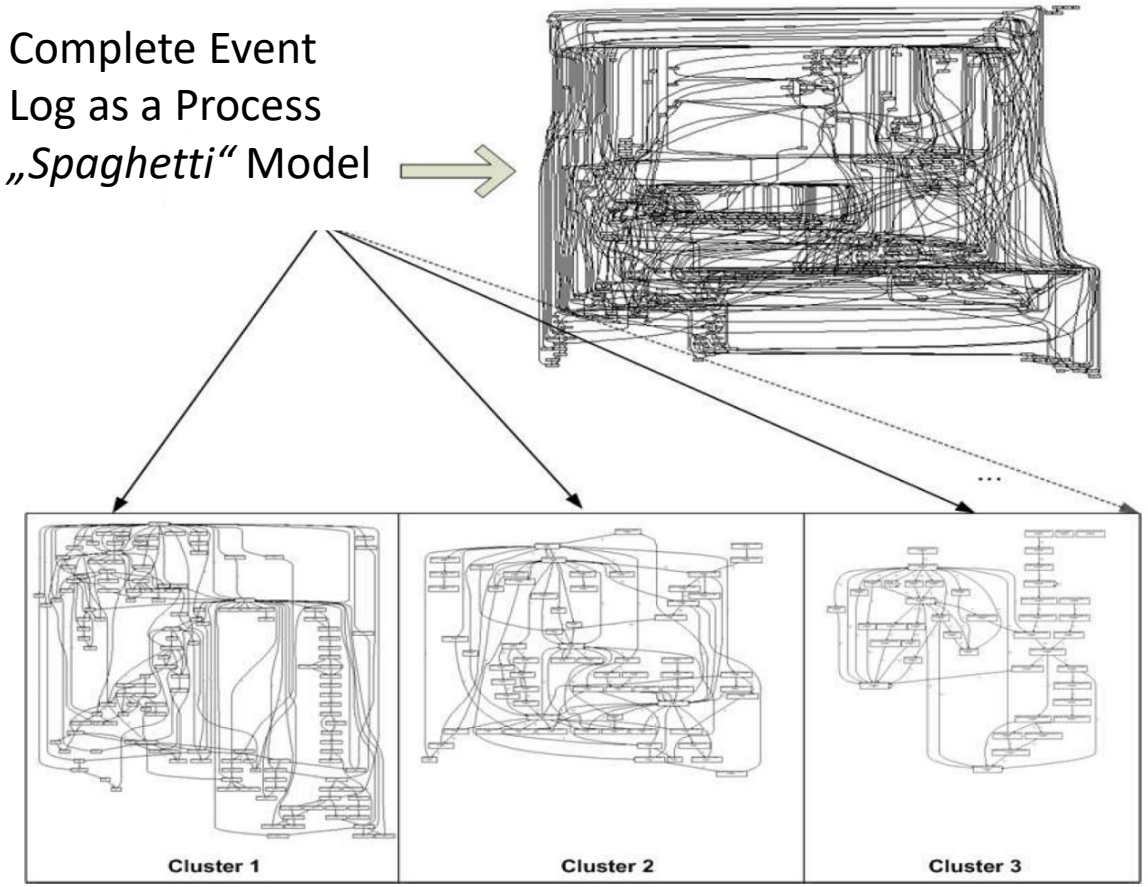
Trace Clustering – Motivation

High *diversity:*

Single cases differ significantly from one another

→ possibly very complex models

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



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

Process Models – Roundup

Process models considered so far

- Transitions systems
- BPMN
- Petri Nets (and Workflow Nets)
- Trace Clustering

There are still others like

- Reachability graphs
- Causal networks
- …

Benefits

- Process analysis gets simplified
- Predict performance via simulation
- Predetermine guidelines
- Purpose determines outcome
- …

Agenda

1. Introduction
2. Basics
3. Supervised Methods
4. Unsupervised Methods
5. Process Mining
   - Introduction
   - Process Models
   - Log Files
   - Process Discovery
   - Conformance Checking

Event Logs as Starting Point

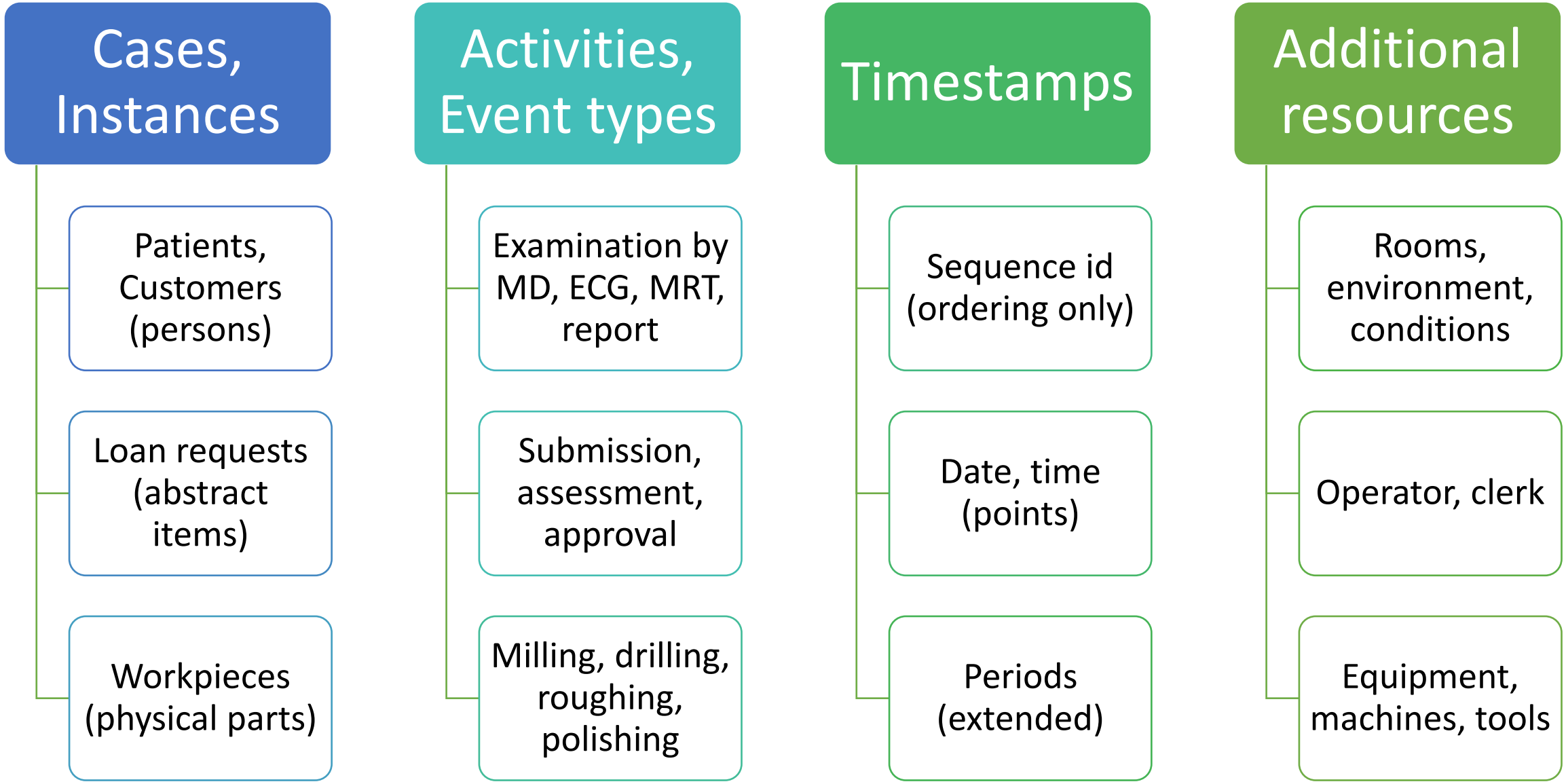
case id	activity	timestamp	resource 1	resource 2	execution quality
		...			
Sushi 113	get ingredients	09:31	Andreas	bag	good
Sushi 239	slice salmon	09:35	Bianca	knife 1	medium
Sushi 239	spread on nori sheet	09:42	Bianca		very good
Sushi 248	eat	09:43	Charlie		-
Sushi 249	get ingredients	09:47	Andreas	bag	good
Sushi 113	cook rice	09:51	Bianca	rice cooker 3	poor
Sushi 239	roll and slice	09:51	Charlie	knife 1	good
Sushi 113	peel avocado	09:53	Andreas	knife 2	poor
Sushi 239	add soy sauce	09:54	Bianca		good
Sushi 239	add soy sauce	09:55	Bianca		poor
Sushi 239	eat	09:57	Andreas		-
		...			

Event Logs Technically

- Data collection mostly fully automated.
- Process-Aware Information Systems (PAIS)
  - ERP (Enterprise-Resource Planning) [SAP, Oracle]
  - BPM (Business Process Management) [IBM BPM]
  - CRM (Customer Relationship Management)
- Popular data format: XES
  - XML-based
  - easy to understand

```
<?xml version="1.0" encoding="UTF-8" ?>
<log xes.version="2.0" xes.features="arbitrary-depth" xmlns="http://www.xes-standard.org
    /">
    <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.
        xesext"/>
    <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.xesext"/>
    <global scope="trace">
        <string key="concept:name" value=""/>
    </global>
    <global scope="event">
        <string key="concept:name" value=""/>
        <date key="time:timestamp" value="1970-01-01T00:00:00.000+00:00"/>
        <string key="system" value=""/>
    </global>
    <classifier name="Activity" keys="concept:name"/>
    <classifier name="Another" keys="concept:name system"/>
    <float key="log attribute" value="2335.23"/>
    <trace>
        <string key="concept:name" value="Trace number one"/>
        <event>
            <string key="concept:name" value="Register client"/>
            <string key="system" value="alpha"/>
            <date key="time:timestamp" value="2009-11-25T14:12:45:000+02:00"/>
            <int key="attempt" value="23">
                <boolean key="tried hard" value="false"/>
            </int>
        </event>
        <event>
            <string key="concept:name" value="Mail rejection"/>
            <string key="system" value="beta"/>
            <date key="time:timestamp" value="2009-11-28T11:18:45:000+02:00"/>
        </event>
    </trace>
</log>
```

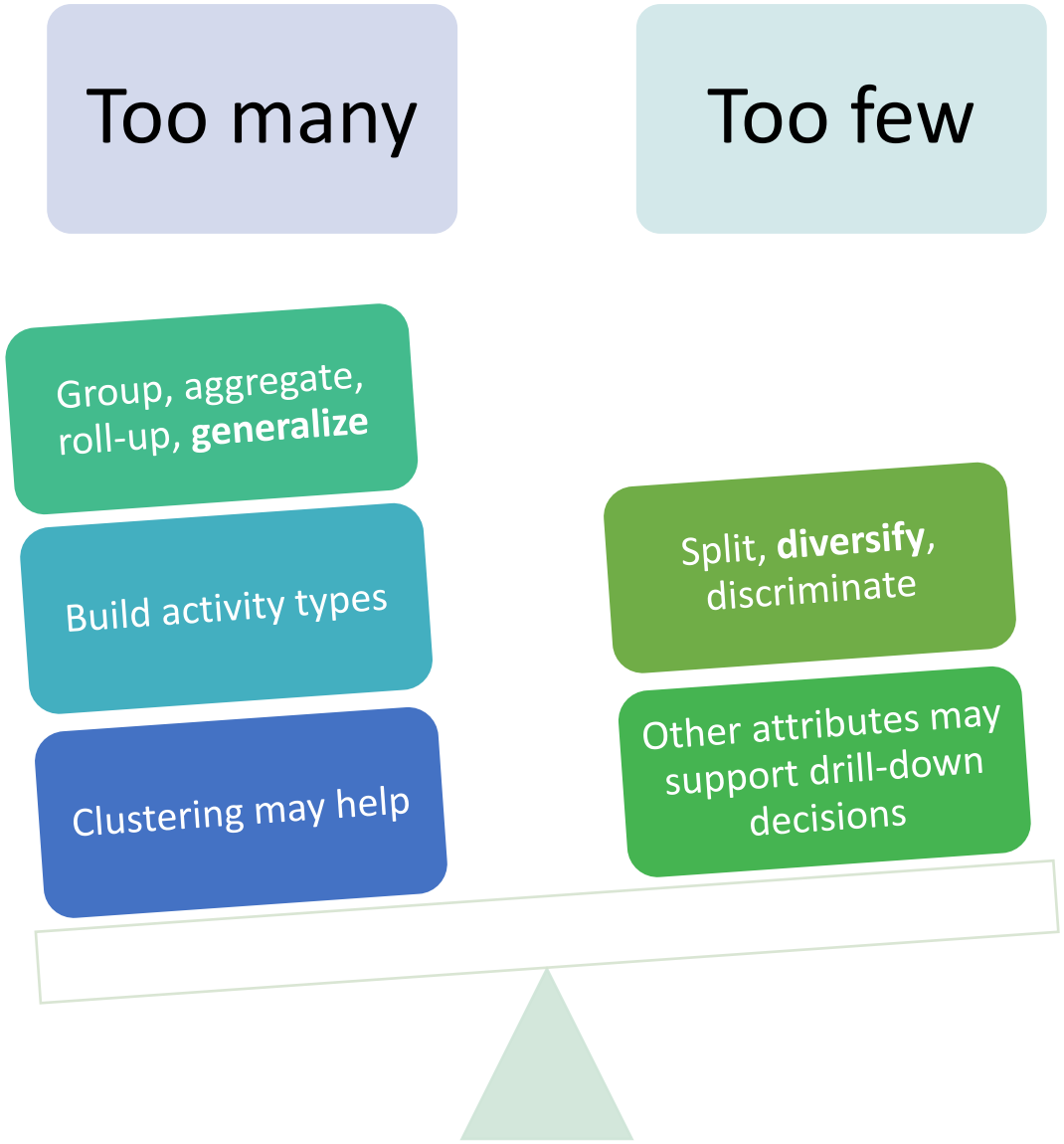
Ingredients of process log files



Are **activities** easy to identify? – tune their granularity

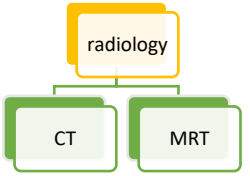
Activities, Event types

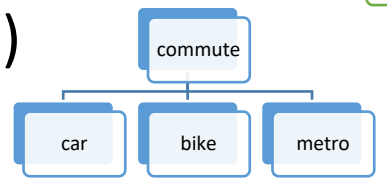
- Too many: yield *confusing models*
- Useful granularity
- Too few: yield *poor models*

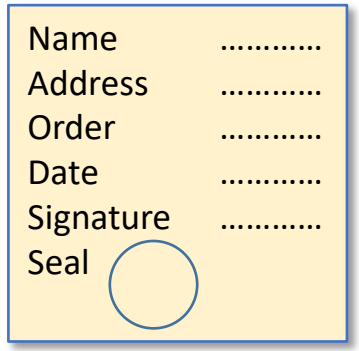


Examples for tuning the granularity of activities

- „Too many“ individual activities
  - Process models tend to be confusing and incomprehensible (overfitting?)
  - Tune by generalization of activities to activity types („roll-up“)
    - Example: group „CT“ and „MRT“ to „radiology“
    - Example: group „car drive“, „bike ride“, „metro trip“ to „commute“
  - Group individual steps to subprocesses
    - Example: merge „fill in name“ – „fill in address“ – „sign“– „seal“ to „finish form“
  - Strategies for aggregation of activities
    - Exploit existing generalization hierarchy
    - Cluster activities by taking additional information into account







Design choices for tuning the granularity of activities

- „Too few“ individual activities
  - Tend to poor models, overgeneralization
  - Tune by splitting, diversifying, discriminating activites („drill down“)
  - Strategy: Derive discrimination of activities from context information
    - Take additional attributes from the logfile entries into account
    - Explore and exploit external information from the application environment

- At which stage to tune?
  - Early, at preprocessing: Transform log file / traces before model creation
  - Interactively, at analysis time: Allow for interactive tuning by users

Are **cases** easy to identify?

- Cases are individuals, they cannot be overfitted per se.
- Their traces are extracted by grouping the respective events.

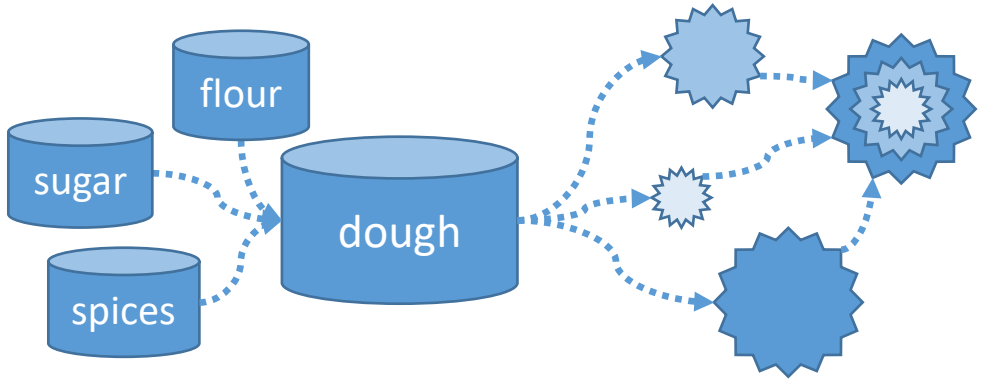
- Sometimes there are individual parts but their case ids are missing
  - People in a crowd (visitors in stadion, tourists in hot-spot areas)
  - Vehicles in traffic
  - Bulk material in production

- If we want to follow the individuals rather than treating them as bulk
  - Prefer particle model (e.g., state systems) over fluid dynamics (partial differential equations) if we want to explain the diverse behaviors in the traces

- Strategies
  - Assume some traces and fit them to the bulk log
  - May help for remaining time predictions, root cause analysis of system malfunctions, etc.

Metamorphosis of cases

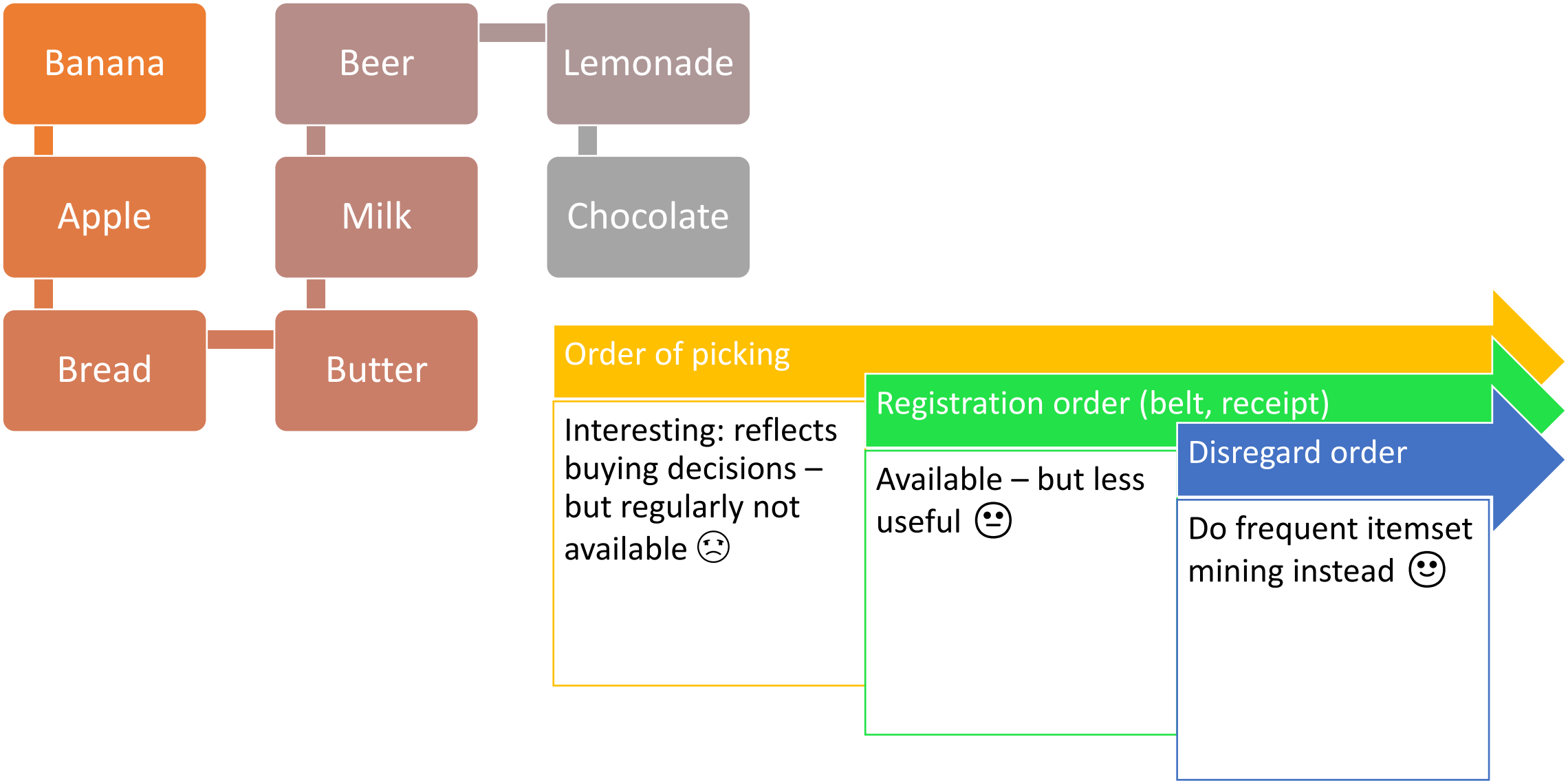
- Complex process structures may resist to identify cases
  - In production, material is merged or separated, or it comes in batches
  - I.e., the structure of cases undergoes some evolution
- Examples
  - Biscuits: packages of flour, sugar, spices → portion of dough → cut pieces → assembled cookies
  - Car assembly: engine, chassis, tires, screws, fluids
- How to model those processes?
  - Identify subprocesses at individual stages and identify cases anew at hand-over points
  - Keep track of parts in assemblies (engine, tires)
  - Switch case structure (screws, fluids)



Temporal aspects, type of measurements

- Static data
  - Disregard time at all
  - Customers, transactions, gene expression, etc.
- Sequences
  - Order is important, time ist not
  - Text, genetic sequences, protein sequences
- Time series
  - Periodic or sporadic measurements
  - Temperature logs, traffic monitoring, etc.
- Process logs
  - Series of events / activities, event logfiles
  - Patient treatment, procurement, admissions
- Stream analytics
  - Real-time monitoring, online algorithms
  - Analyze concept drift, online learning

Timestamps: does order matter? – supermarket example



Ordering in frequent pattern mining (sequences, traces, itemsets)

Totally ordered

Sequences

- Frequent subsequences
- Prefixspan
- etc.

Time series

- Motif search
- Subspace clustering
- Dynamic Time Warping, etc.

Partially ordered

Concurrent / Interleaved Traces

- **Process Mining**
- Heuristic Miner, Inductive Miner, Trace clustering, CMRules, etc.

Unordered

Itemsets

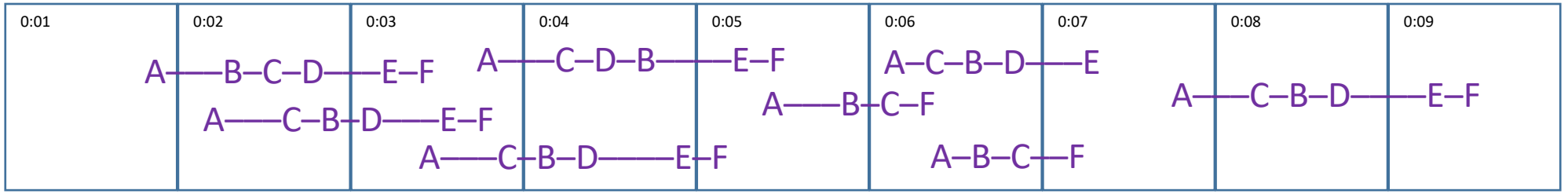
- Apriori algorithm
- FP-growth

How to treat temporal attributes

- Disregard time at all
  - Marginal distributions, without time
  - Project to time-less attributes
- Abstract time, unit steps
  - Positions in sequences only
  - Map indices, not points in time
- Quantized, generalized time
  - Roll-up in OLAP, „group-by“ in SQL
  - By week / month / quarter / year / decade / century
- Normalized, aligned time
  - Shifted to origin, start at zero
  - Relative durations, stretched / squeezed
- Original, as is
  - Absolute periods, durations, resolutions

Timestamps: too coarse resolution?

- Example: time resolution is in seconds, but activities are processed below seconds



- Model coincidences of events within same frame as concurrent activities
- Maybe cuts of traces help in identifying a „normalized“ process model
- Limitations in detection of non-conformant traces

Agenda

1. Introduction
2. Basics
3. Supervised Methods
4. Unsupervised Methods
5. Process Mining
   - Introduction
   - Process Models
   - Log Files
   - Process Discovery
   - Conformance Checking

Motivation

Process models are generated either prescriptive or descriptive

- Prescriptive
  - 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 create process graphs

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

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 $\boldsymbol{a >_L b}$
$\Leftrightarrow \exists\, trace\ \sigma = \langle t_1, t_2, t_3, \dots, t_{n-1} \rangle\ and\ i \in \{1,2, \dots, n-2\}$
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:
ii) Find the potential parallel tupels and mark them in the table

	a	b	c	d
a				
b				
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)}

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“ $\boldsymbol{a \parallel_L b}$
$\Leftrightarrow a >_L b\ and\ b >_L a$

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:
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			$\#_L$	$\parallel_L$
d			$\parallel_L$	$\#_L$

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

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:
(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)}

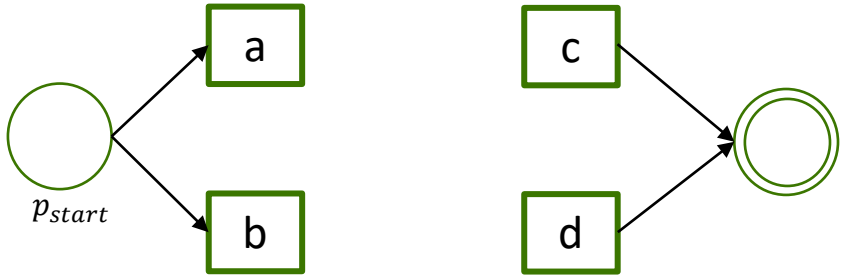
Process Discovery Algorithm „α-Miner“

- Execute 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, $\boldsymbol{T_{out}}$
  - …
- Derive a Workflow net
  - Let the activities A be the set of **transitions**
  - A starting place is created and connected to each node in $\boldsymbol{T_{in}}$
  - 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“

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

- Execute 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.

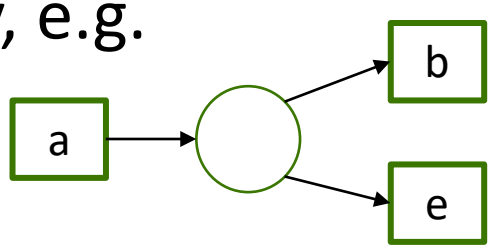
  $(\{a\},\{c\}),(\{a\},\{d\}),(\{a\},\{c,d\}) \Longrightarrow (\{a\},\{c,d\})$

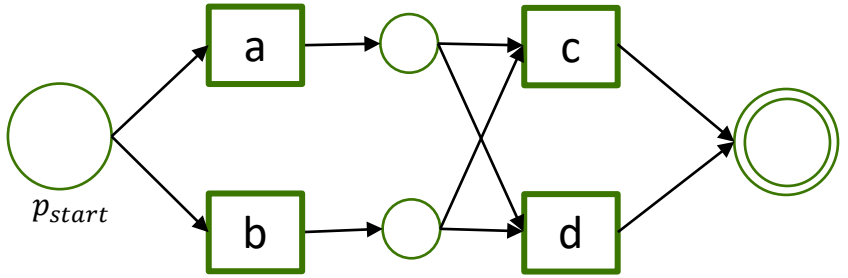
valid set of „maximal pairs“:

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

- A place is added in between A and B and connected accordingly, e.g.

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





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*

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

[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 $>_L$

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

$>_L$	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

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

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

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

$|a \Rightarrow_L b| \to 1$, 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

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

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

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

$|a \Rightarrow_L b| \to 1$, 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

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

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

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

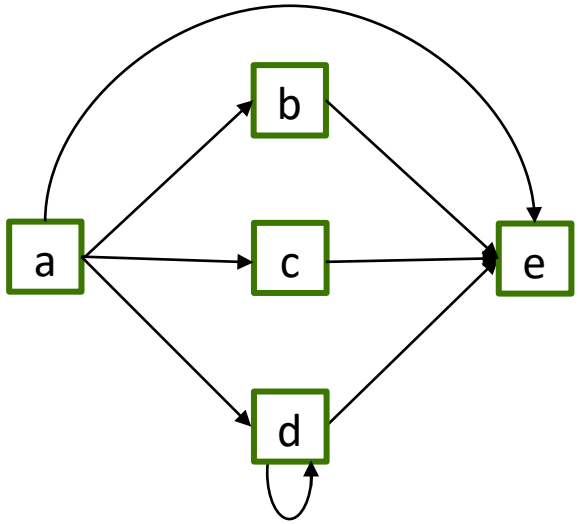
Process Discovery Algorithm „Heuristics Miner“

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:



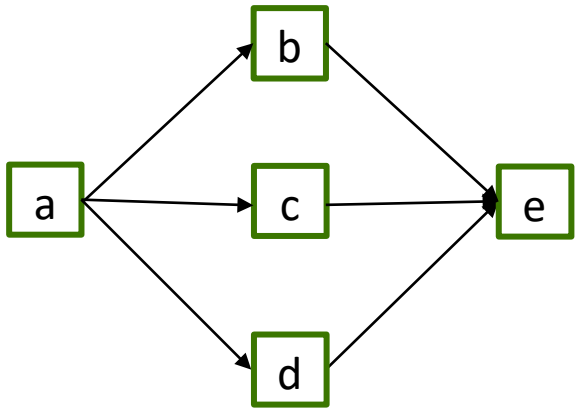
Process Discovery Algorithm „Heuristics Miner“

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

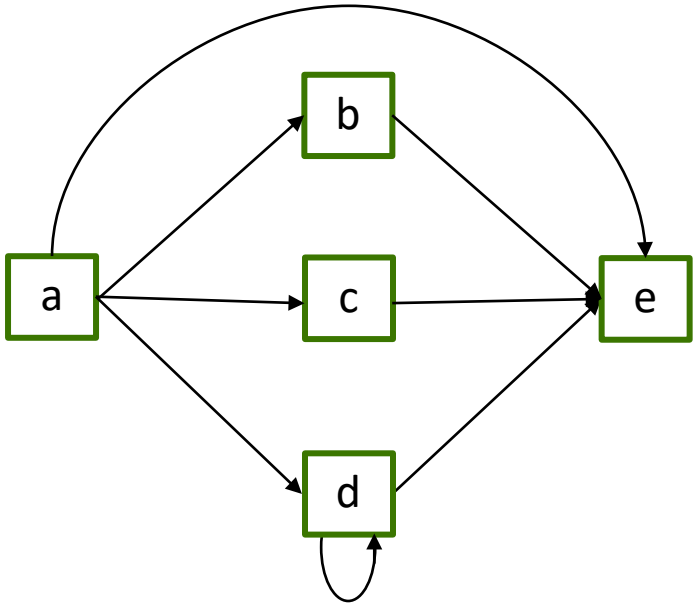
$\lvert\Rightarrow_L\rvert$	a	b	c	d	e
a	0	0.92	0.92	0.93	~~0.83~~ 0
b	~~-0.92~~ 0	0	0	0	0.92
c	~~-0.92~~ 0	0	0	0	0.92
d	~~-0.93~~ 0	0	0	~~0.80~~ 0	0.93
e	~~-0.83~~ 0	~~-0.92~~ 0	~~-0.92~~ 0	~~-0.93~~ 0	0

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

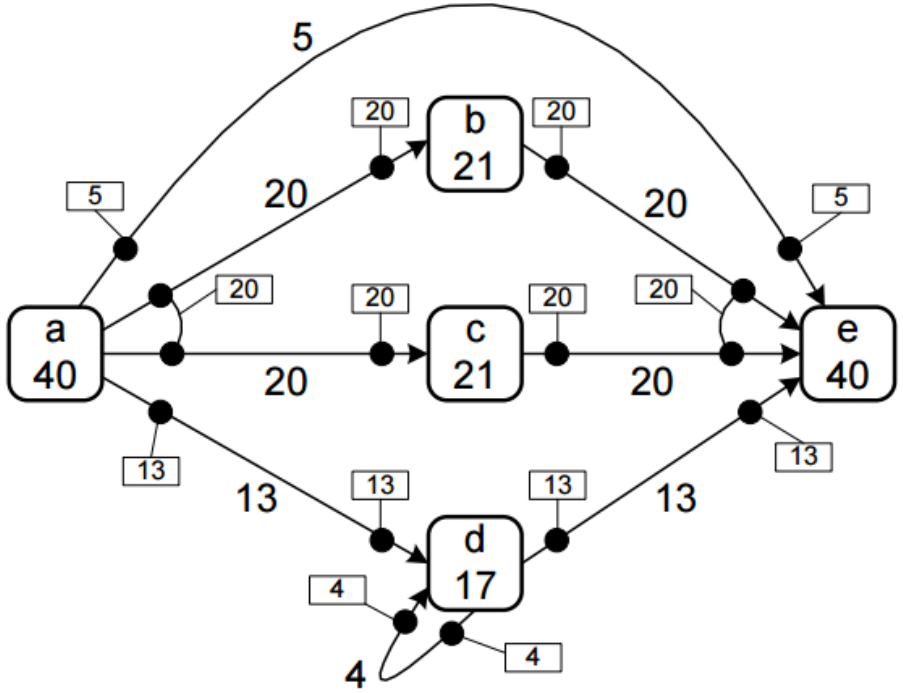


Process Discovery Algorithm „Heuristics Miner“

4. Last step – *not in this lecture*:
   dependency graph → causal net



→



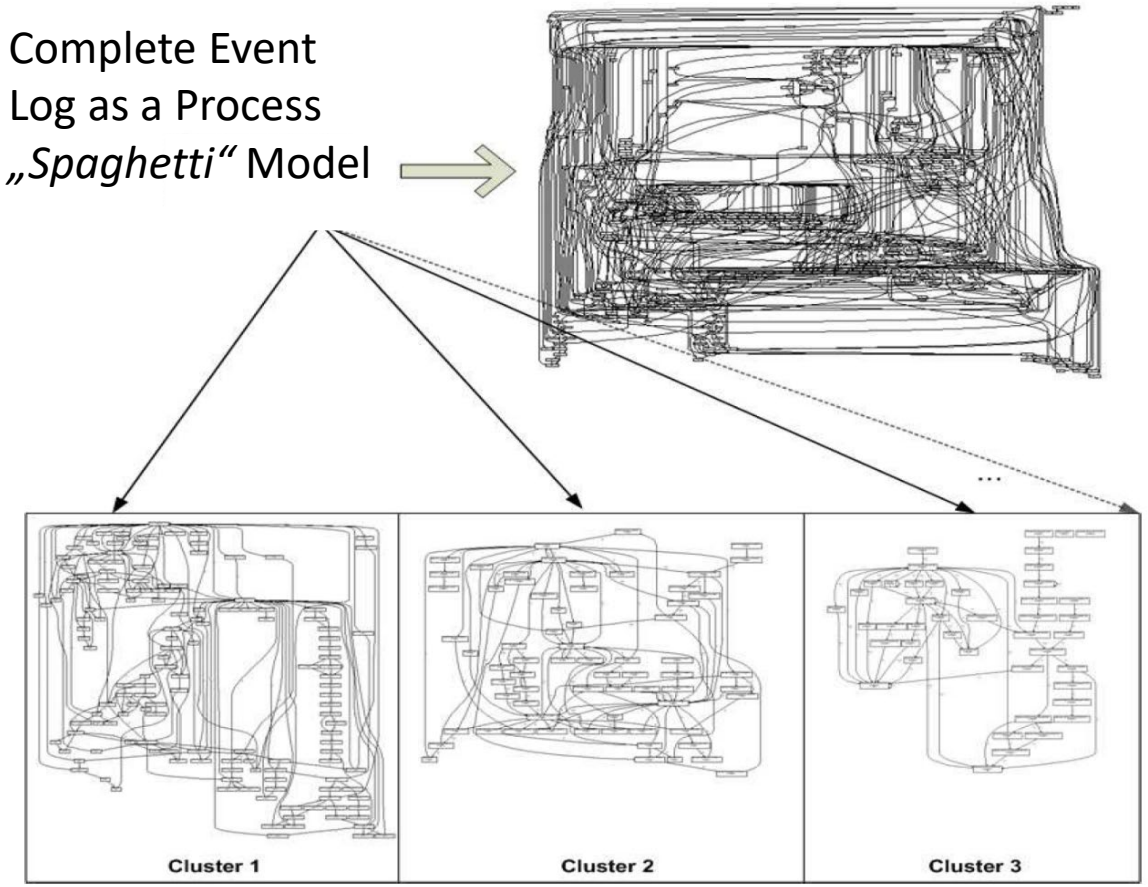
Trace Clustering – Motivation

High *diversity:*

Single cases differ significantly from one another

→ possibly very complex models

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



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

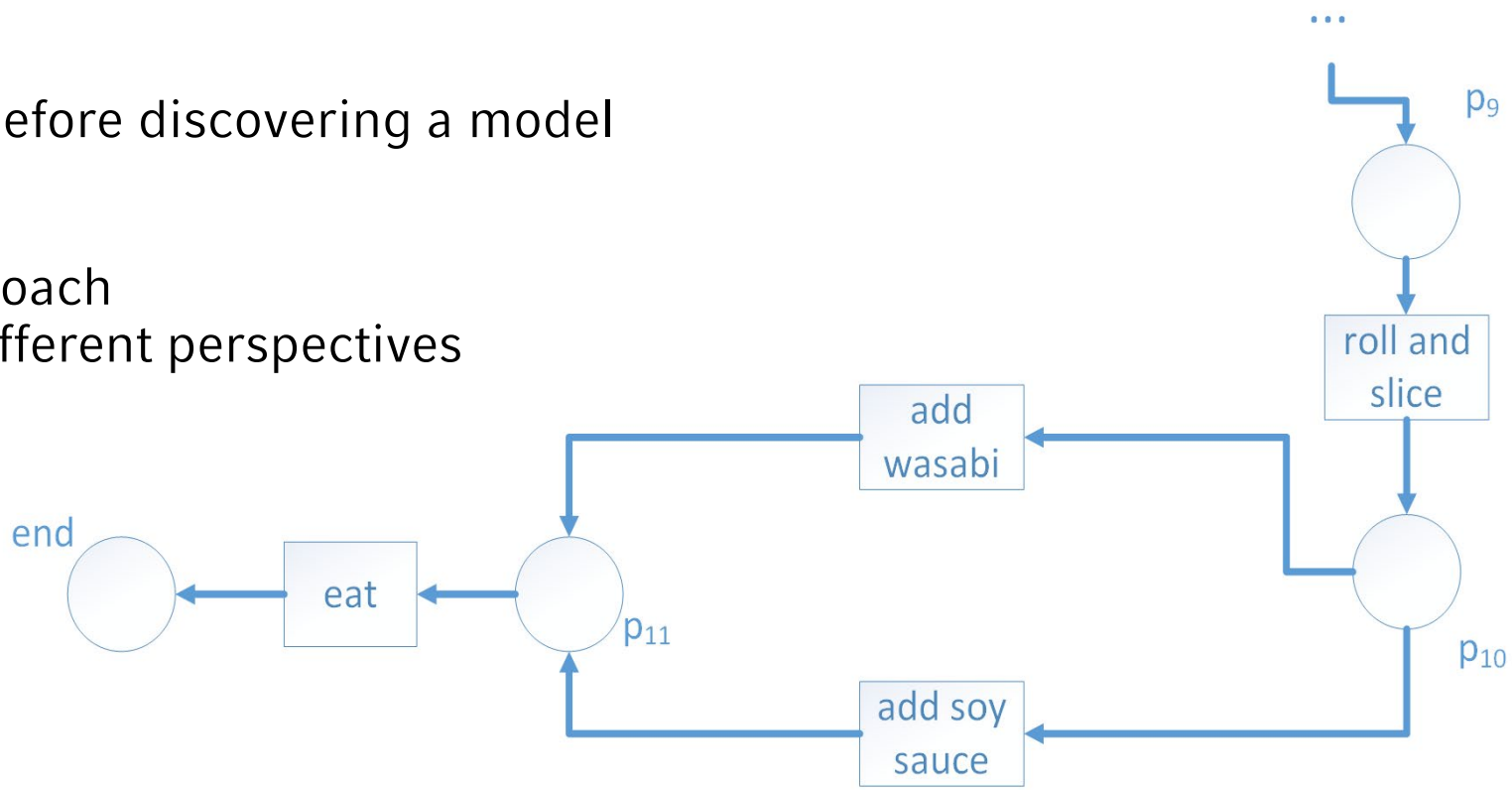
Trace Clustering – Motivation

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

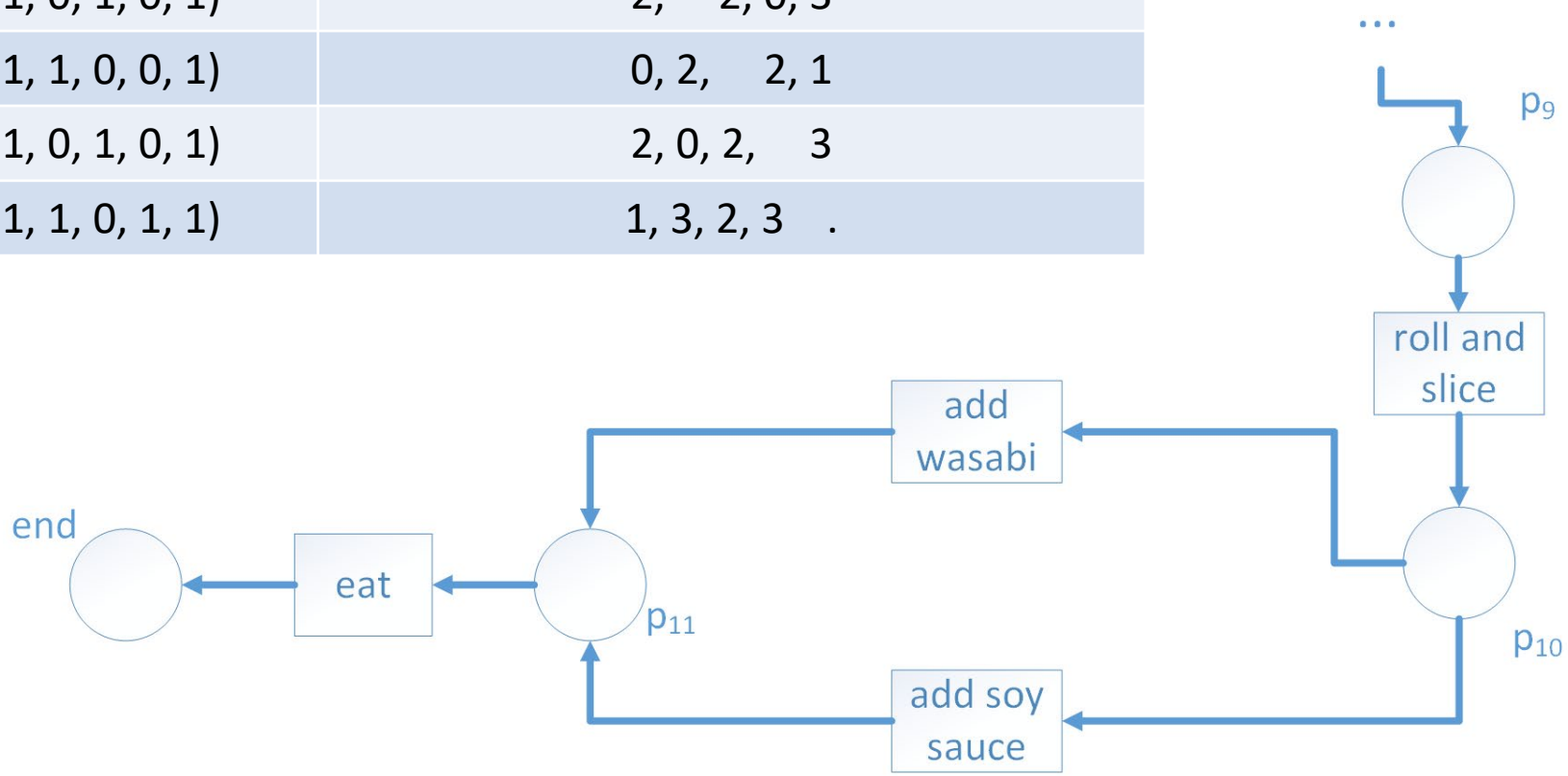
=> Embedding of *cases* into vector space necessary → **Profiles**

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

$\Rightarrow$ 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**

$\Rightarrow$ **Various clustering methods can be applied now**

Trace Clustering – Methods

More examples:

Transition profile:

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**

Performance profile:

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)

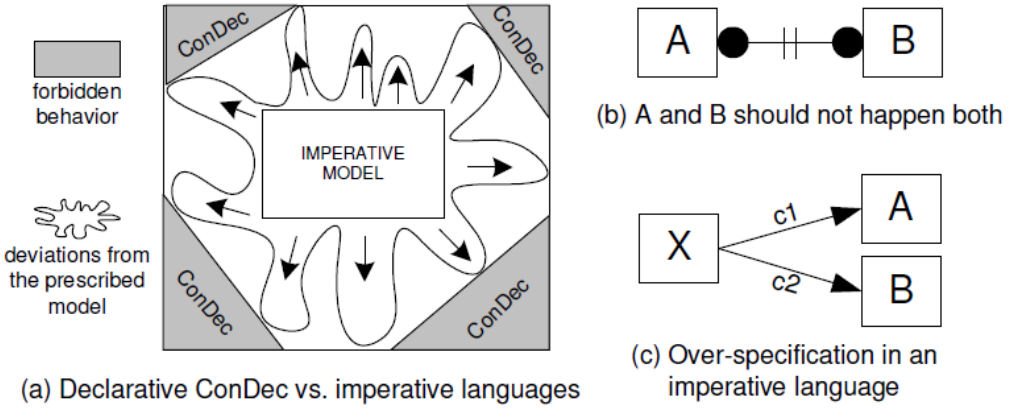
Process Discovery Algorithm – Some Others

- **„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.

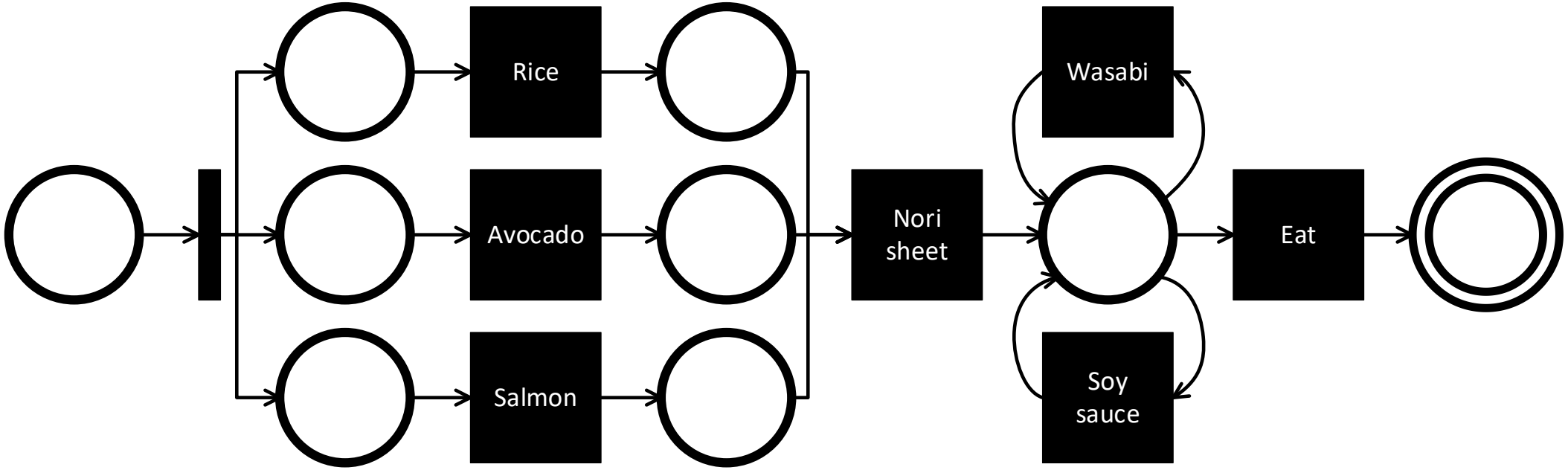
[4] Pesic, Maja, Helen Schonenberg, and Wil MP Van der Aalst. "Declare: Full support for loosely-structured processes." *11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007)*. IEEE, 2007.

Agenda

1. Introduction
2. Basics
3. Supervised Methods
4. Unsupervised Methods
5. Process Mining
   - Introduction
   - Process Models
   - Log Files
   - Process Discovery
   - Conformance Checking

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: 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?

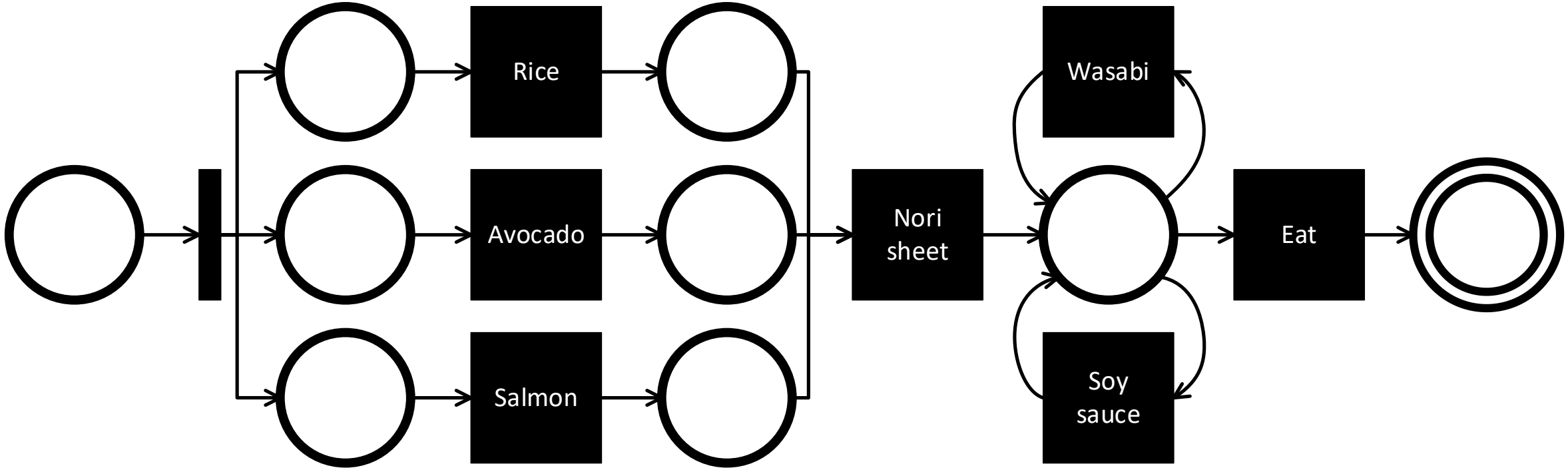
Image credit PSFK

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.

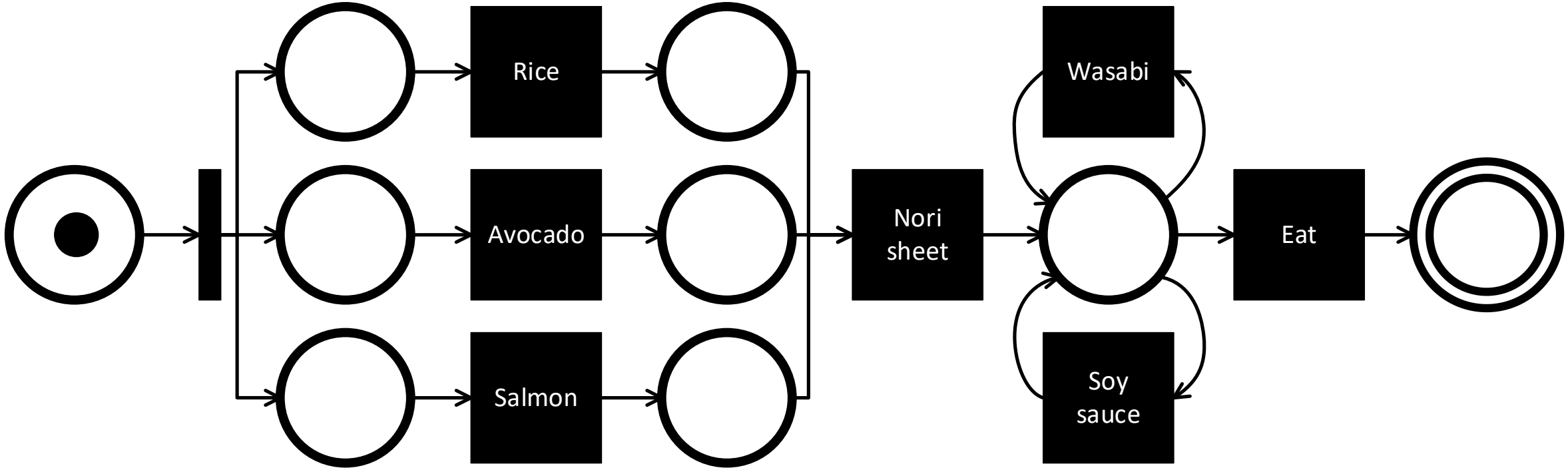
[1] A.K. Alves de Medeiros,W.M.P. van der Aalst, and A.J.M.M.Weijters. Quantifying Process Equivalence Based on Observed Behavior. *Data and Knowledge Engineering*, **64**(1):55–74, 2008.

Petri Net Membership Test



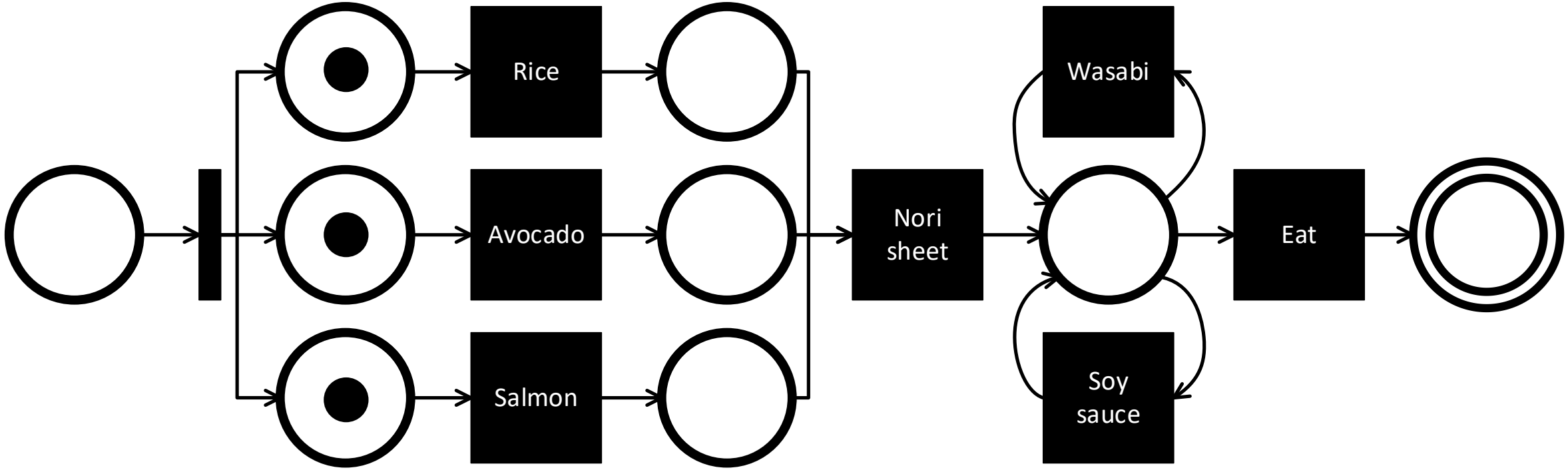
- Checking: <Avocado, Rice, Salmon, Nori, Eat>
- produced (p): 0 consumed (c): 0

Petri Net Membership Test



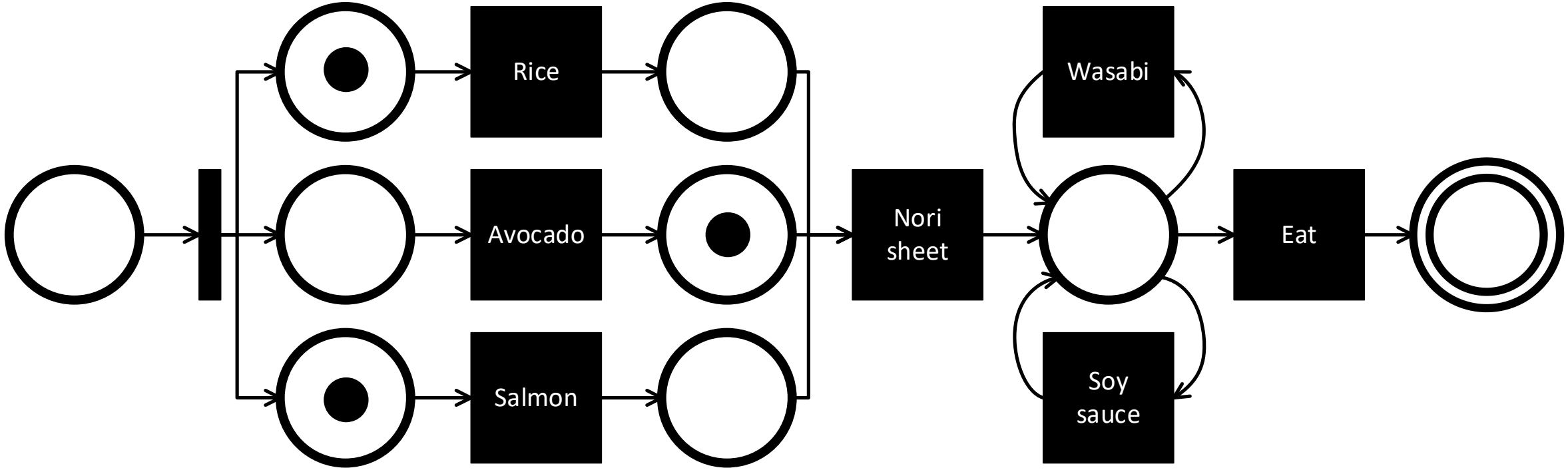
- Checking: <Avocado, Rice, Salmon, Nori, Eat>
- produced (p): 1 consumed (c): 0

Petri Net Membership Test



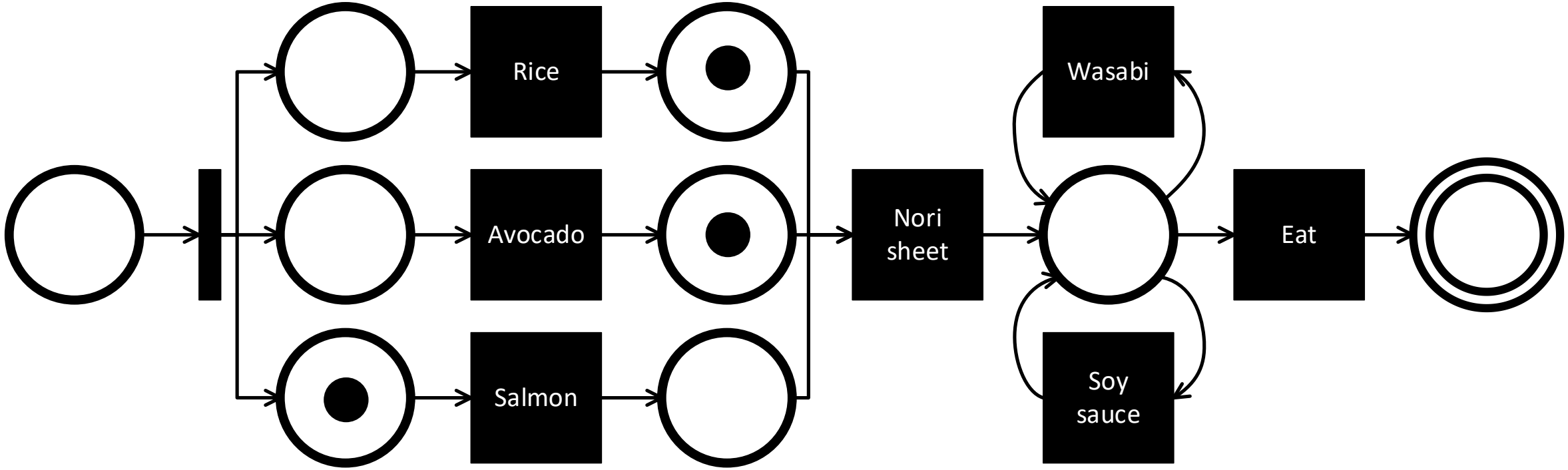
- Checking: <Avocado, Rice, Salmon, Nori, Eat>
- produced (p): 4 consumed (c): 1

Petri Net Membership Test



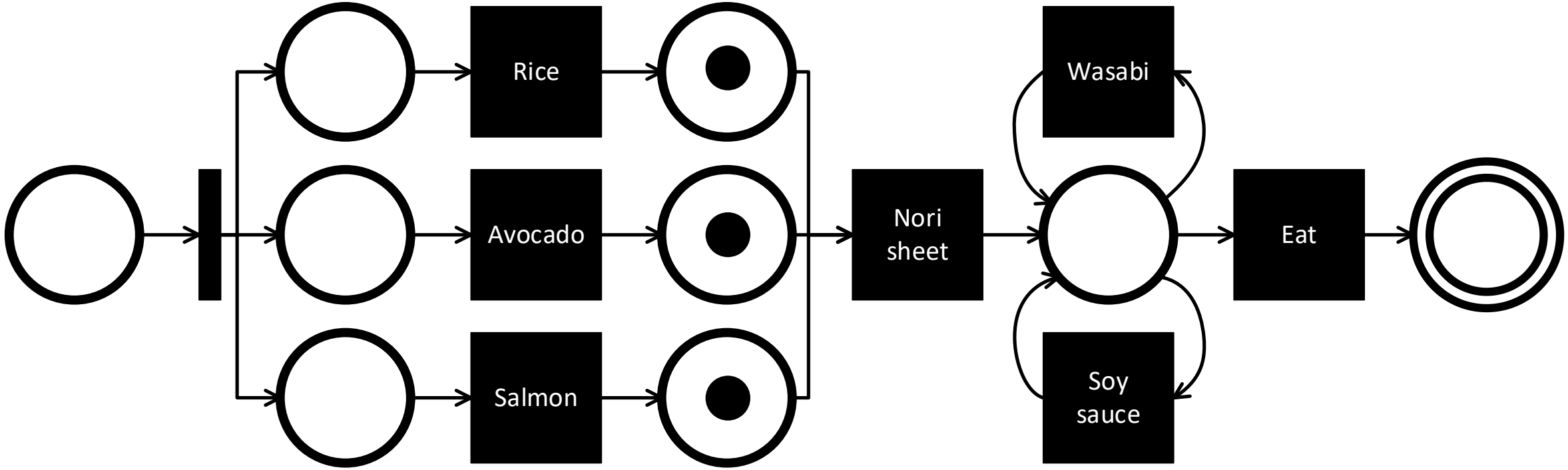
- Checking: <Avocado, Rice, Salmon, Nori, Eat>
- produced (p): 5 consumed (c): 2

Petri Net Membership Test



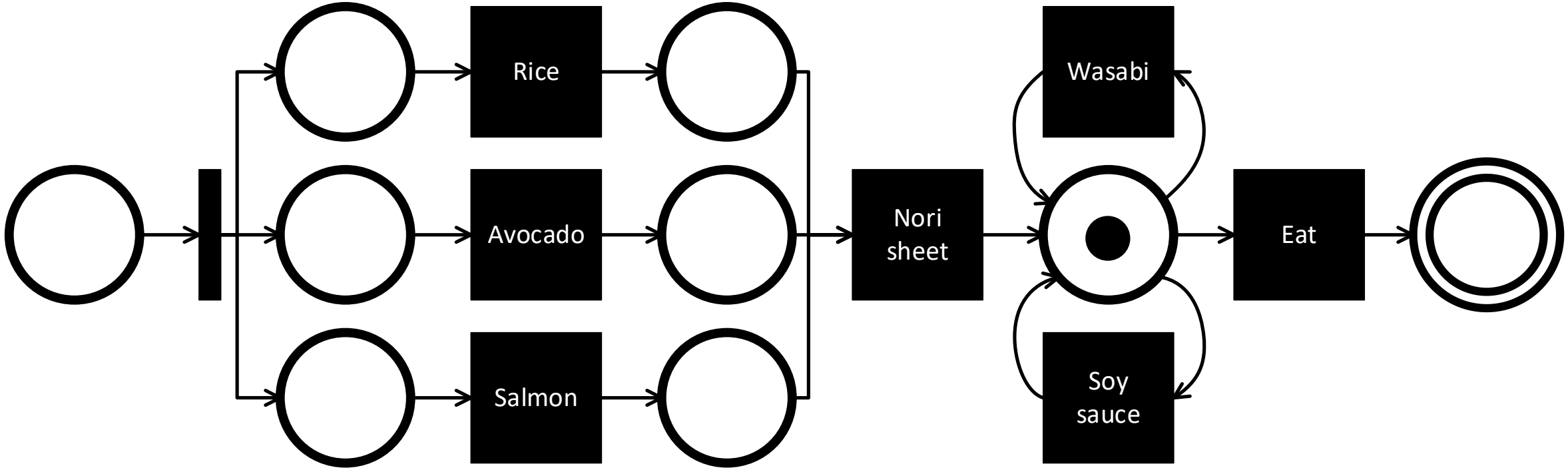
- Checking: <Avocado, Rice, Salmon, Nori, Eat>
- produced (p): 6 consumed (c): 3

Petri Net Membership Test



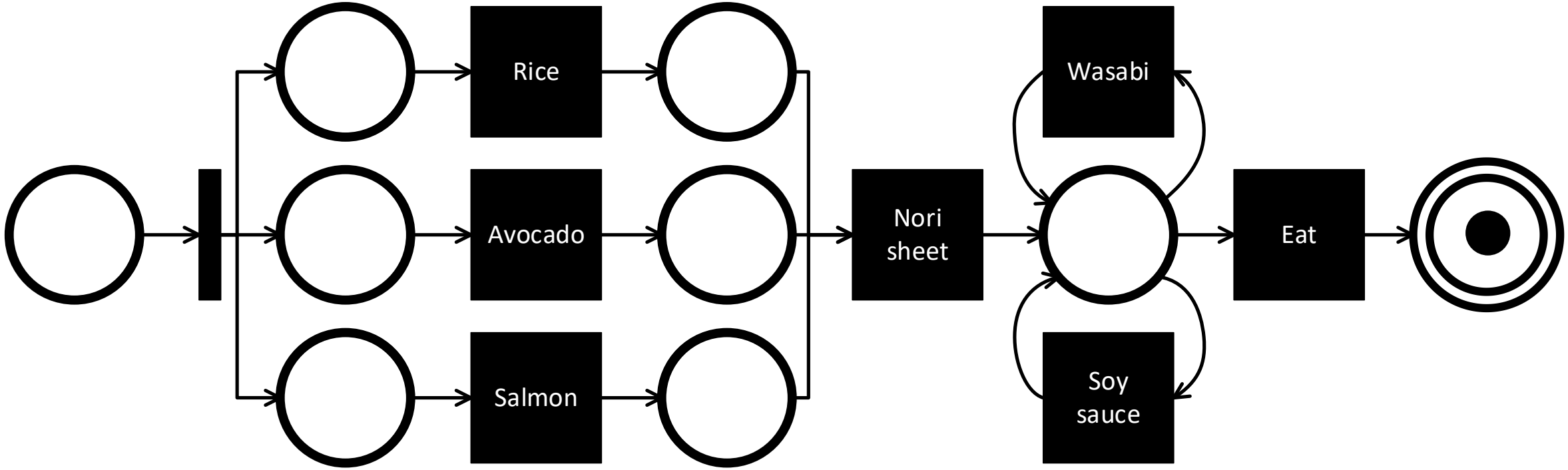
- Checking: <Avocado, Rice, Salmon, Nori, Eat>
- produced (p): 7 consumed (c): 4

Petri Net Membership Test



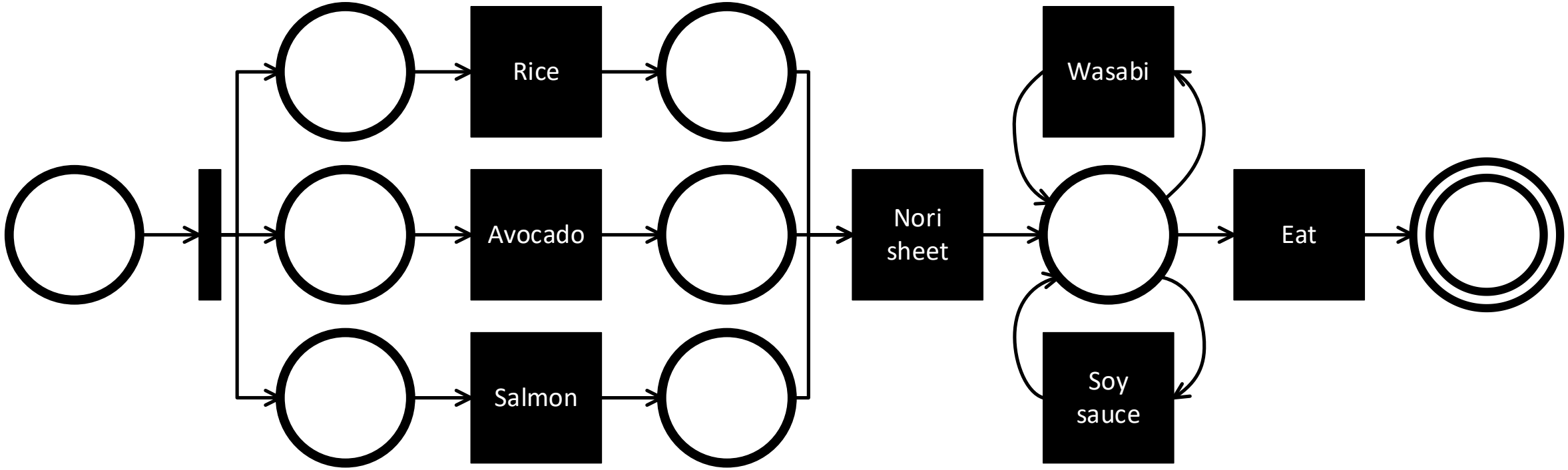
- Checking: <Avocado, Rice, Salmon, Nori, Eat>
- produced (p): 8 consumed (c): 7

Petri Net Membership Test



- Checking: <Avocado, Rice, Salmon, Nori, Eat>
- produced (p): 9 consumed (c): 8

Petri Net Membership Test



- Checking: <Avocado, Rice, Salmon, Nori, Eat>
- produced (p): 9 consumed (c): 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

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

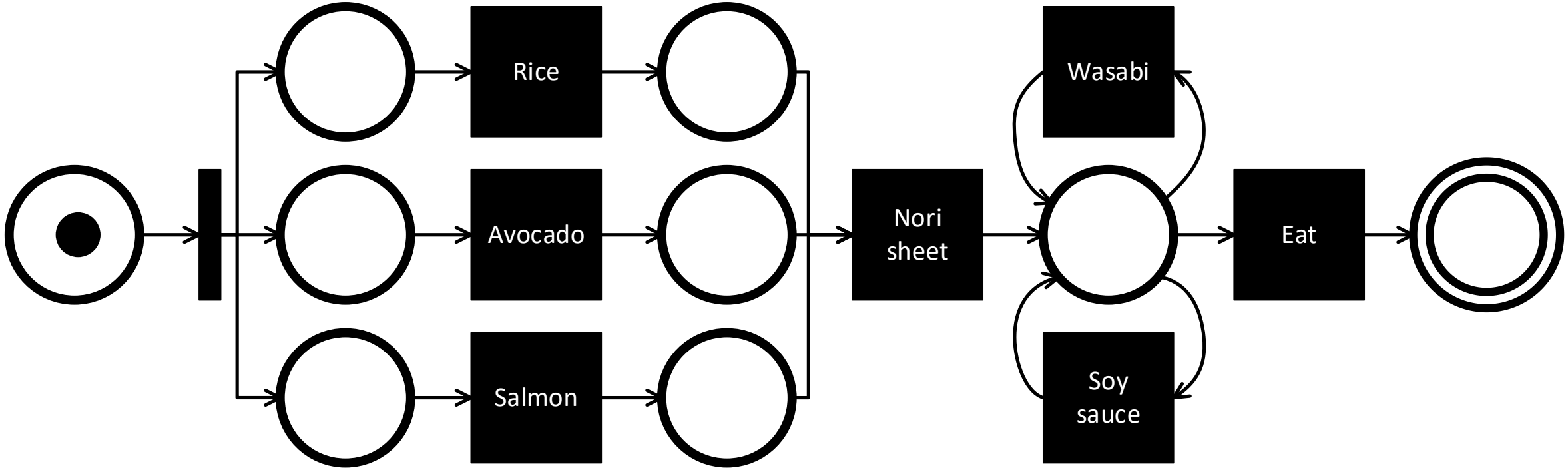
Token Replay[1]

- Problem with pure Automata approach:
  - Binary classifier: „almost fitting“ and „critically deviating“ is not distinguished
  - In practical applications often some flexibility to execute processes is needed

- Modification
  - 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

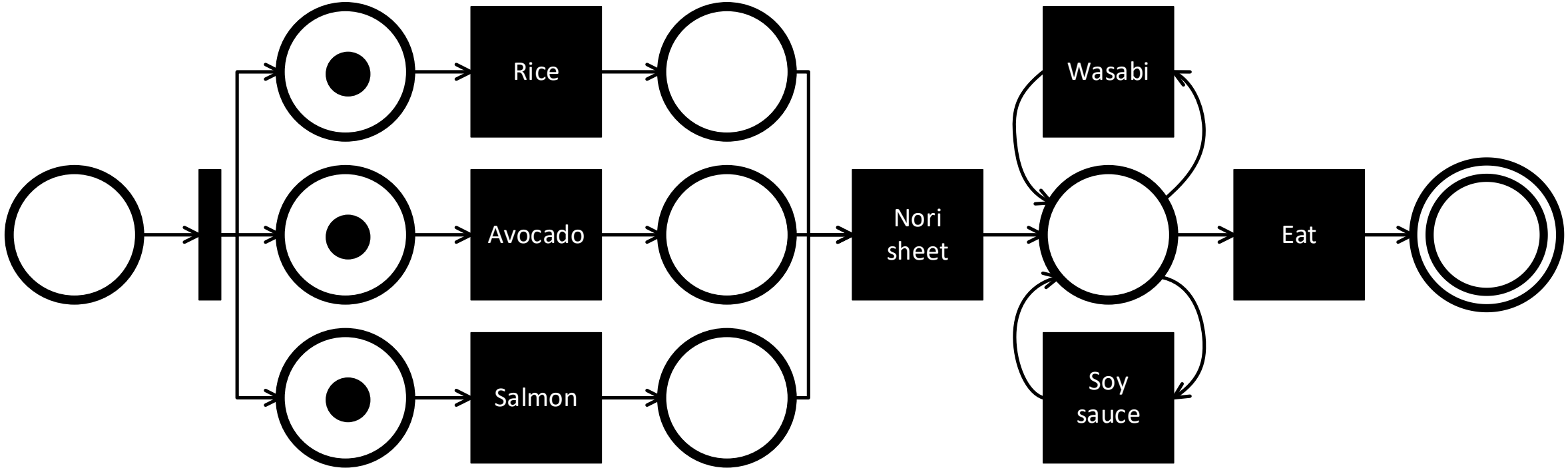


Checking: <Rice, Salmon, Wasabi>

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

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

Token Replay Example

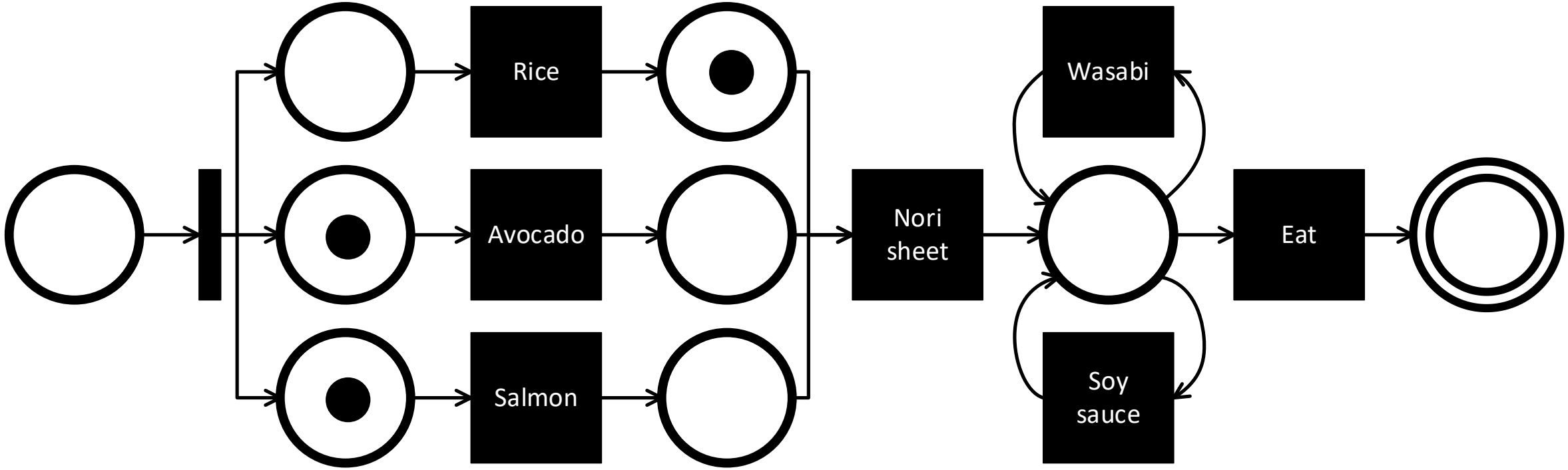


Checking: <Rice, Salmon, Wasabi>

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

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

Token Replay Example

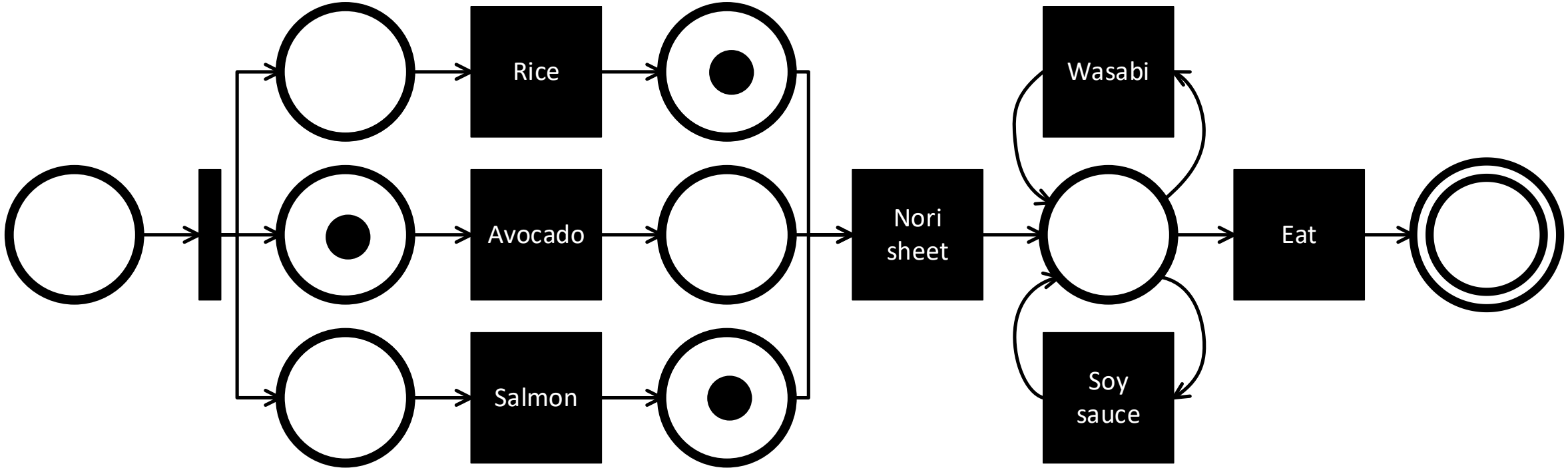


Checking: <Rice, Salmon, Wasabi>

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

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

Token Replay Example

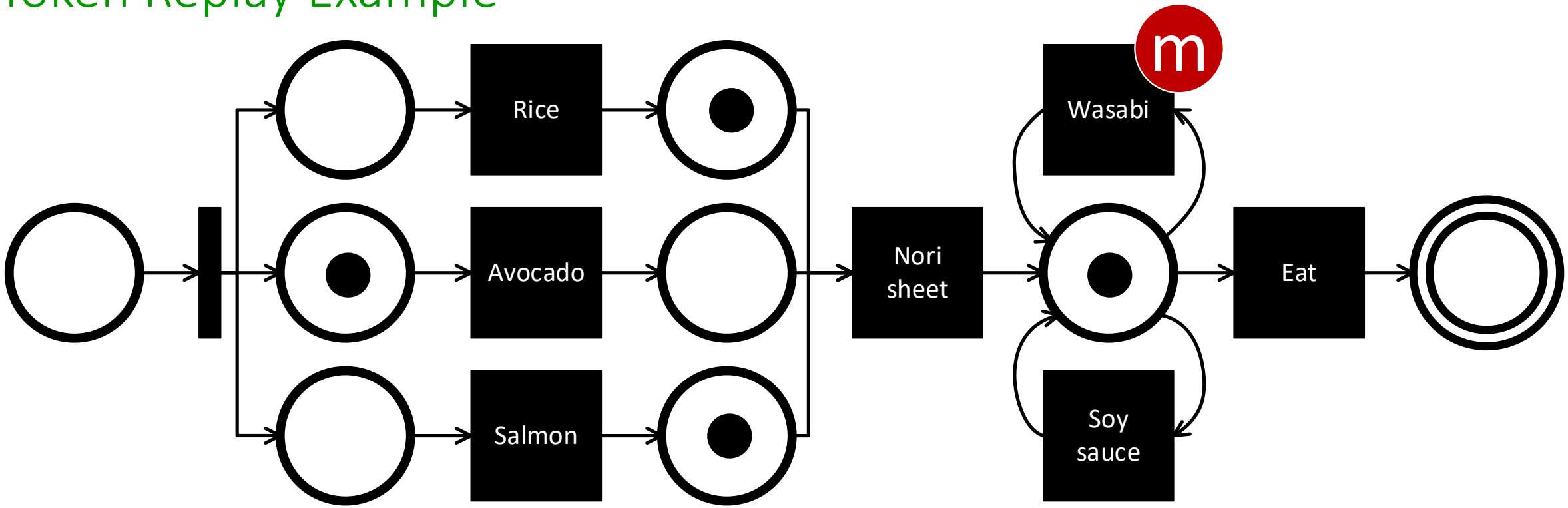


Checking: <Rice, Salmon, Wasabi>

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

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

Token Replay Example

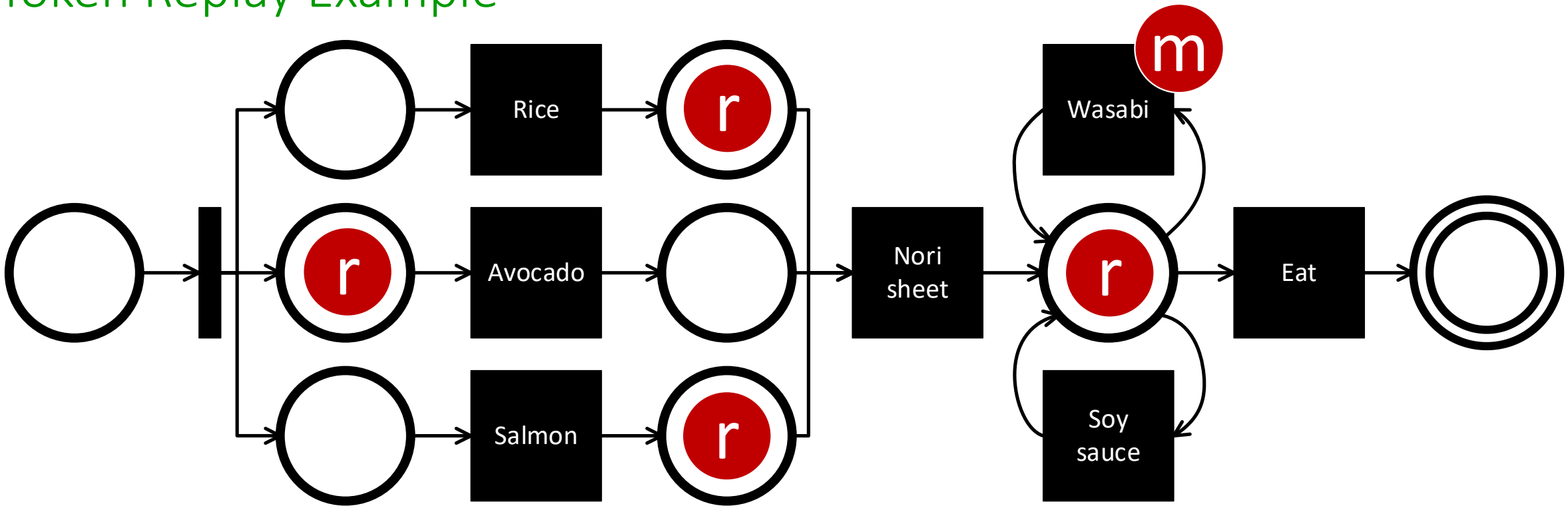


Checking: <Rice, Salmon, Wasabi>

(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: σ = <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.
  $\rightarrow$ 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.

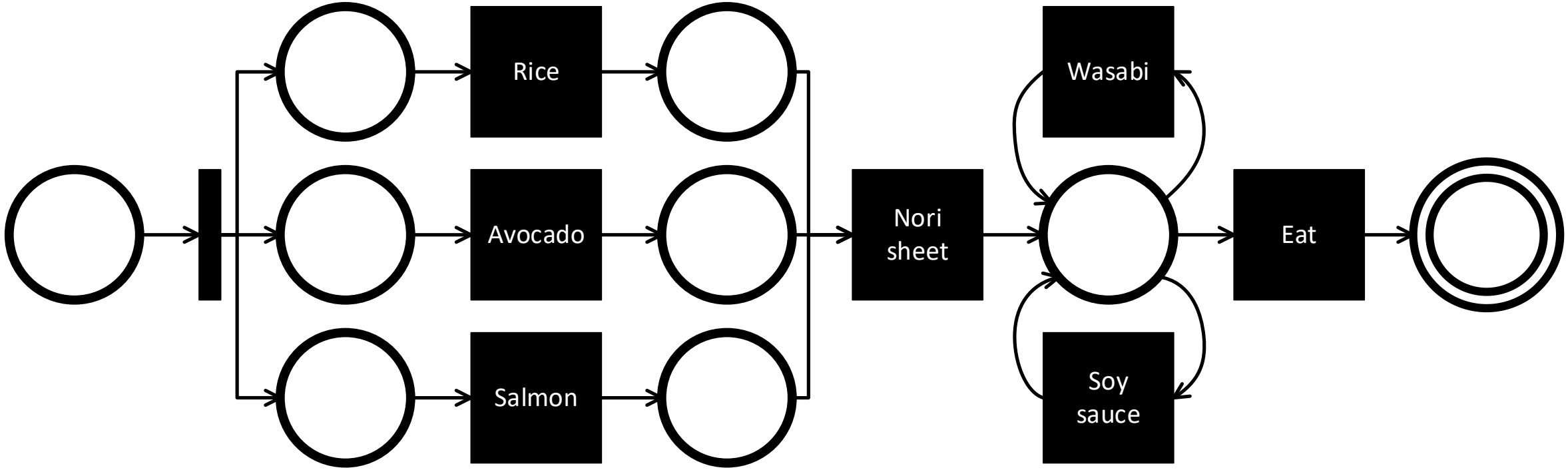
- Idea:
  - Consider all mappings between a model and a trace.
  - Simulate moves in the model and in the trace.
  - Optimize for most synchronuous moves
    (fire transition a and read a in the trace in parallel).
  - Finally, compare the optimal alignment with the worst alignment possible to determine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

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

>> indicates asynchronous moves

[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.

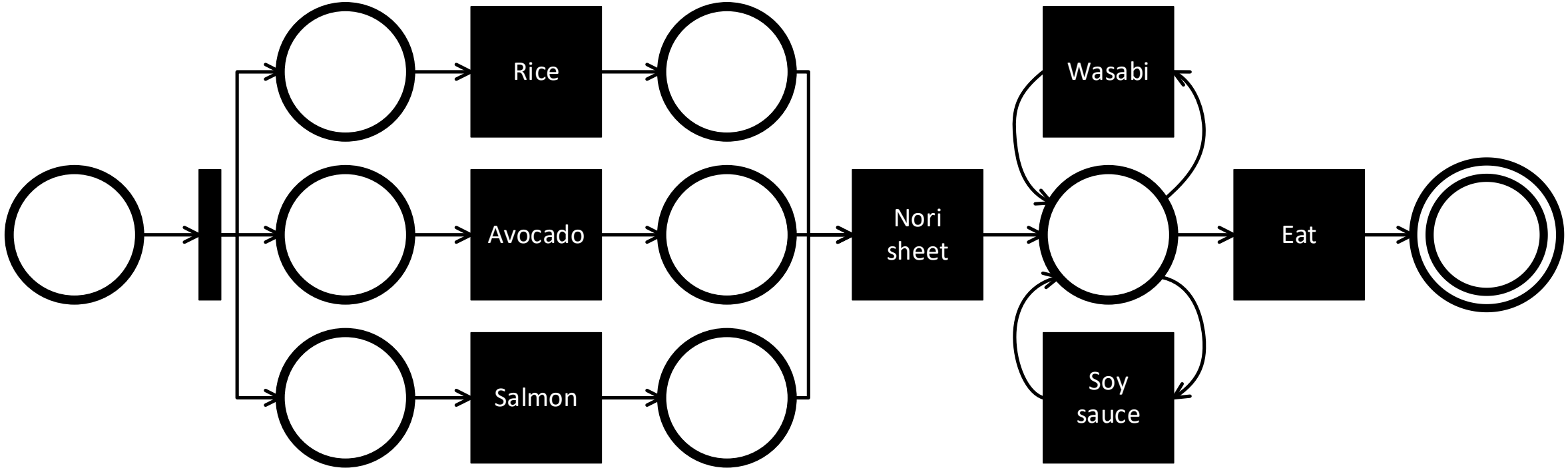
Alignments



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

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

Alignments



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

Rice	>>	Salmon	Salmon	Wasabi	>>
Rice	Avocado	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 is equal for all optimal alignments.

Alignments

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

$\lambda^M_{opt}(\sigma) =$

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

$$\delta\left(\lambda^M_{opt}(\sigma)\right) = 3$$

- Worst alignment:

$\lambda^M_{worst}(\sigma) =$

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

$$\delta\left(\lambda^M_{worst}(\sigma)\right) = 8$$

- The fitness is defined as

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

Alignments Discussion

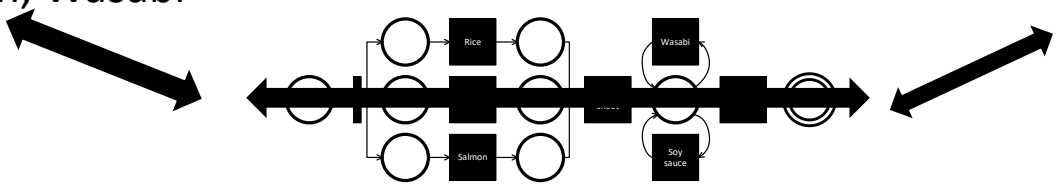
- Alignments are more intuitive
  - skipped and inserted events, rather than tokens as in Petri-nets
- Higher accuracy
  - Token Replay suffers from token flooding
- Fitness values for Alignments tends to be to low
  - Token Replay often yields higher values
- More flexibility due to modifications of the cost δ
  - E.g., activity "avocado" might be cheaper to drop than dropping the activity "rice"
- Not depending on Petri-nets only
- However, computationally very 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:

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

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

1. Introduction
2. Basics
3. Supervised Methods
4. Unsupervised Methods
5. Process Mining
   - Introduction
   - Process Models
   - Log Files
   - Process Discovery
   - Conformance Checking