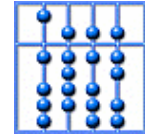


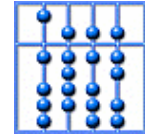
Specification of Distributed Systems

Dr. Bernhard Schätz
Leopold-Franzens Universität Innsbruck
Sommersemester 2005



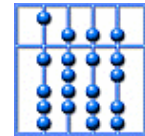
Overview

1. Introduction
2. Basics: Behavior, Interaction, Concurrency
3. Coroutines
4. Communicating Processes
5. Data Flow Models
6. Coordination
7. Executions
8. State-Based Models
9. Property Descriptions

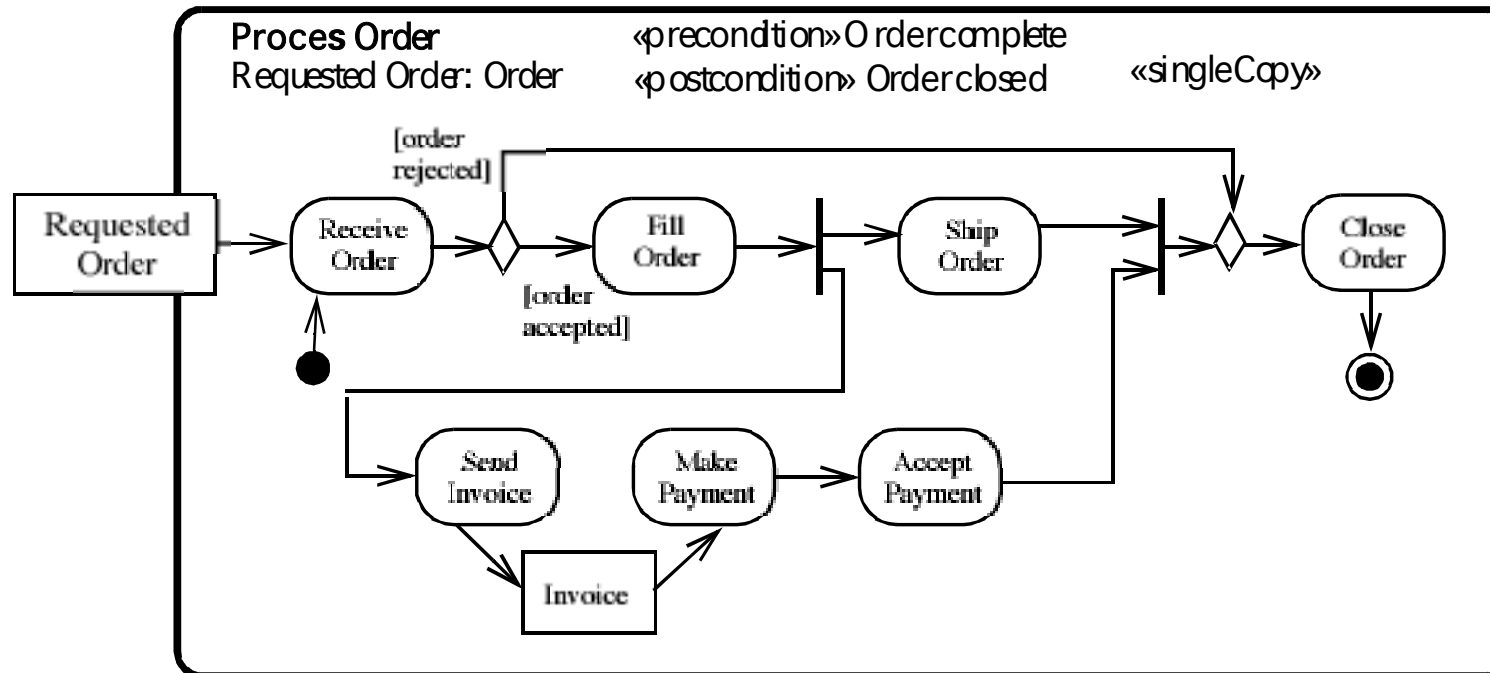


Overview

1. Introduction
2. Basics: Behavior, Interaction, Concurrency
3. Coroutines
4. Communicating Processes
5. Data Flow Models
6. Coordination
 1. Simple Coordination
 2. Data Flow Extensions
7. Executions
8. State-Based Models
9. Property Descriptions



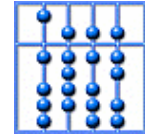
Motivation: Describing Flow of Activation



Source: Unified Modeling Language:
Superstructure version 2.0, OMG, 2003.

Flow of Activation:

- Splitting into independent execution paths
- Joining of independent execution paths



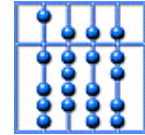
2.3 Modeling Coordination: Condition Event Nets

Goal: Define a notation to describe the coordination of synchronized activities

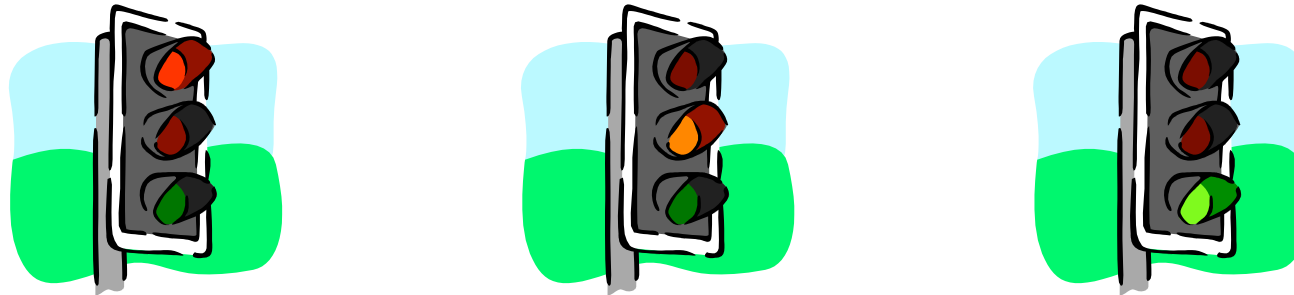
- Relevant: Address aspects evolving with scheduling
- General: Cover aspects independent of specific model of synchronization
- Abstract: Ignore aspects like relative duration of execution or synchronization

Concept: Condition, event, flow

Notation: Condition event nets



Concept: Condition

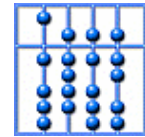


Purpose: Describe *observations about the system of a system*

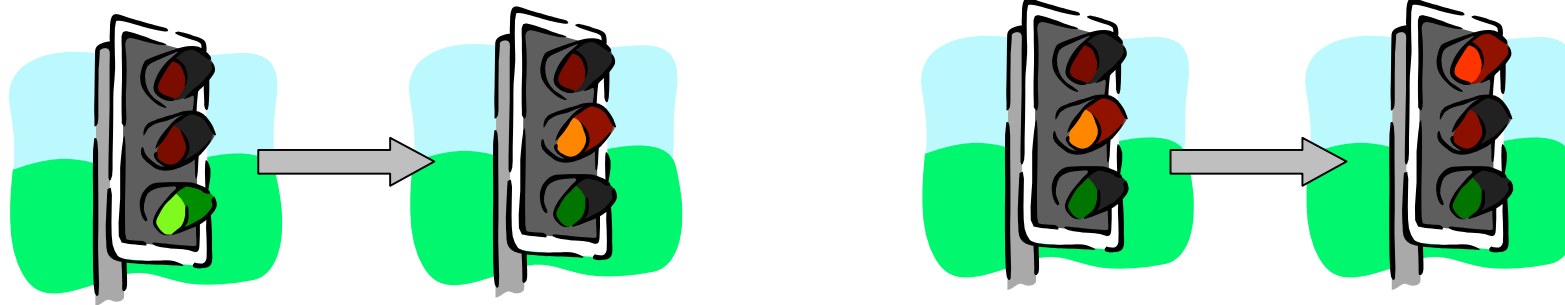
Concept: Condition = Observations about the state of the system influencing the flow of events

- Observable: Condition is either true or false
- Independent: Condition does not contain independent sub-conditions

Example: “Traffic light is red”, “Traffic light is yellow”, “Pedestrian light is green”



Concept: Action

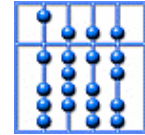


Purpose: Describe *changes in the state of the system*

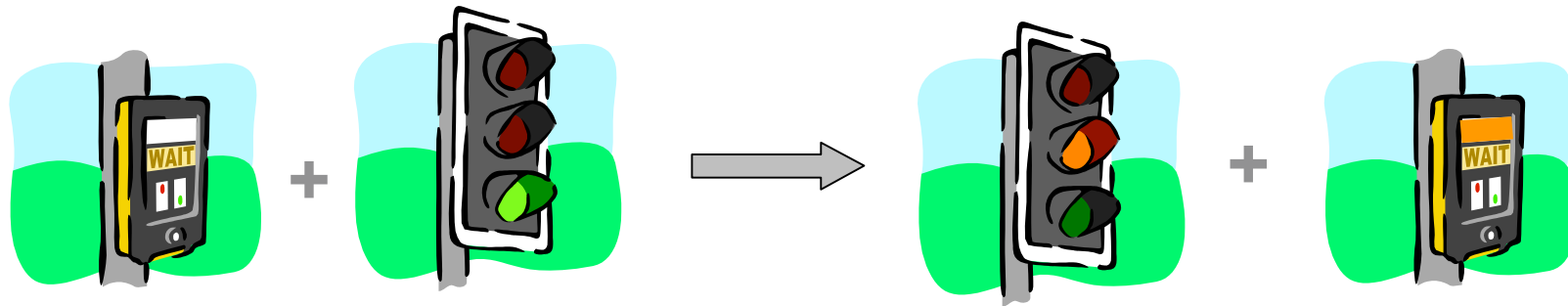
Concept: Action

- Observable: Changes observations about the system
- Atomic: Changed observations occur instantaneously and simultaneously

Example: “Traffic light changes from green to yellow”, “Traffic light changes from yellow to red”



Concept: Per/Post Conditions

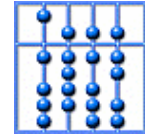


Purpose: Describe *situations holding prior to or after an event*

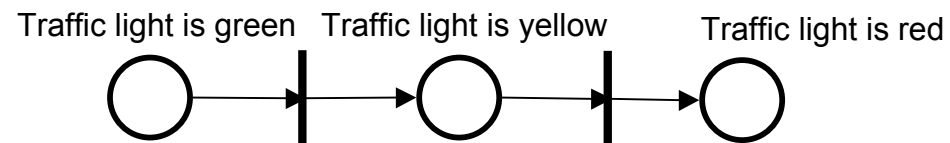
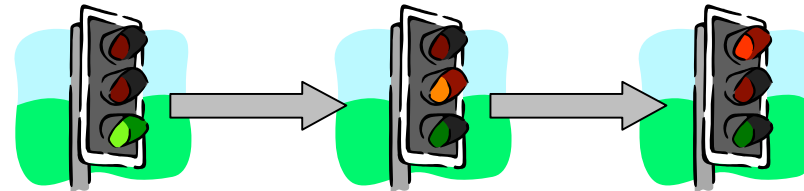
Concept: Condition = Observations about the state of the system influencing the flow of events

- Pre Conditions: Conditions that must be fulfilled prior to the occurrence of an event
- Post Conditions: Condition that are fulfilled after the occurrence of an event

Example: Pre-condition: “Traffic light is green”, “Button has been pressed”; Post-condition: “Traffic light is yellow”



Notation: Place

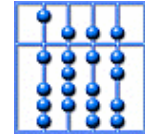


Purpose: Describe *validity of a condition*

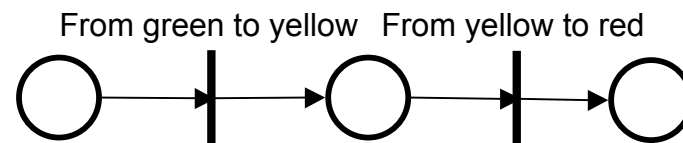
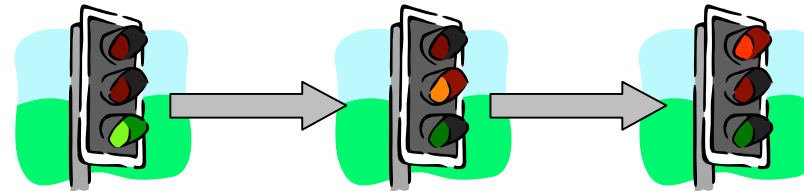
Notation: Place set $P = \{p_1, \dots, p_m\}$

- Concept: Place corresponds to a condition
- Interpretation: State of the place (marked/unmarked) describes validity of condition

Example: $P = \{ \text{“Traffic light is green”}, \text{“Traffic light is yellow”}, \text{“Traffic light is red”} \}$



Notation: Transition



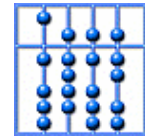
Purpose: Describing *change of observable conditions*

Notation: Transition set $T = \{ t_1, \dots, t_n \}$

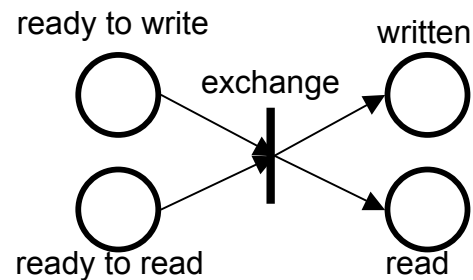
Concept: Transition corresponds to an action of the system

Interpretation: Firing of transition corresponds to the occurrence of an action

Example: $T = \{ \text{"From green to yellow"}, \text{"From yellow to red"} \}$



Notation: Flow

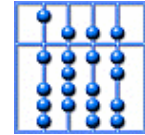


Purpose: Describing *pre- and post-conditions of a transition*

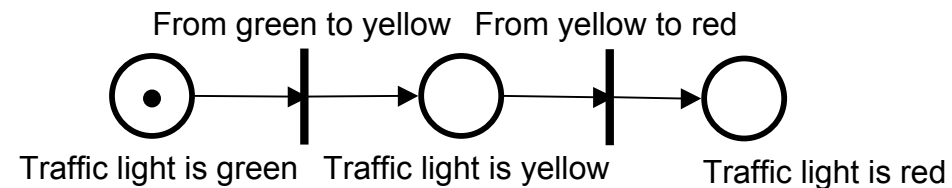
Notation: Flow $F \subseteq P \times T \cup T \times P =$ Connections between places and transitions

- Concept: Conditions checked and changed by an action
- Interpretation:
 - State-transition: Validity required to enable firing of transition, non-validity ensured by firing
 - Transition-state: Non-validity required to enable firing of transition, validity ensured by firing of transition

Example: $T = \{(\text{"ready to write"}, \text{"exchange"}), (\text{"exchange"}, \text{"written"}), (\text{"ready to read"}, \text{"exchange"}), (\text{"exchange"}, \text{"read"})\}$



Notation: Mark

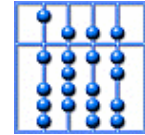


Purpose: Describing a *currently valid condition* of a system

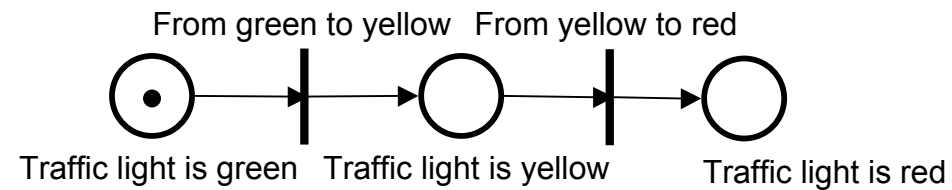
Notation: Mark $m \in P$

- Concept: Validity of an observation about system/fulfilled condition at current state of system
- Interpretation: Place identified by mark corresponds to condition over the current state that currently holds

Example: $P = \{ \text{"Traffic light is green"}, \text{"Traffic light is yellow"}, \text{"Traffic light is red"} \}$,
 $m = \text{"Traffic light is green"}$



Notation: Marking

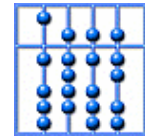


Purpose: Describing *currently valid conditions* of a system

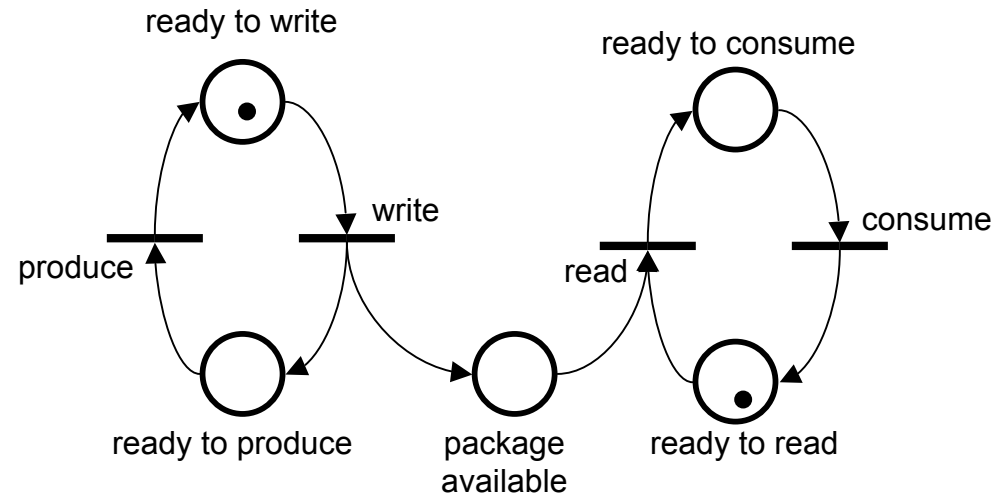
Notation: Marking $M \subseteq P$

- Concept: Validity of observations about system/fulfilled conditions at current state of system
- Interpretation: Marked places correspond to conditions over the current state that currently hold

Example: $P = \{ \text{"Traffic light is green"}, \text{"Traffic light is yellow"}, \text{"Traffic light is red"} \}$,
 $M = \{ \text{"Traffic light is green"} \}$



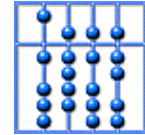
Notation: Condition Event Net



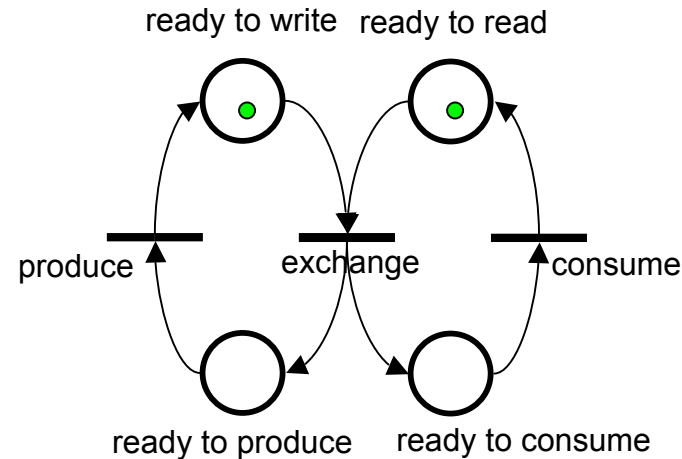
Purpose: Describing dependency in the order of occurrence of events

Notation: Condition Event Net $N = (P, T, F, M)$ with

- P: Places of N
- T: Transitions of N
- F: Flow relation of N
- M: Initial marking of N



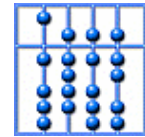
Interpretation: Enabled Transition



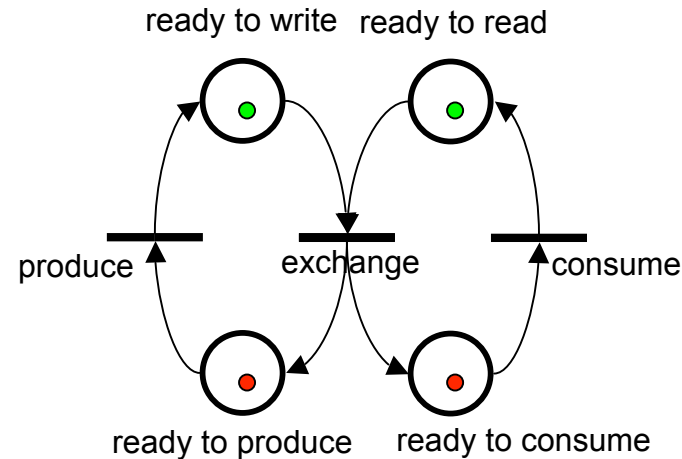
Purpose: Describing *possibility of an atomic change* of a system

Concept: Transition $t \in T$ *enabled by marking* M

- Valid pre-domain:
 - All conditions required by t hold
 - $\{ p \mid (p,t) \in F \} \subseteq M$
- Invalid post-domain:
 - No conditions assured by t hold
 - $\{ p \mid (t,p) \in F \} \cap M = \emptyset$



Interpretation: Firing of Transition

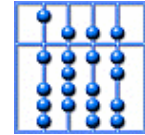


Purpose: Describing *occurrence of an atomic change* of a system

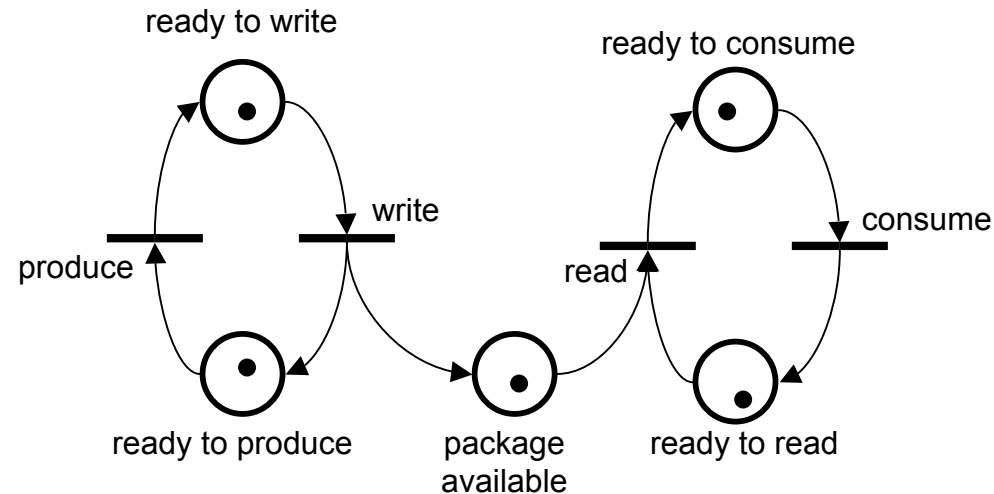
Concept: Firing of an enabled transition t in marking M

- Interpretation: Require conditions become invalid, ensure conditions become valid
- Result: New marking $M' = (M \setminus \{ p \mid (p,t) \in F \}) \cup \{ p \mid (t,p) \in F \}$

Example: $M = \{ \text{"ready to write"}, \text{"ready to read"} \}$, "exchange" is enabled,
 $M' = \{ \text{"ready to produce"}, \text{"ready to consume"} \}$



Interpretation: Execution Trace

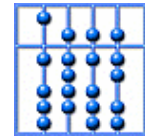


Purpose: Describing a sequence of changes performed by a system

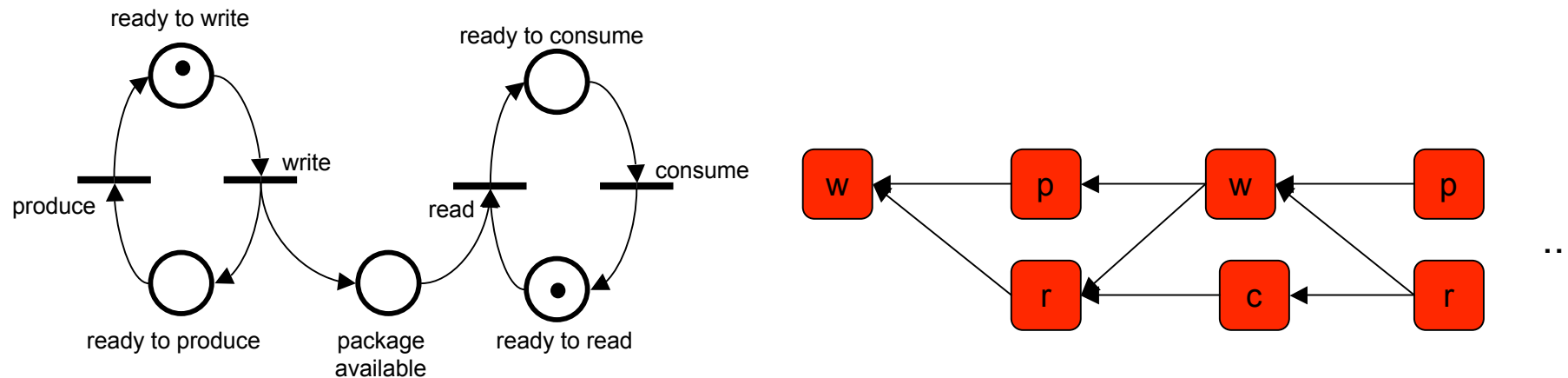
Concept: Execution trace t = Sequence of observations about the system

- Formalization: $t \in (2^P)^\omega$
- Interpretation: Sequence of markings obtained through firings of transitions

Example: {"ready to write", "ready to read"} • {"ready to produce", "package available", "ready to read"} • {"ready to produce", "ready to consume"}



Interpretation: Event Trace

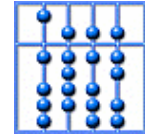


Purpose: Describing a sequence of actions performed by a system

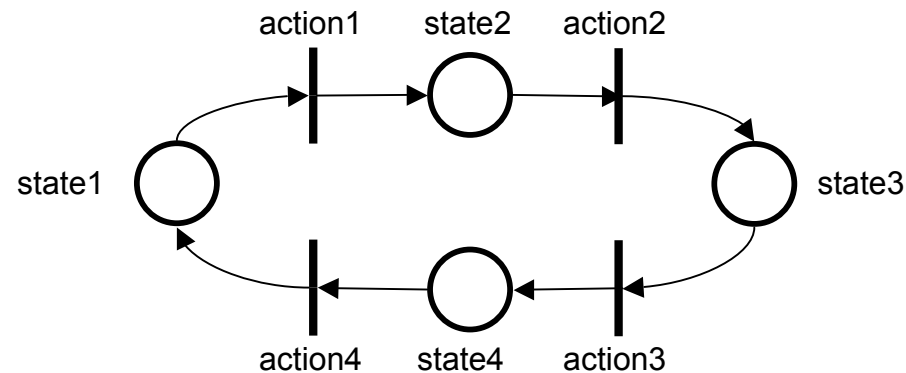
Concept: Event trace t = Sequence of events performed by the system

- Formalization: $t = (E, \leq, L)$ with alphabet T
 - E : Firing of transitions
 - \leq : Causal order induced by generated markings
 - L : Map of firing events to transitions
- Interpretation: Sequence of transitions fired during execution of the system

Example: $E = \cup_{0 < i} \{w_i, p_i, r_i, c_i\} \cup \{w_0\}$,
 $\leq = (\cup_{0 \leq i} \{(w_i, p_{i+1}), (w_i, r_{i+1}), (p_i, w_{i+1}), (r_i, w_{i+1}), (r_i, c_{i+1}), (c_i, r_{i+1})\})^*$,
 $L = \{ w_i \rightarrow \text{"write"}, p_i \rightarrow \text{"produce"}, r_i \rightarrow \text{"read"}, c_i \rightarrow \text{"consume"} \}$



Pattern: Sequential Execution

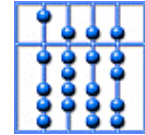


Purpose: Describing execution of a sequential process

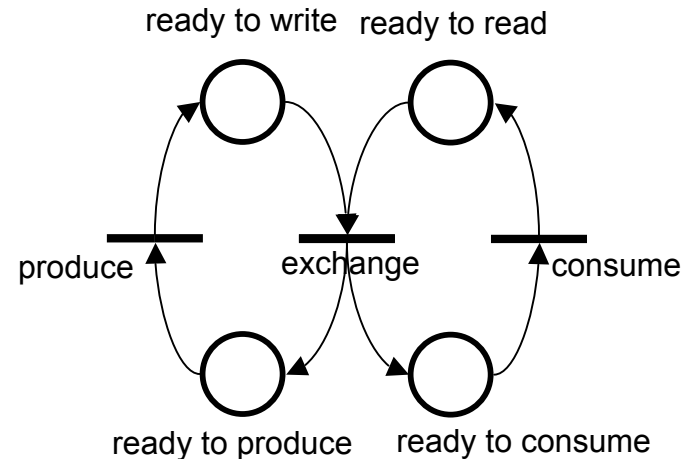
Pattern: Sequential execution = Simple flow graph without forks/joins

- State: Place corresponding to observation “Program is at state s ”
- (Inter-)Action: Transition corresponding to “Change from state s to s ”
- Recursion: Loop in the flow graph

Example: Producer: $P = \{ \text{“ready to write”, “ready to produce”} \}$, $T = \{ \text{“produce”, “write”} \}$



Pattern: Synchronization

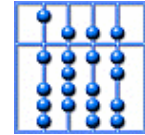


Purpose: Describing synchronization of concurrent processes

Pattern: Synchronization = Shared transition of sequential processes

- Pre-transition places: States of processes waiting for synchronization
- Transition: Interaction of processes
- Post-transition places: States of processes after synchronization

Example: Producer/consumer with shared transition “exchange”

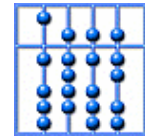


2.4 Modeling Coordination: Extended Notations

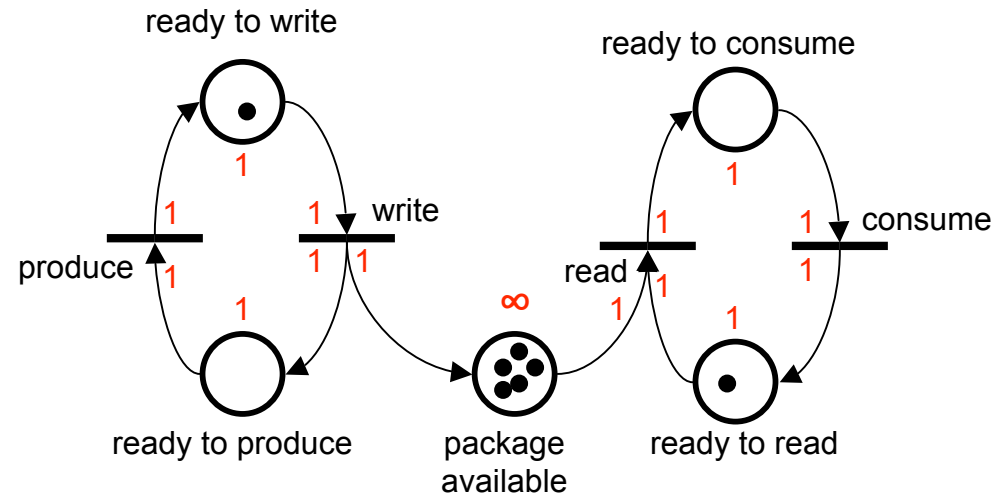
Goal: Define a notation to describe the coordination of complex synchronized activities

- Relevant: Address aspects evolving with scheduling of data-dependent activities
- General: Cover aspects independent of specific model of synchronization
- Abstract: Ignore aspects like relative duration of execution or synchronization

Notation: Weighted Nets, Predicate/Transition nets



Extension: Place/Transition Nets

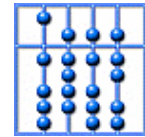


Purpose: Describing synchronized data flow systems

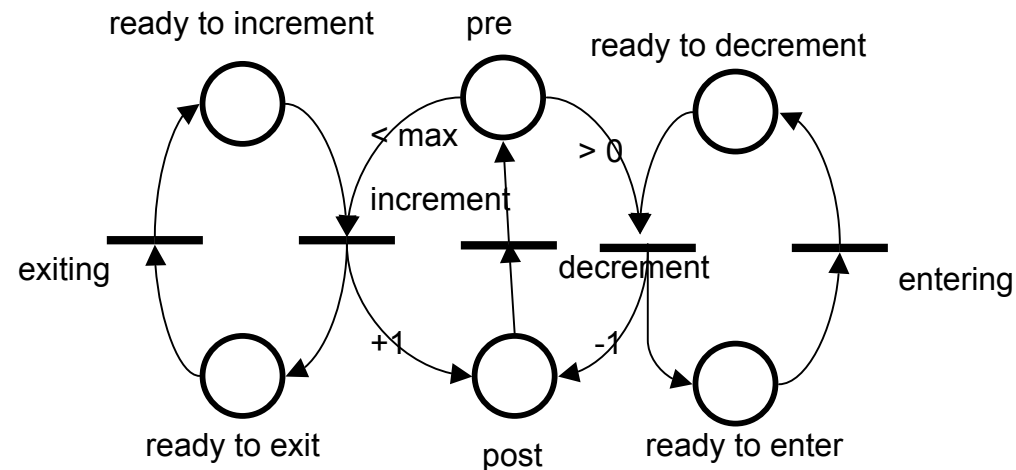
Notation: Weighted P/N Nets:

- Places: Marked with maximum capacity
- Edges: Marked with consumed/produced marks
- Marking: Maps places to current number of contained marks
- Firing: Pre-places hold at least amount of consumed marks, post-places lack at least amount of produced marks, firing removes consumed marks/adds produced marks

Example: Producer/Consumer with unlimited buffer



Extension: Predicate Transition Nets

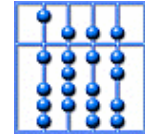


Purpose: Describing complex behavior with compact notation

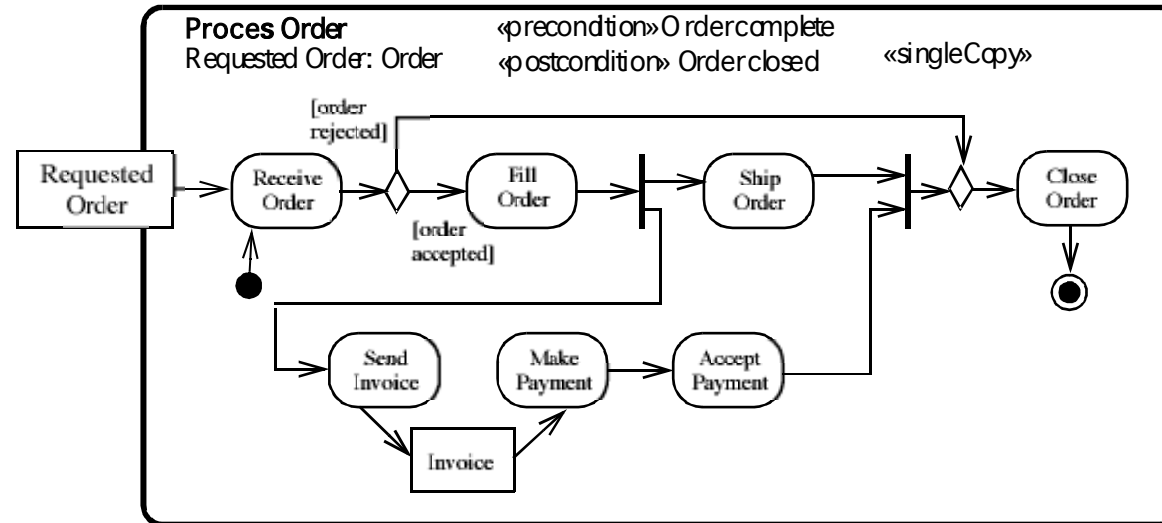
Notation: Predicate/Transition Nets = Net with Individualized Tokens

- Tokens: Assigned with values
- Places: Holding sets of tokens(values)
- In-edges of transitions: Marked with predicates over tokens
- Out-edges of transitions: Marked with functions over tokens

Example: Counter for incrementing/decrementing process



Variant: Activity Diagrams



Purpose: Describing combinations of data flow and coordination

Notation: Combination of Condition/Event nets with Predicate/Transition

- Condition/Event net:
 - Describing control flow/coordination
 - Extension: Marked fork/join points
- Predicate/Transition net:
 - Describing data flow/computations