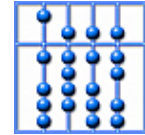


Specification of Distributed Systems

Dr. Bernhard Schätz
Leopold-Franzens Universität Innsbruck
Sommersemester 2005



Overview

1. Introduction

2. Basics: Behavior, Interaction, Concurrency

3. Coroutines

4. Communicating Processes

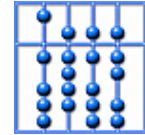
5. Data Flow Models

6. State-Based Models

7. Coordination

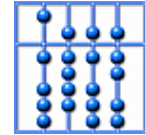
8. Executions

9. Property Descriptions

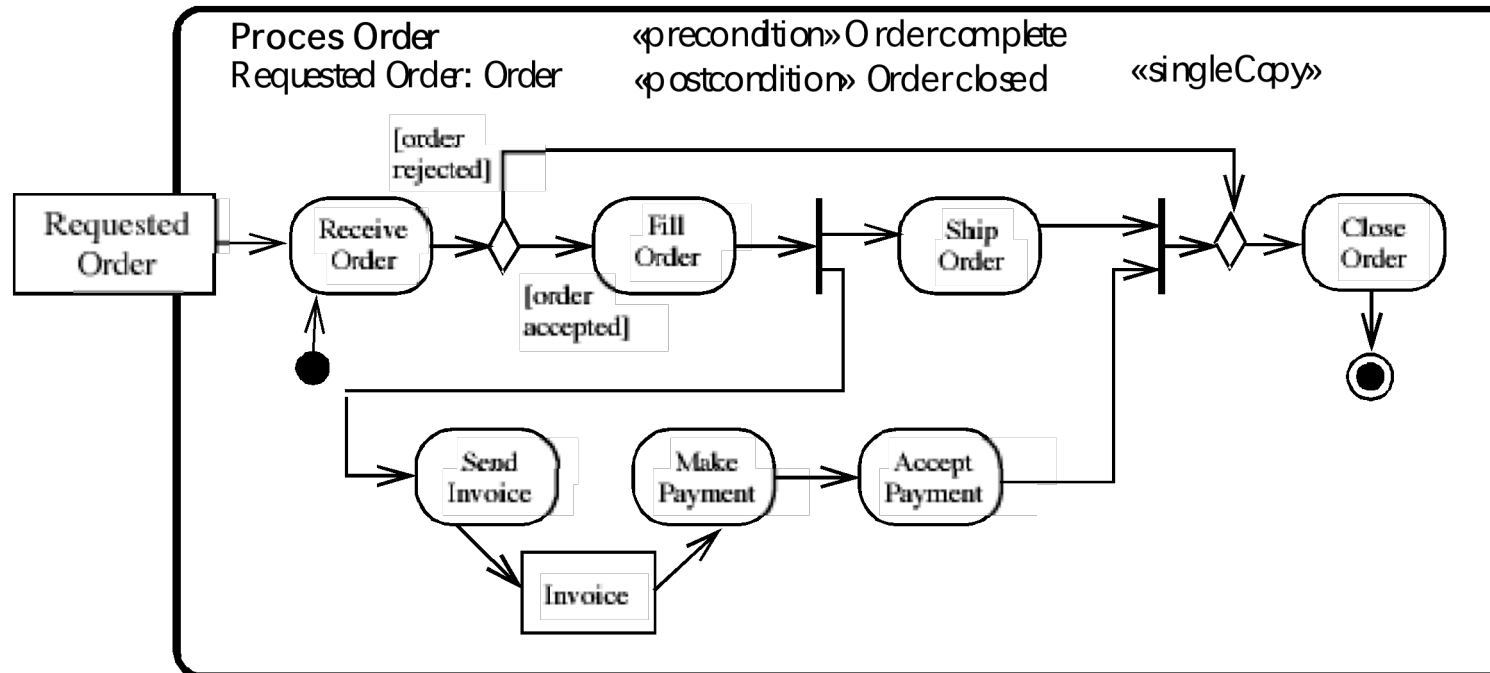


Overview

1. Introduction
2. Basics: Behavior, Interaction, Concurrency
 1. Modeling Computation: State Transition Systems
 2. Modeling Interaction: Labeled Transition Systems
 3. Modeling Concurrency: Synchronized Transition Systems
 4. Modeling Behavior: Streams of Observations
 5. Modeling Communication: Synchronized Behaviors
 6. Modelling Parallelism: Event Structures
3. Coroutines
4. Communicating Processes
5. Data Flow Models
6. State-Based Models
7. Coordination
8. Executions
9. Property Descriptions



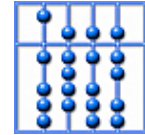
Motivation: Modeling Flow of Activation



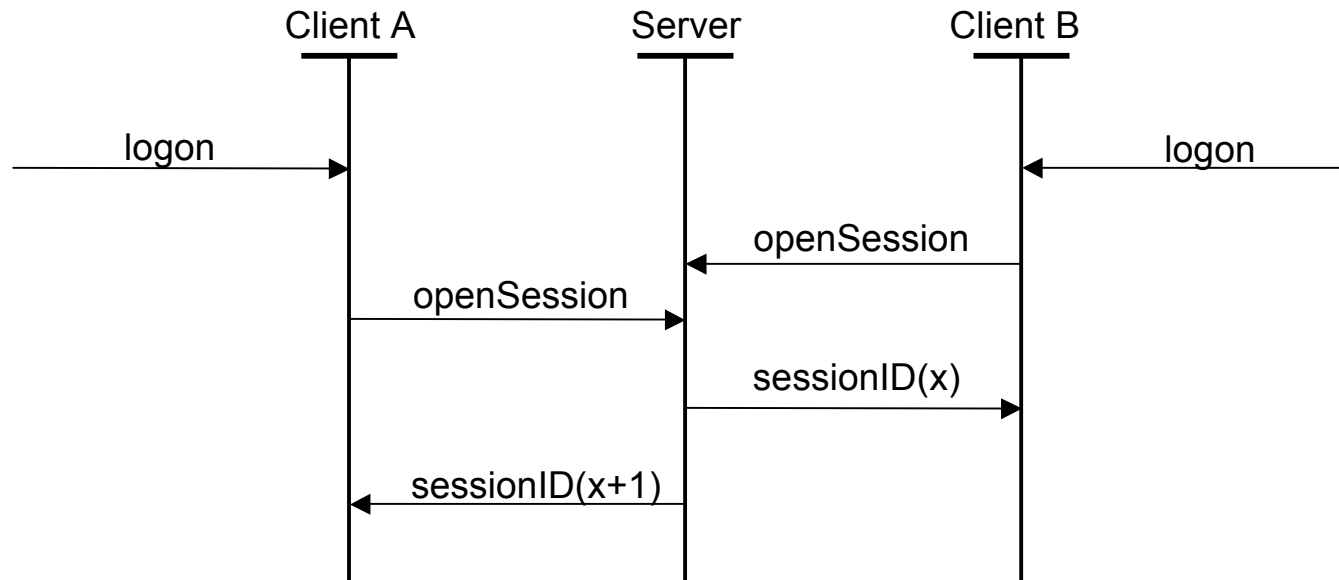
Source: Unified Modeling Language: Superstructure version 2.0, OMG, 2003.

Flow of Activation:

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

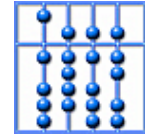


Motivation: Modeling Flow of Interaction

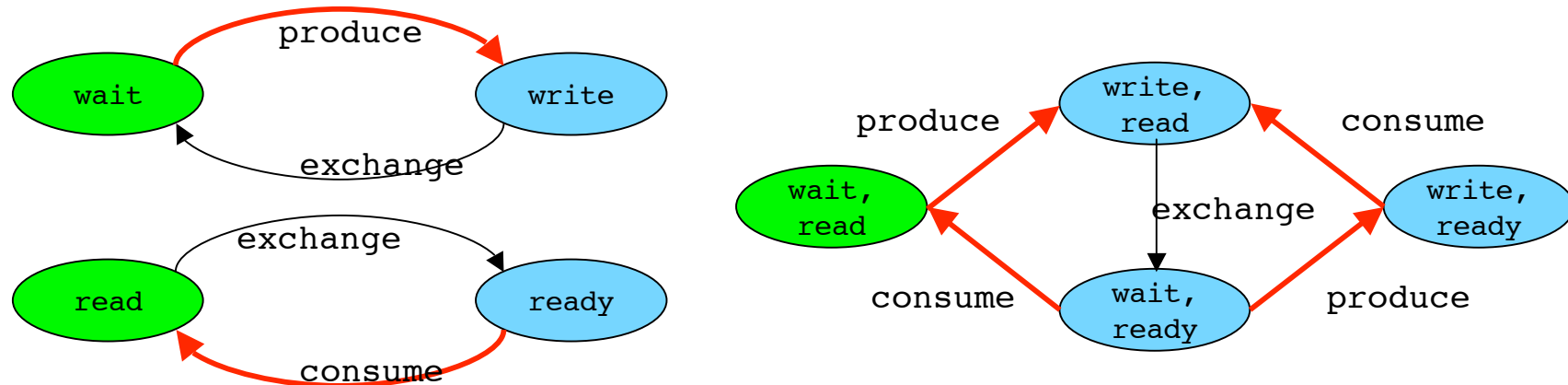


Sequence-Based Description: Timing Abstraction

- Independent signals: Any order
- Dependent signals: Defined order

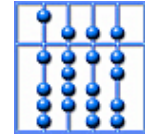


Motivation: Modeling Independence



Abstract model for independent execution of actions:

- Transition System: “Diamond” execution path
- Observation Trace: Arbitrary interleaving



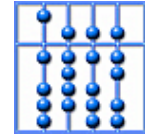
2.3 Modeling Parallelism: Event Structures

Goal: Define a model to describe the behavior of parallel paths of execution

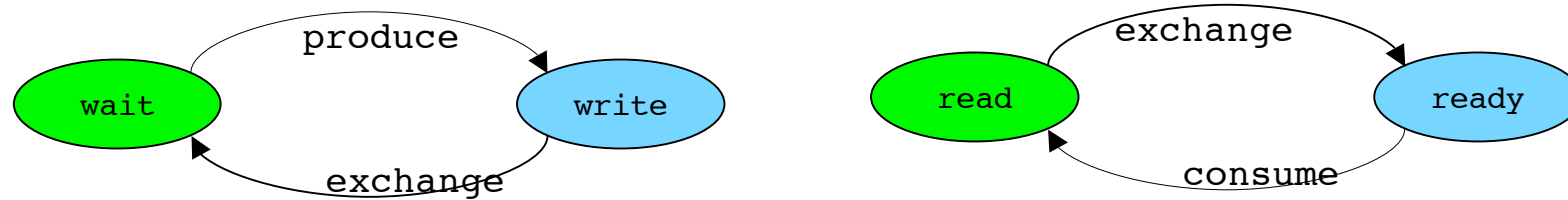
- Relevant: Address aspects evolving with parallelism
- General: Cover aspects independent of specific model of computation
- Abstract: Ignore aspects like relative communication or execution speed, architecture of system

Concept: Action, event, order

Model: Event structures



Concept: Interaction



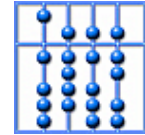
Purpose: Describe synchronizations between system and environment

Concept: Interaction

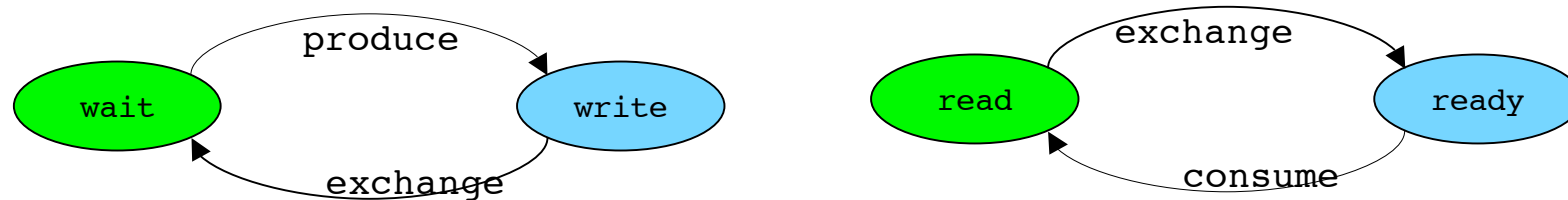
- Observable: Action taking place at interface of a system
- Atomic: Not constructed of sub-interactions

Example: produce, consume

Note: “Interaction” often also called “action” (e.g., Winskel)



Concept: Alphabet

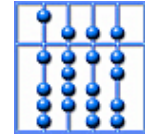


Purpose: Describe complete interaction possibilities

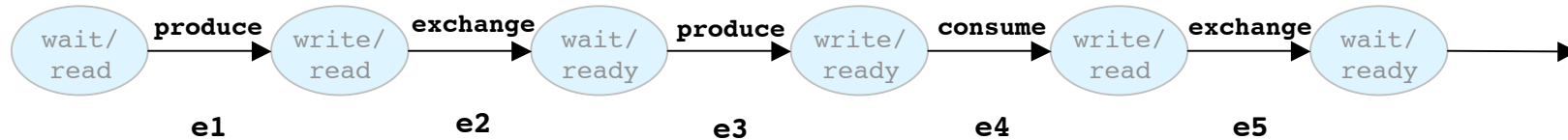
Concept: Alphabet

- Set of all potential interactions
- Potential: Interaction may not be executable by system

Example: { produce, consume, exchange }



Concept: Event

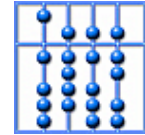


Purpose: Describe *occurrence* of an interaction

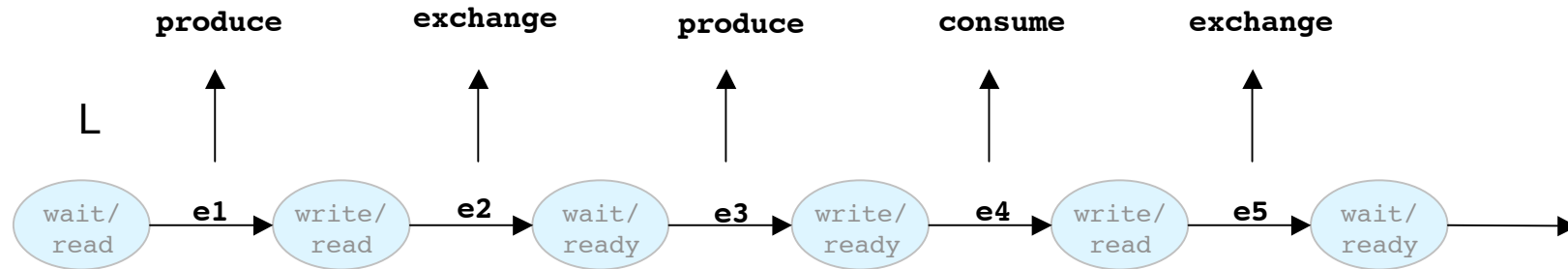
Concept: Events E of a system

- Identifiable: Each occurrence of an interaction marks an event
- Distinguishable: Each event corresponds to the occurrence of a single interaction

Example: “e1” corresponds to occurrence of “produce”, “e2” corresponds to occurrence of “exchange”, “e4” corresponds to occurrence of “consume”



Concept: Labeled Event

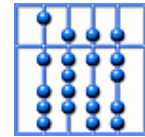


Purpose: Describe *occurrence* of an interaction

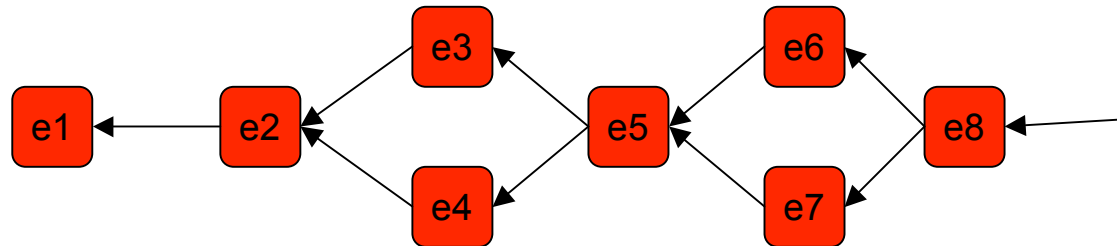
Concept: Labeled event

- Labeling function $L: E \rightarrow A$
- Total: Each event as a corresponding label

Example: $L = \{ e1 \rightarrow \text{produce}, e2 \rightarrow \text{exchange}, e3 \rightarrow \text{produce}, e4 \rightarrow \text{consume}, e5 \rightarrow \text{exchange}, \dots \}$



Concept: Causal Order

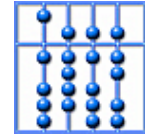


Purpose: Describing dependency in the order of occurrence of events

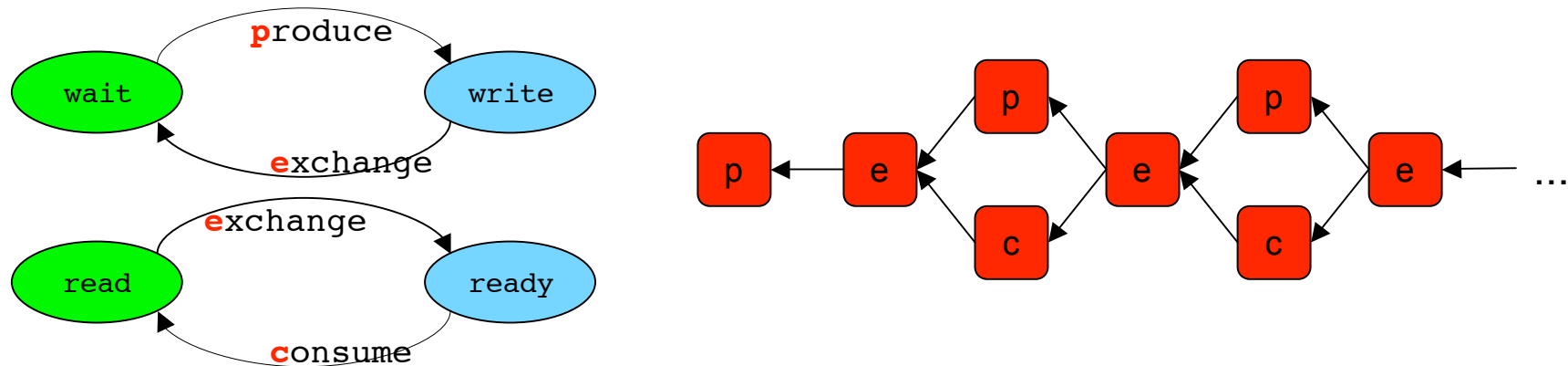
Concept: Causal order $\leq \subseteq E \times E$:

- Precedence order: Events
- Partial order:
 - Reflexive: $e \leq e$
 - Transitive: $e1 \leq e2$ and $e2 \leq e3$ implies $e1 \leq e3$
 - Anti-symmetric: $e1 \leq e2$ and $e2 \leq e1$ implies $e1 = e2$
- Finally founded: Each event is caused by a finite set of events

Example: $\leq = \{ (e1,e2), (e2,e3), (e1,e3), (e2,e4), (e1,e4), (e3, e5), (e2,e5), (e1,e5), \dots \}$



Concept: Event Trace



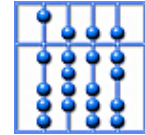
Purpose: Describing finite and infinite order of observations

Concept: Event trace s = Finite or infinite structure of events

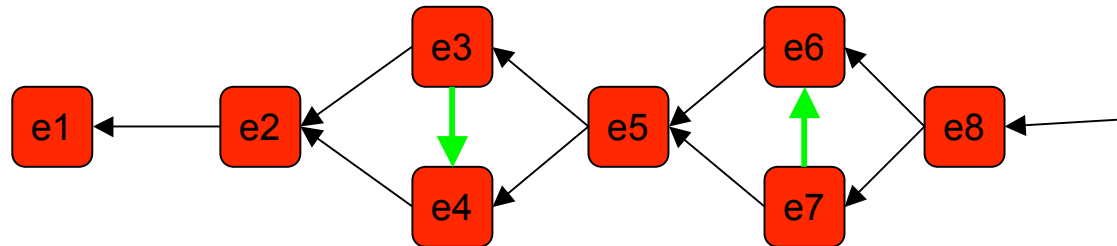
- Ordered: Causal relation as imposed by the system
- Labeled: Alphabet as defined by the interface of the system

Example $s = (E, \leq, L)$ with

- $E = \{e_0, e_1, e_2, e_3, e_4, \dots\}$
- $\leq = \{ (e_0, e_1), (e_1, e_2), (e_1, e_3), (e_2, e_4), (e_3, e_4), \dots \}$
- $L = \{ e_0 \rightarrow \text{produce}, e_1 \rightarrow \text{exchange}, e_2 \rightarrow \text{produce}, e_3 \rightarrow \text{consume}, \dots \}$



Special Case: Sequential Trace

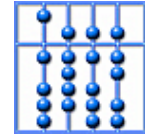


Purpose: Describing dependency in the order of occurrence of events

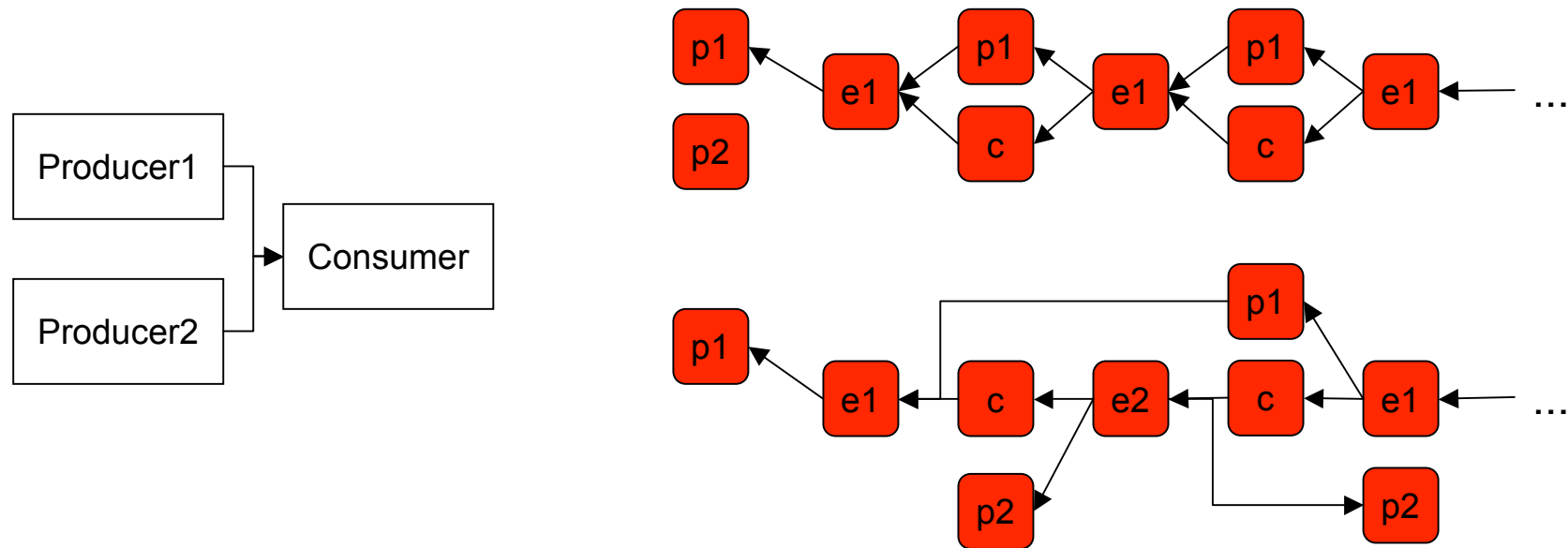
Concept: Sequential trace

- Events: Subset of \mathbb{N}_0
- Causal order: Linear order (“less or equal”)
- Corresponds to: Observation trace

Example: produce • exchange • produce • consume • exchange • consume •
produce • exchange • ...



Concept: Behavior



Purpose: Describing all observable event traces of a system

Concept: Behavior = (A,S)

- Alphabet A: Set of interactions
- Executions S: Set $\{s_1, s_2, \dots\}$ of event traces $s_i = (E_i, \leq_i, L_i)$