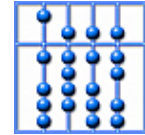


---

## **Vorlesung Specification of Distributed Systems**

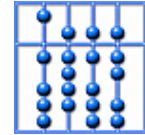
Dr. Bernhard Schätz  
Leopold-Franzens Universität Innsbruck  
Sommersemester 2005



## Overview

---

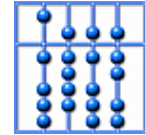
1. Introduction
2. Basics: Behavior, Interaction, Concurrency
3. Coroutines
4. Data Flow Models
5. Communicating Processes
6. State-Based Models
7. Coordination
8. Executions
9. Property Descriptions



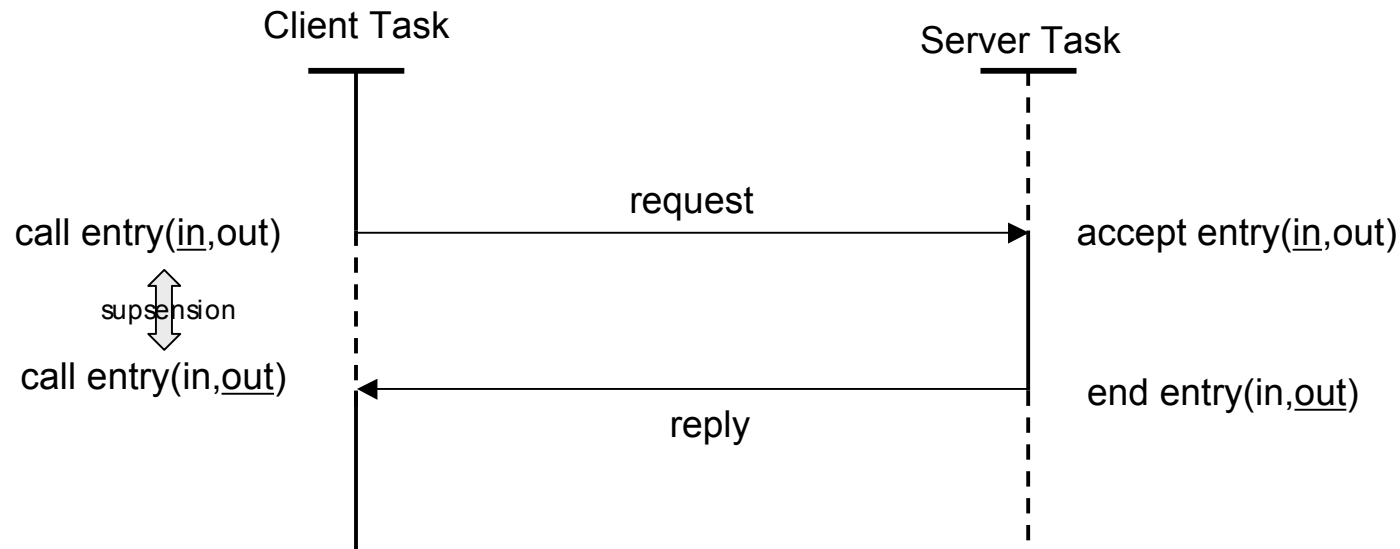
## Overview

---

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

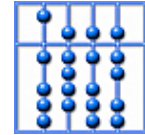


## Motivation: Rendezvous Based Communication



### Rendezvous-Based Communication:

- Synchronous communication between sender and receiver
- Examples: Occam input/output, Ada rendezvous,

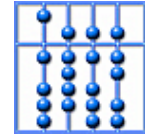


## Recap: Models

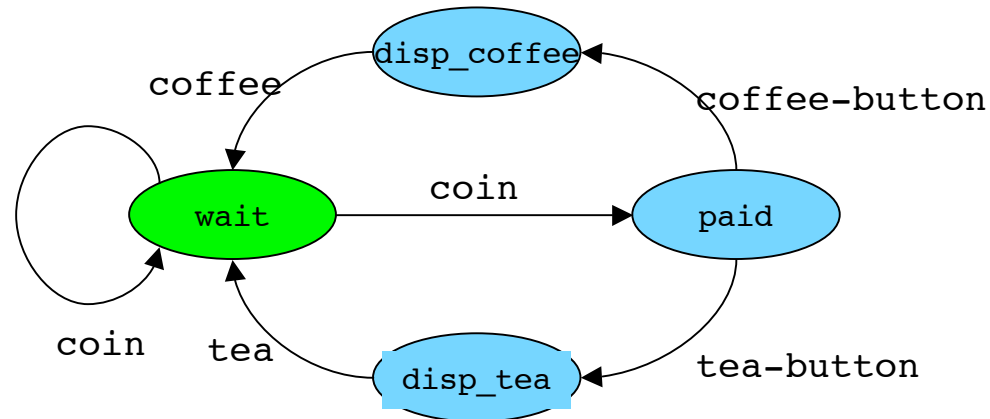
---

### Classes of models:

- **Operational model:** Specification in form of an abstract machine
  - Concepts: State, (labeled) transition, transition system
  - Verification: (Bi-)Simulation
  - Example: (Timed) Automata
- **Denotational model:** Specification in form of observable behavior
  - Concepts: Interface, event, (observation) trace,
  - Verification: Behavior Inclusion
  - Example: Trace-based models
- **Algebraic model:** Specification in form of syntactic formulae
  - Concepts: Process, action, choice,
  - Verification: Term equivalence
  - Example: (T)CSP



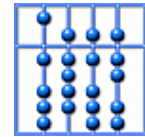
## Recap: Modeling Reactivity



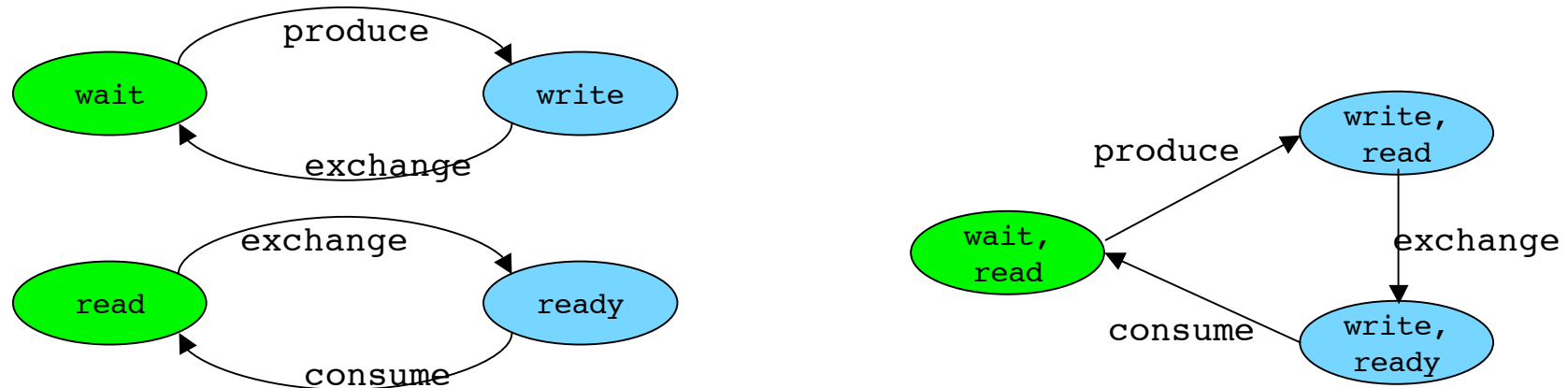
### Concepts:

- Interactions (alphabet): Joint actions/observations of system and environment
- Observation Trace: Sequence of interaction during an execution
- Choice: Alternative behavior offered by a system
- Nondeterminism: Alternative behavior enforced by a system
- Input/Output: Interaction controlled by the environment/system

Model: Labeled Transition System  $(S, A, S_0, T)$



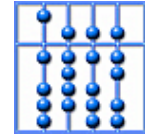
## Recap: Modeling Concurrency



### Concepts:

- Concurrent execution: Synchronized alternative execution
- Independent interaction: Interleaved execution of interactions
- Synchronized Interaction: Simultaneous execution of interactions

### Model: Synchronized Product Automaton



## 5.1 Communicating Processes: Sequential Processes

---

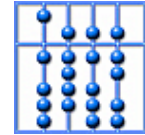
**Goal:** Introduce notation to describe the behavior of rendezvous-based systems

- Introduction of processes as interacting entities
- Definition of sequential behavior
- Support of elements of control flow

**Concept:** Sequential interacting processes

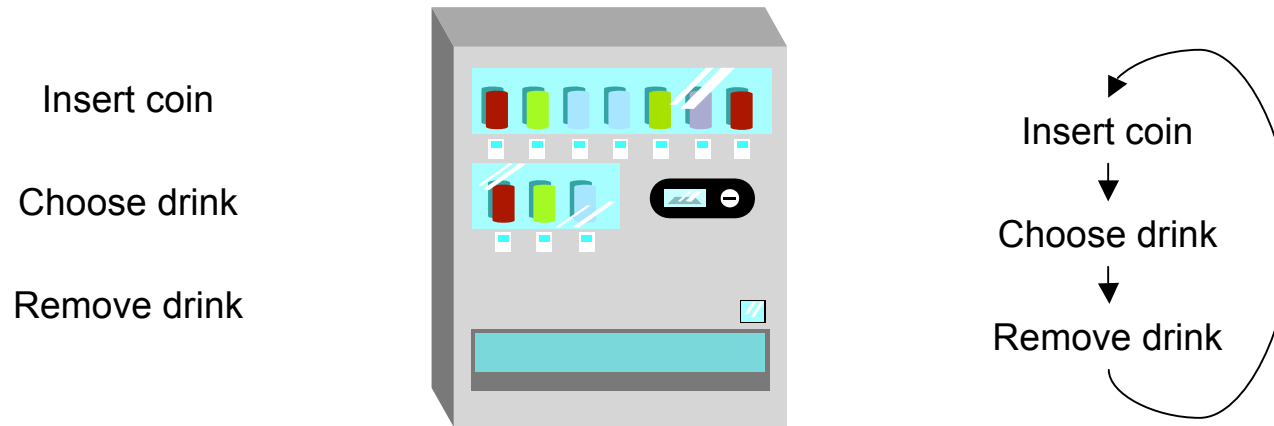
**Notation:** (T)CSP (Communicating Sequential Processes)

**Model:** Labeled Transition Systems



## Concept: Process

---

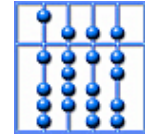


Purpose: Defining abstract “building blocks” of behavior

Process: (Abstract) unit of execution

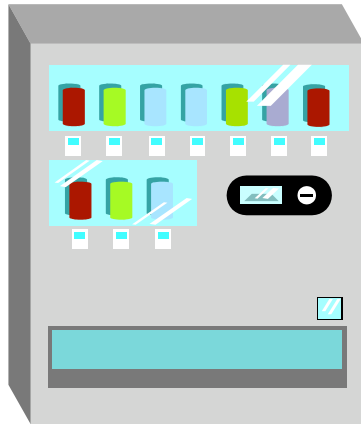
- Interactions: Set of atomic observable joint actions of a process and its environment
- Behavior: Set of executions ordering interactions of a process

Sequential process: Execution of an *interacting* sequential program



## Notation: Algebraic Specification

---



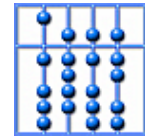
Machine = coin  $\rightarrow$  Wait  
Wait = (coffee-button  $\rightarrow$  Coffee) |  
          (tea-button  $\rightarrow$  Tea)  
Coffee = coffee  $\rightarrow$  Machine  
Tea = tea  $\rightarrow$  Machine

Purpose: Describing processes using a textual notation forming process terms

Process algebra: Calculus to textually define and formally relate processes

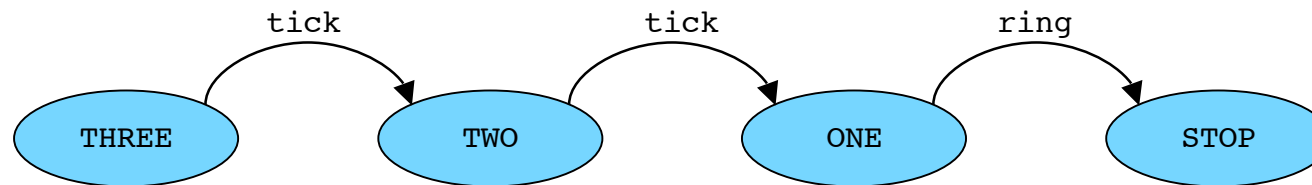
- Basic processes: Primitive processes as the basic building blocks
- Operators: Constructors forming complex processes from simpler ones
- (Equational) Laws: Relating process terms according to their behavior

Example: *STOP* = immediately deadlocking basic process



## Notation: Action Prefixing

---



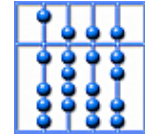
Purpose: Defining sequential behavior of an interacting process

Action Prefixing: Process term prefixing an action  $a$  before a process  $P$

- Action  $a$ : Action to be performed initially
- Process  $P$ : Process to be performed after action  $a$  has been executed

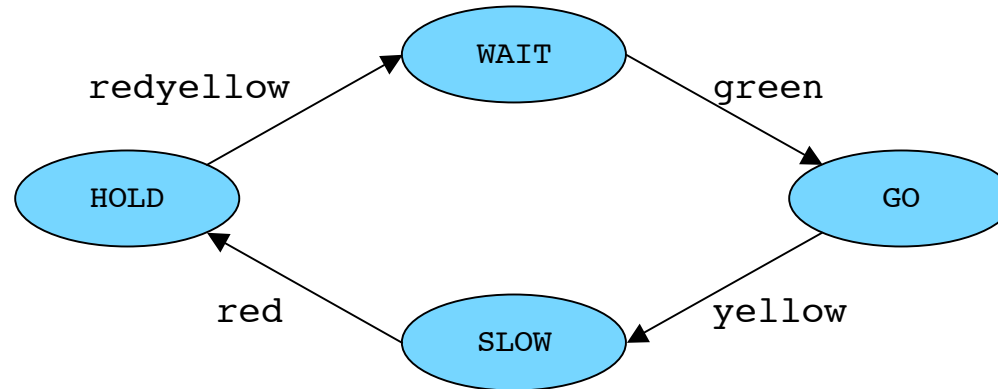
Notation:  $Process := action \rightarrow Process$

Example:  $THREE = tick \rightarrow TWO$ ,  $TWO = tick \rightarrow ONE$ ,  $ONE = ring \rightarrow STOP$



## Concept: Recursion

---



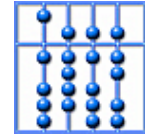
Purpose: Defining recursive behavior of a possibly non-terminating process

Recursion: Definition of a process  $P$  with body  $F$  (using a recursion variable  $X$ )

- Direct notation  $P = F(P)$ : Process label(s)  $P$  occurs on left-hand and right-hand side
- Fixed point notation  $P = \mu X. F(X)$ : Explicit definition for recursion variable  $X$

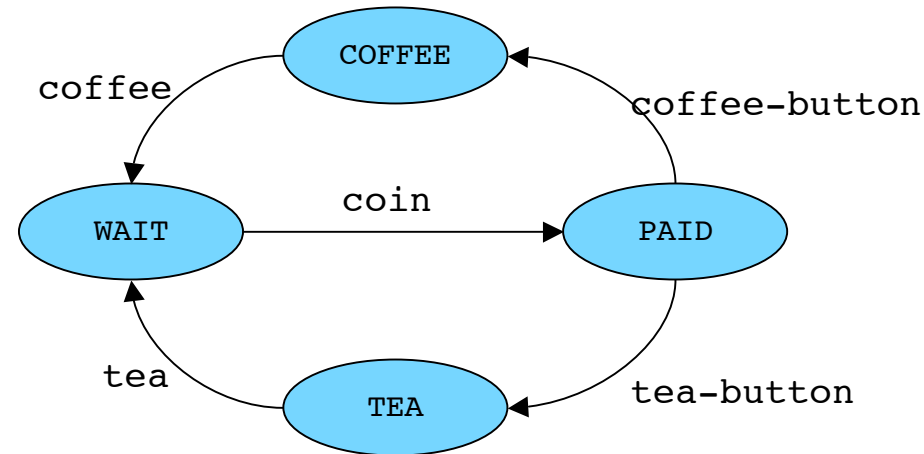
Example:

- Direct notation:
  - $\text{HOLD} = \text{redyellow} \rightarrow \text{green} \rightarrow \text{yellow} \rightarrow \text{red} \rightarrow \text{HOLD}$
  - $\text{HOLD} = \text{redyellow} \rightarrow \text{WAIT}, \text{WAIT} = \text{green} \rightarrow \text{GO}, \text{GO} = \text{yellow} \rightarrow \text{SLOW}, \text{SLOW} = \text{red} \rightarrow \text{HOLD}$
- Fixed point notation:  $\text{HOLD} = \mu X. \text{redyellow} \rightarrow \text{green} \rightarrow \text{yellow} \rightarrow \text{red} \rightarrow X$



## Notation: Choice

---



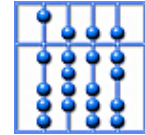
Purpose: Defining alternative behavior of a sequential interacting process

Choice: Process term combining two alternative processes  $P$  and  $Q$

- Process can perform actions that either  $P$  or  $Q$  can perform
- Process will reject only actions that both  $P$  and  $Q$  will reject

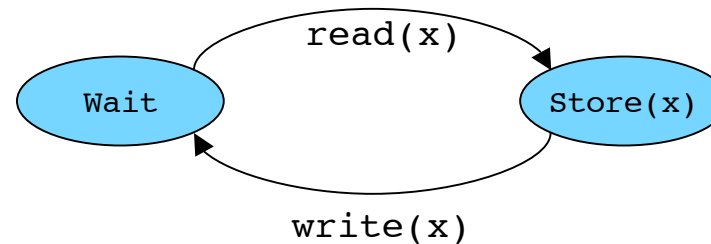
Notation:  $Process := Process \mid Process$

Example:  $PAID = (tea-button \rightarrow TEA) \mid (coffee-button \rightarrow COFFEE)$



## Notation: Parameter

---



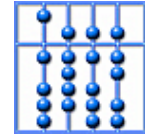
Purpose: Defining parametric behavior of a sequential interacting process

Parameter: Process term depending on a formal parameter  $x$  with domain  $D$

- Parametric action: Process definition is
- Parametric process:

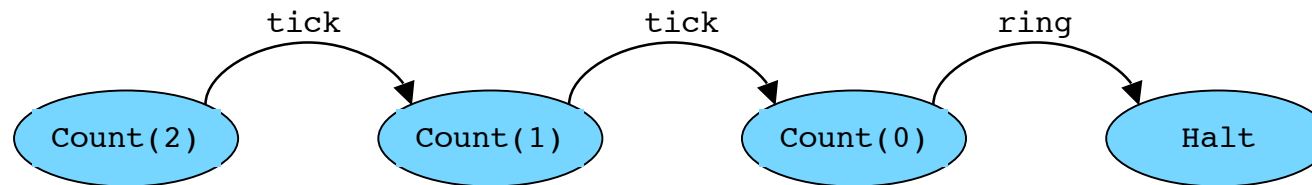
Notation:  $action := action(x); Process := Process(x)$

Example:  $Wait = read(x) \rightarrow Store(x), Store(x) = write(x) \rightarrow Wait$



## Concept: Guard

---



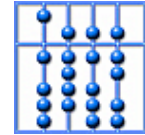
Purpose: Defining control flow of actions, depending on parameters

Guarded action: Process term depending on a condition  $c$  and a process  $P$

- Valid condition: If  $c$  is valid, the process is equal to  $P$
- Invalid condition: If  $c$  is invalid, the process is equal to  $STOP$

Notation:  $Process := Condition \triangleright Process$

Example:  $Count(x) = ((x > 0) \triangleright (tick \rightarrow Count(x-1))) \mid ((x = 0) \triangleright (ring \rightarrow Halt))$



## 5.2 Communicating Processes: Concurrent Processes

---

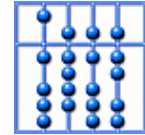
**Goal:** Introduce notations to describe the behavior of rendezvous-based systems

- Definition of alphabet of possible interactions
- Composition of process to concurrent processes
- Hiding of internal actions

**Concept:** Concurrent execution, synchronous communication

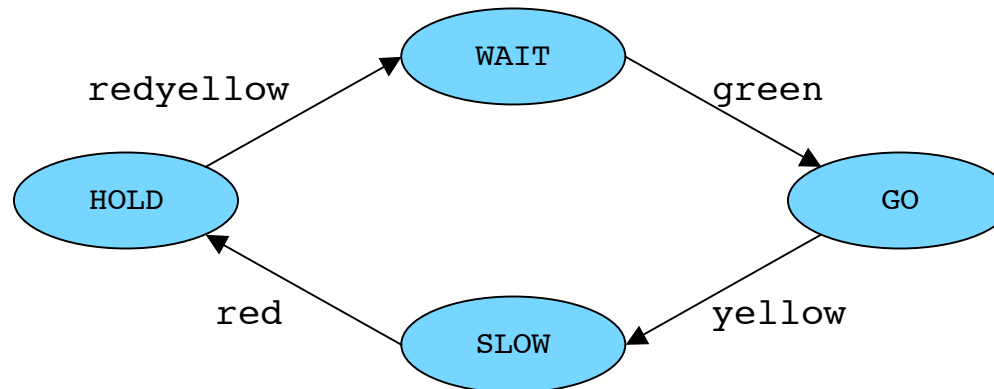
**Notation:** (T)CSP (Communicating Sequential Processes)

**Model:** Synchronized Labeled Transition Systems



## Concept: Alphabet

---

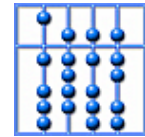


Purpose: Describing interactions between process and environment

Alphabet: Set of interactions a process can possibly participate in

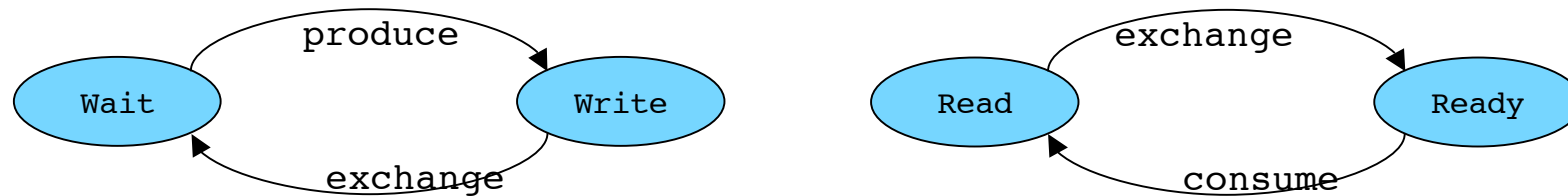
- Implicit interactions: Actions occurring in prefixes
- Explicit alphabet: Actions defined in explicit annotations (e.g.,  $STOP_{\{green, red\}}$ )

Example: Alphabet of process HOLD is { redyellow, green, yellow, red }



## Notation: Composition

---



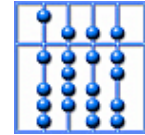
Purpose: Defining concurrent processes

Composition: Process term composing processes  $P$  and  $Q$

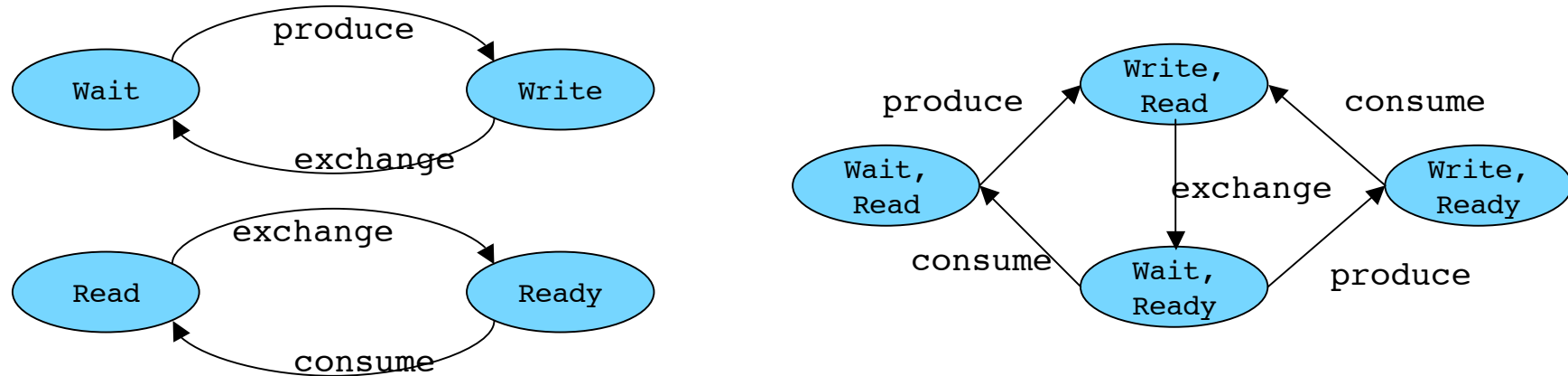
- Process: Concurrent execution of processes  $P$  and  $Q$
- Alphabet: Joint alphabet of processes  $P$  and  $Q$

Notation:  $Process := Process \parallel Process$

Example:  $ProdCon = Wait \parallel Read$ ,  $Wait = produce \rightarrow exchange \rightarrow Wait$ ,  
 $Read = exchange \rightarrow consume \rightarrow Read$



## Concept: Synchronized composition



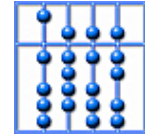
Purpose: Defining behavior of concurrent processes

Synchronization: Concurrent execution of shared

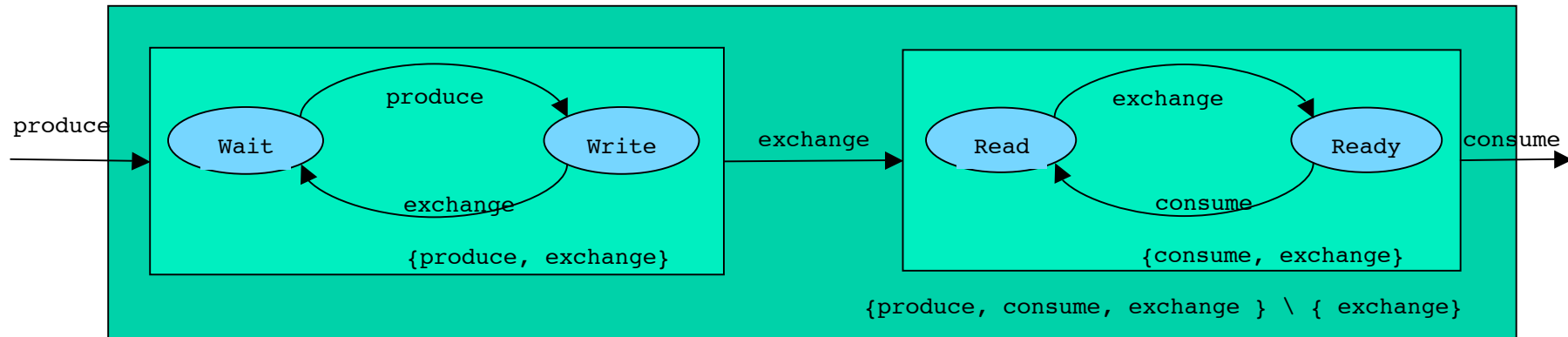
- Joint actions: Actions in the common alphabet are executed simultaneously
- Separate actions: Actions in the disjoint alphabet are executed sequentially

Equivalent Model: Synchronized labeled transition system of processes  $P$  and  $Q$

Example:  $\text{ProdCons} = \text{prod} \rightarrow \text{exch} \rightarrow (\text{ProCon} \mid \text{ConPro})$ ,  $\text{ProCon} = \text{prod} \rightarrow \text{cons} \rightarrow \text{exch} \rightarrow (\text{ProCon} \mid \text{ConPro})$ ,  $\text{ConPro} = \text{cons} \rightarrow \text{prod} \rightarrow \text{exch} \rightarrow (\text{ProCon} \mid \text{ConPro})$



## Concept: Hiding



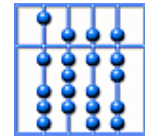
Purpose: Distinguishing internal interactions (between processes) and external interactions (between processes and environment)

Hiding: Eliminating interactions  $H$  from a Process  $P$

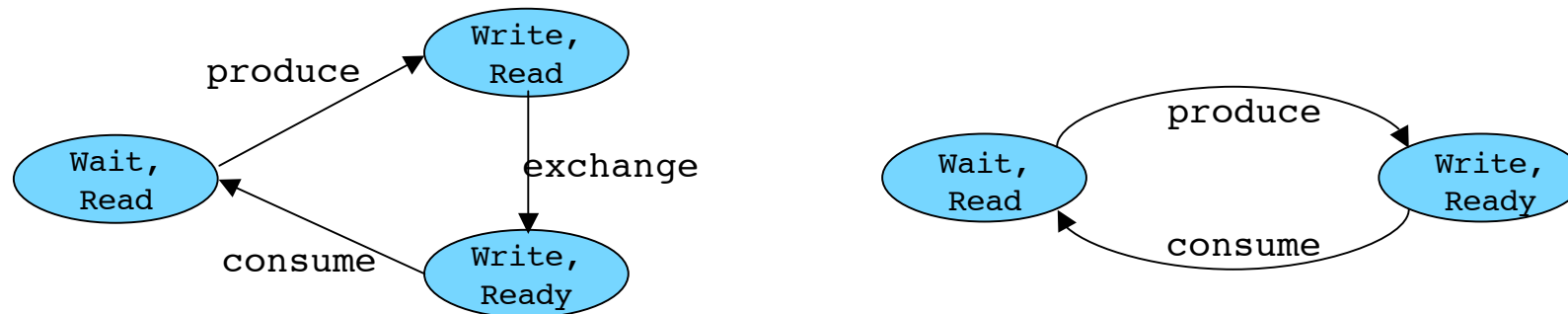
- Abstraction: Make interactions from  $H$  invisible to the environment
- Alphabet: Alphabet is alphabet of  $P$  without interactions from  $H$

Notation:  $Process := Process \setminus Alphabet$

Example: Alphabet of process  $ProdCon \setminus \{exchange\}$ , is  $\{produce, consume\}$



## Concept: Hiding

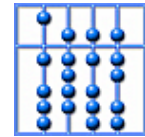


Purpose: Abstracting from internal actions performed by a process

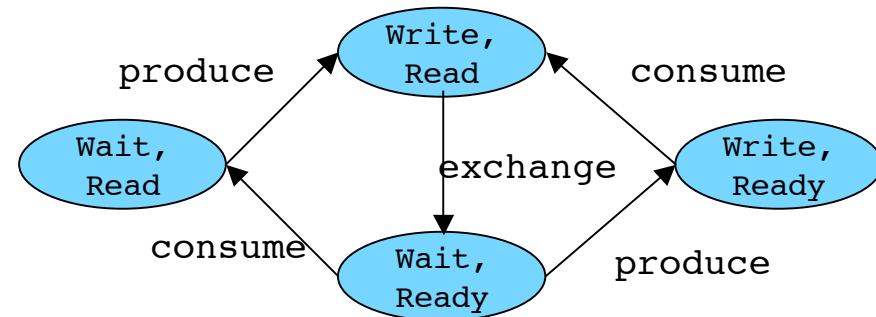
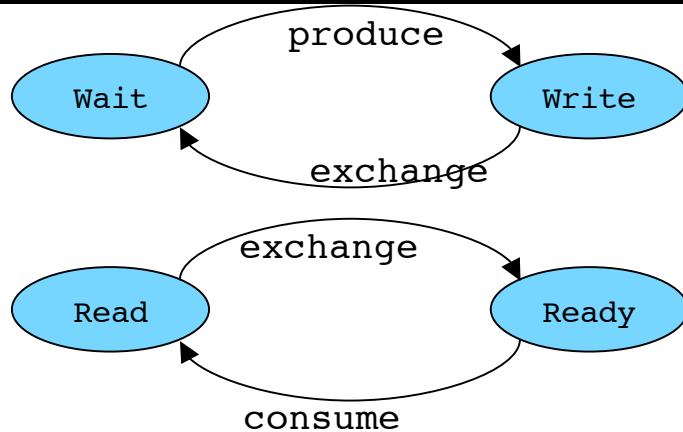
Hiding: Behavior of process abstracting from internal actions

- Internal actions: Occurring immediately, as soon as action is enabled
- External actions: Occuring synchronized with environment

Example:  $\text{AbsProCon} = \text{ProdCons} \setminus \{\text{exchange}\}$ ,  $\text{AbsProCon} = \text{produce} \rightarrow \text{APC}$ ,  
 $\text{APC} = (\text{produce} \rightarrow \text{consume} \rightarrow \text{APC}) \mid (\text{consume} \rightarrow \text{produce} \rightarrow \text{APC})$



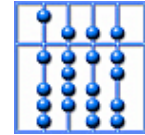
## Process Algebra: Laws



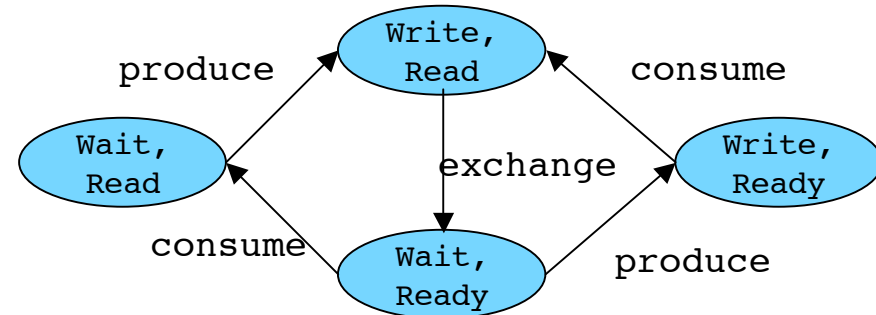
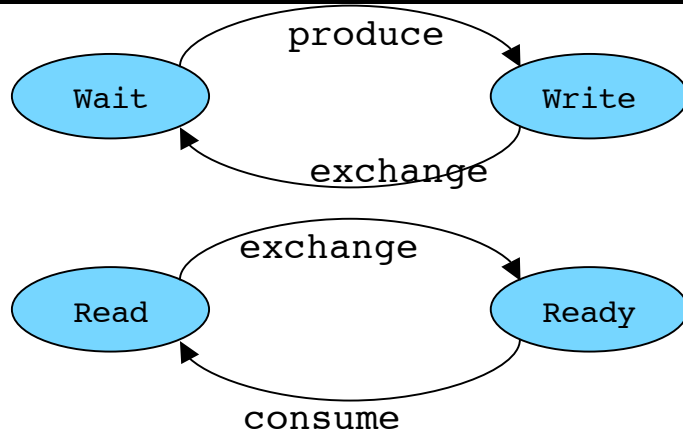
Algebraic Laws: Equations relating process terms with identical behavior

Examples:

- $P \mid \text{STOP} = P$
- $\text{false} \triangleright P = \text{STOP}$
- $\mu X. F(X) = F(\mu X. F(X))$
- $\text{STOP}_{A \setminus H} = \text{STOP}_{A \setminus H}$
- $(a \rightarrow P) \setminus \{a\} = P \setminus \{a\}$



## Process Algebra: Laws



Algebraic Laws: Equations relating process terms with identical behavior

Examples:

- $P \parallel Q = Q \parallel P$
- $P \parallel (Q \parallel R) = (P \parallel Q) \parallel R$
- $P \parallel \text{STOP}_A = \text{STOP}_A$  (for processes  $P$  with alphabet  $A$ )
- $(a \rightarrow P) \parallel (a \rightarrow Q) = a \rightarrow (P \parallel Q)$
- $(a \rightarrow P) \parallel (b \rightarrow Q) = \text{STOP}$  (for shared actions  $a, b$ )
- $(a \rightarrow P) \parallel (b \rightarrow Q) = (a \rightarrow (P \parallel (b \rightarrow Q))) \mid (b \rightarrow ((a \rightarrow P) \parallel Q))$  (for disjoint actions  $a, b$ )