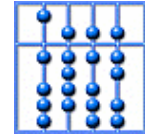


---

## **Specification of Distributed Systems**

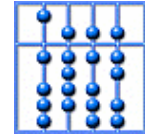
Dr. Bernhard Schätz  
Leopold-Franzens Universität Innsbruck  
Sommersemester 2005



## 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

1. Modeling Computation: State Transition Systems
2. Modeling Interaction: Labeled Transition Systems
3. Modeling Concurrency: Synchronized Transition Systems

### 3. Coroutines

### 4. Communicating Processes

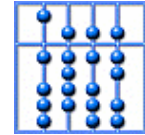
### 5. Data Flow Models

### 6. State-Based Models

### 7. Coordination

### 8. Executions

### 9. Property Descriptions



## 2.1 Modeling Computation: Transition Systems

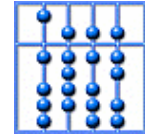
---

**Goal:** Define a model to describe the behavior of algorithmic computations

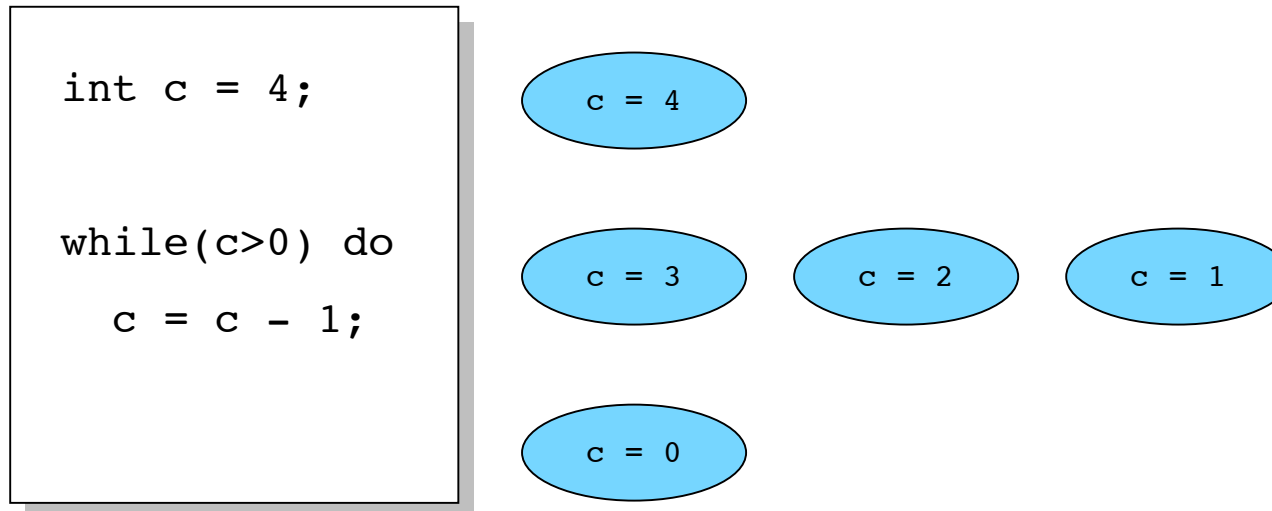
- Relevant: Address aspects to be solved by a computation
- General: Cover aspects independent of specific programming languages
- Abstract: Ignore aspects like execution time, memory limitations

**Concept:** State, transition, execution trace, behavior

**Model:** State Transition System



## Concept: State

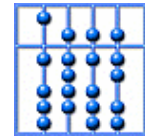


Purpose: Modeling the effect of computations (program steps)

Concept: State = Set of assignments

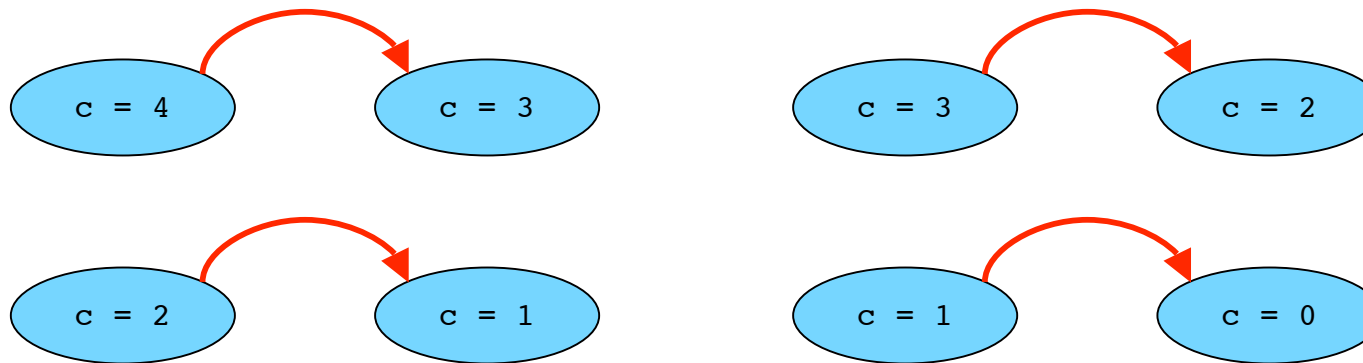
- Data state, e.g., variables, memory locations
- Control state, e.g., program counter

Example:  $\{ c = 3 \}$ ,  $\{ c = 0 \}$



## Concept: Transition

---

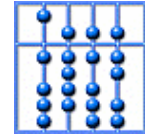


Purpose: Modeling the behavior of a (sequential) program

Concept: Transition = pair of states

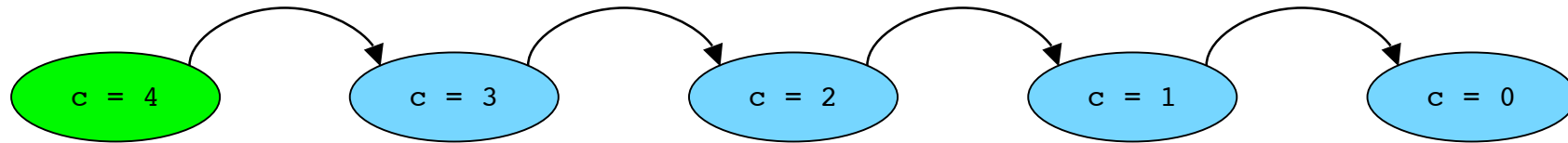
- Describes the change between the states of the program
- Corresponds to an atomic, instantaneous action of the program

Example:  $\{ c = 3 \} \rightarrow \{ c = 2 \}$  or  $(\{ c = 3 \}, \{ c = 2 \})$



## Model: Transition System

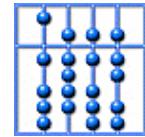
---



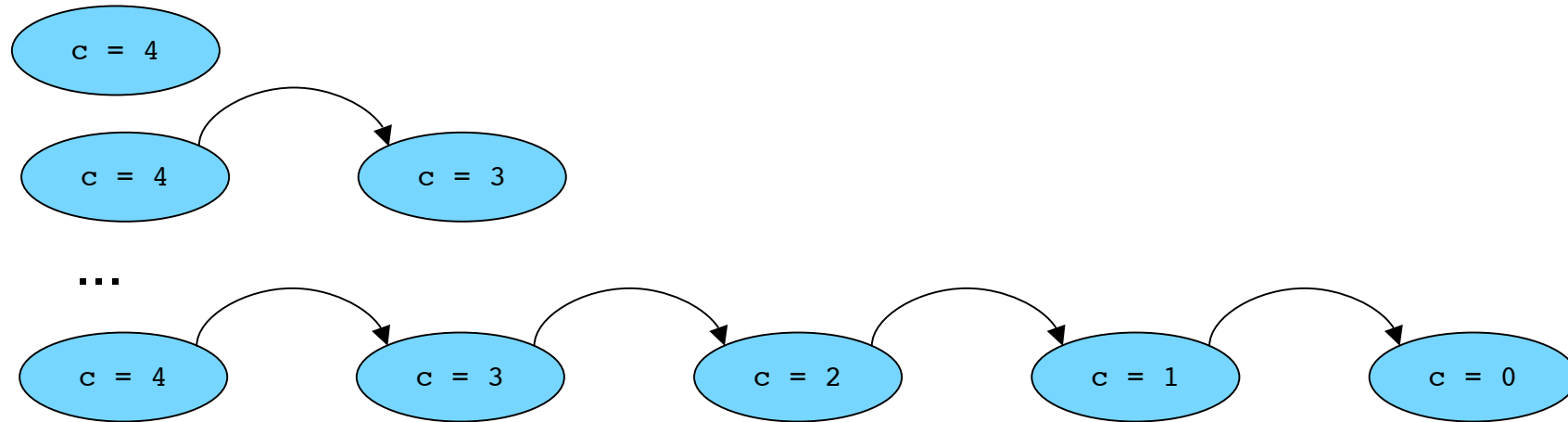
Purpose: Describe complete behavior of a (terminating) program

Model: Transition System  $(S, s_0, T)$

- Set of possible program states  $S$
- Initial state  $s_0 \subseteq S$
- Set of possible transitions  $T \subseteq S \times S$
- Example:
  - $S = \{ \{c = 4\}, \{c = 3\}, \{c = 2\}, \{c = 1\}, \{c = 0\} \}$
  - $s_0 = \{c = 4\}$
  - $T = \{ (\{c = 4\}, \{c = 3\}), (\{c = 3\}, \{c = 2\}), \dots, (\{c = 1\}, \{c = 0\}) \}$



## Concept: Execution Trace

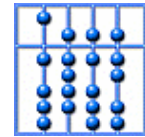


Purpose: Modeling a part of an execution of a program

Concept: Trace = sequence of consecutive states  $s_1 \cdot s_2 \cdot s_3 \cdot s_4 \cdot \dots$

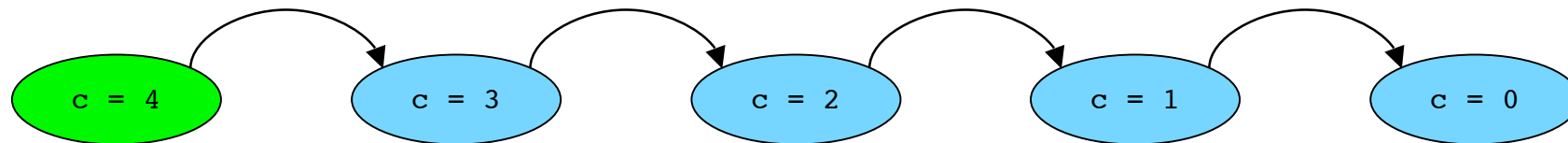
- First state is initial state:  $s_1 = s_0$
- Consecutive states linked by transitions:  $(s_i, s_{i+1}) \in T$

Example:  $\{ c = 4 \} \cdot \{ c = 3 \} \cdot \{ c = 2 \}$



## 3.1 Summary: Modeling Computations

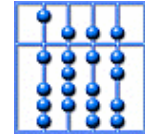
---



Concepts:

- State: Observation of a system at a specific instance of time
- Transition: Atomic action changing the state of a computation
- Transition relation: Set of possible actions of a computation
- Execution Trace: Sequence of states during a computation

Model: Transition System  $(S, s_0, T)$



## 2.2 Modeling Interactions: Labeled Transition Systems

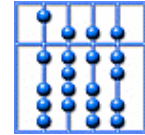
---

**Goal:** Define a model to describe the behavior of interactive and concurrent systems (including sequential systems)

- Relevant: Address aspects to be solved by interaction
- General: Cover aspects independent of specific distributed architecture
- Abstract: Ignore aspects like execution time, communication bandwidth

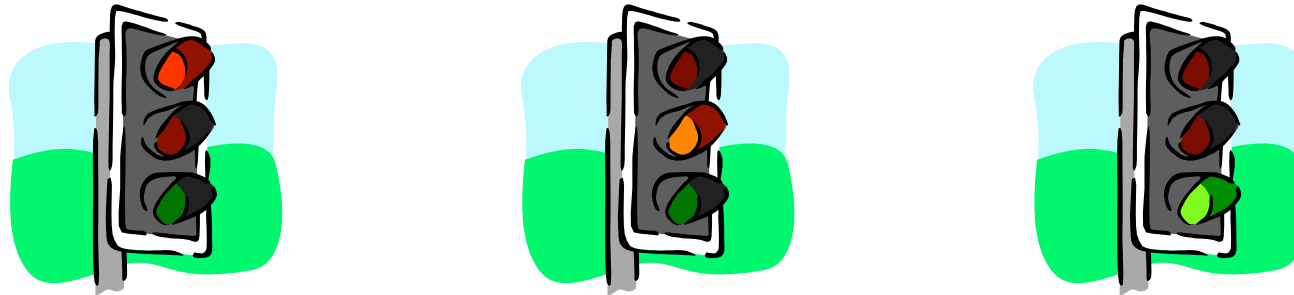
**Concept:** Interaction, observation trace, behavior

**Model:** Labeled Transition System



## Concept: Interaction

---

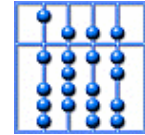


Purpose: Describing interactions between system and environment

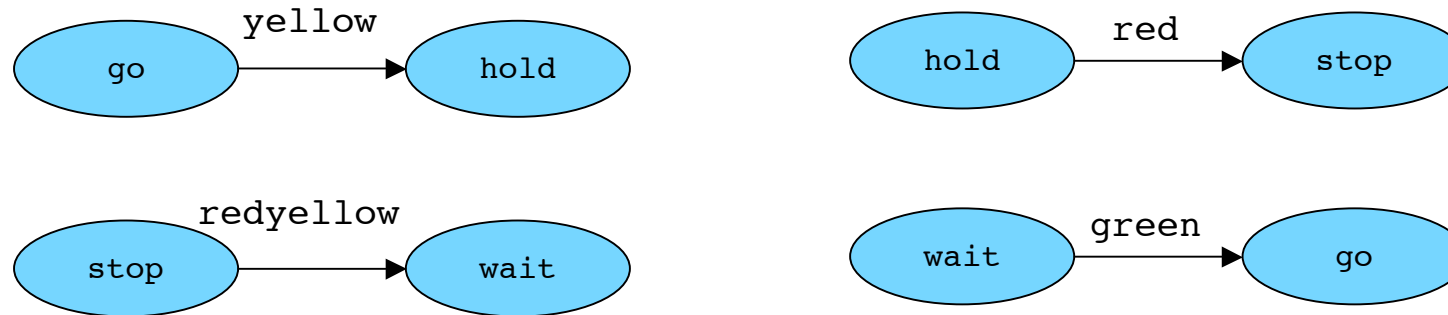
Concept: Interaction = joint action/observation at system interface

- Simple interaction (atomic and instantaneous)
- Complex interaction (combinations of atomic interactions)

Example: “Traffic lights turn red”, “Traffic lights turn green”



## Concept: Labeled Transitions

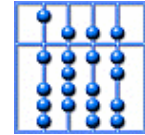


Purpose: Modeling the behavior of a (reactive) system

Concept: Transition = observed interaction and change of states

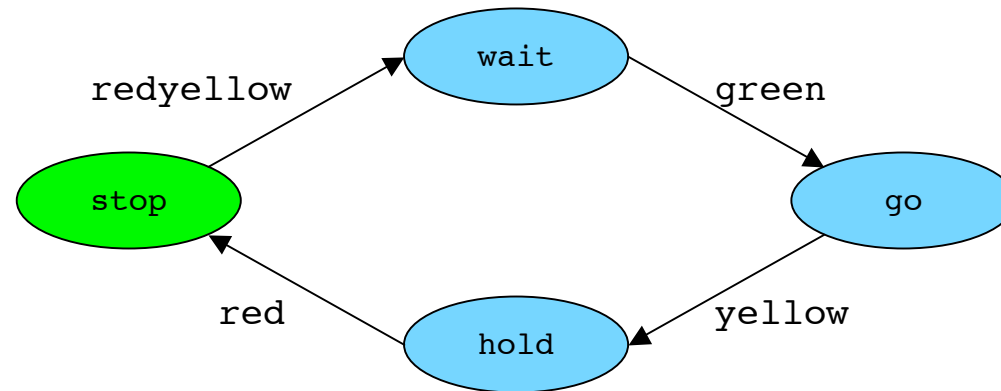
- Describes the change between the states of the system
- Describes the interactions corresponding to this change

Example:  $go \xrightarrow{\text{yellow}} hold$     $wait \xrightarrow{\text{green}} go$   
or (go,yellow,hold), (wait,green,go)



## Model: Labeled Transition System

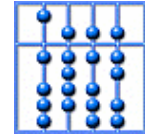
---



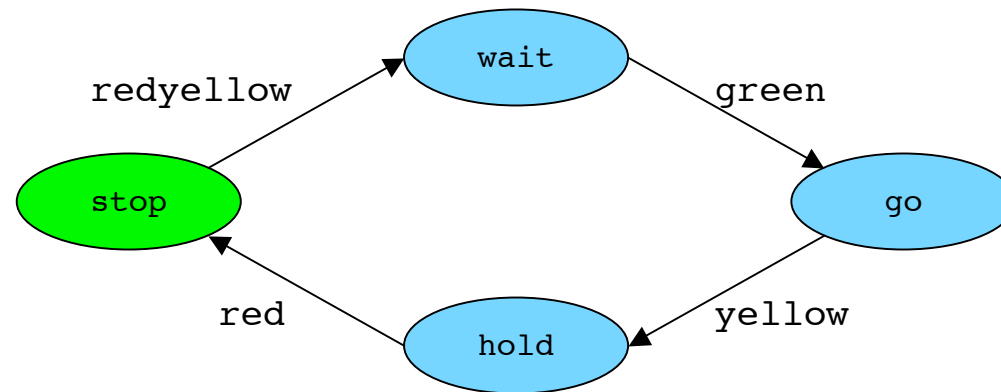
Purpose: Describe complete behavior of a reactive system

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

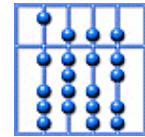
- Set of possible system states  $S$
- Set of possible simple interactions  $A$  (alphabet)
- Set of initial states  $S_0 \subseteq S$
- Set of possible transitions  $T \subseteq S \times A \times S$



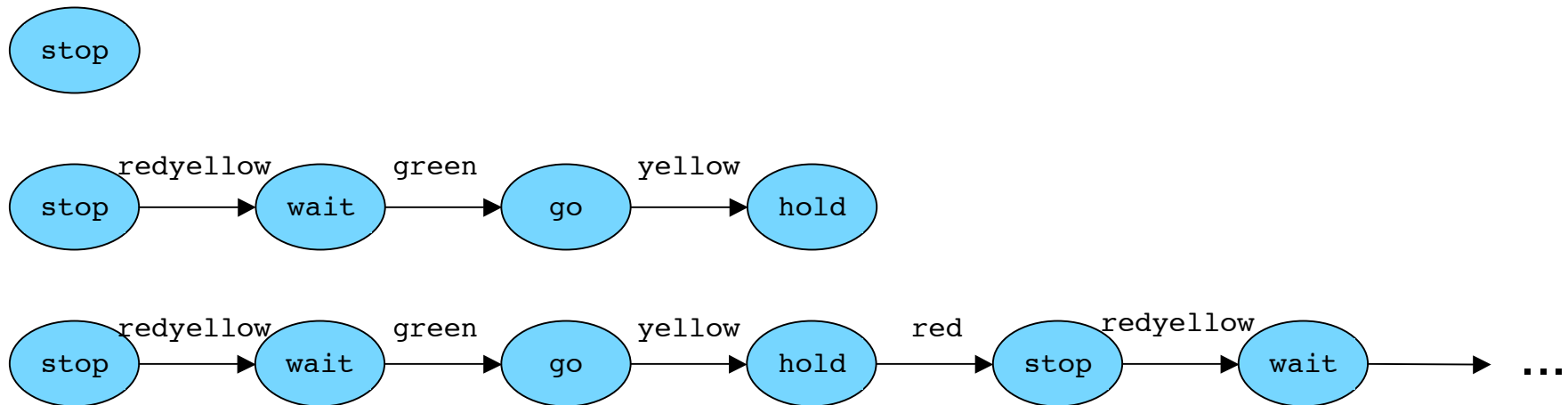
## Model: Labeled Transition System



- Example:
  - $S = \{ \text{stop, wait, go, hold} \}$
  - $A = \{ \text{green, yellow, yellowred, red} \}$
  - $S_0 = \{ \text{stop} \}$
  - $T = \{ (\text{stop, redyellow, wait}), (\text{wait, green, go}), (\text{go, yellow, hold}), (\text{hold, red, stop}) \}$
  - Or:  $T = \{ \text{stop} \xrightarrow{\text{redyellow}} \text{wait}, \text{wait} \xrightarrow{\text{green}} \text{go}, \text{go} \xrightarrow{\text{yellow}} \text{hold}, \text{hold} \xrightarrow{\text{red}} \text{stop} \}$



## Concept: Execution Trace

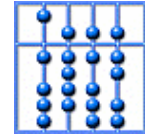


Purpose: Modeling a single (partial) behavior of a program

Concept: Trace = (finite or infinite) sequence of consecutive states and the corresponding interactions  $s_1 \cdot a_1 \cdot s_2 \cdot a_2 \cdot s_3 \cdot a_3 \cdot s_4 \cdot \dots$

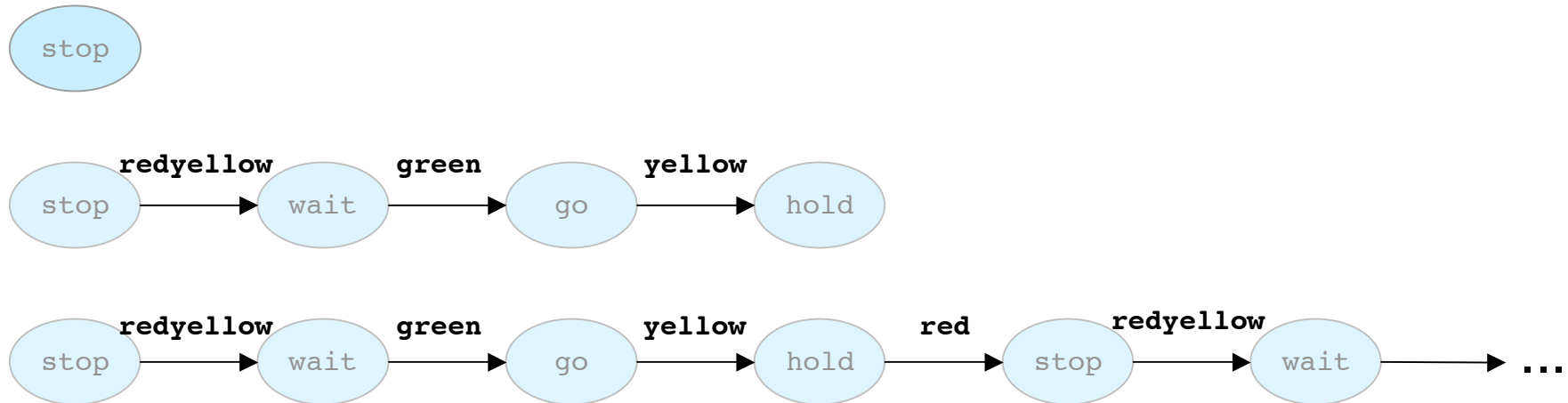
- First state is a initial state:  $s_1 \in S_0$
- Consecutive states linked by transitions:  $(s_i, a_i, s_{i+1}) \in T$

Example: stop • redyellow • wait • green • go

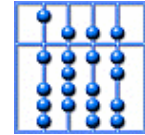


## Concept: Observation Trace

---

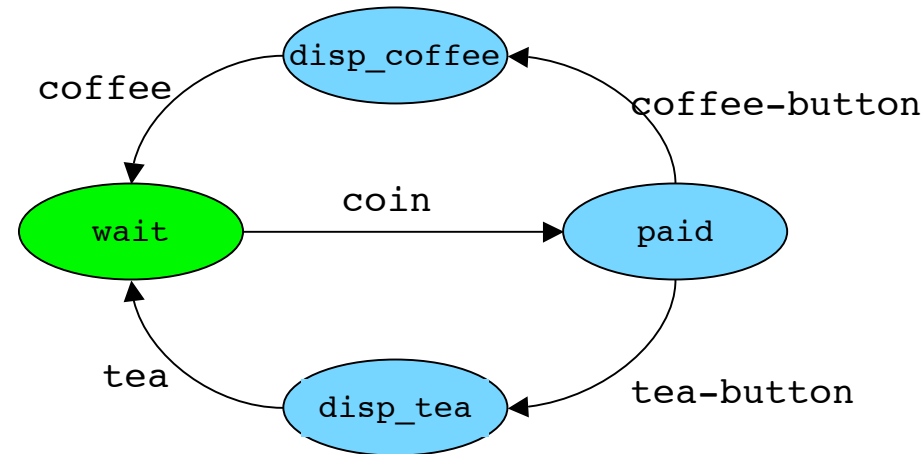


- Purpose: Modeling a single (partial) behavior of a program visible at its interface
- Concept: Trace = (finite or infinite) sequence of consecutive interactions  $a_1 \cdot a_2 \cdot a_3 \cdot a_4 \cdot \dots$ 
  - First state is a initial state:  $s_1 \in I$
  - Consecutive states linked by transitions:  $(s_i, a_i, s_{i+1}) \in T$
- Example:  $\text{redyellow} \cdot \text{green} \cdot \text{yellow} \cdot \text{red} \cdot \text{redyellow} \cdot \text{green} \cdot \text{yellow} \cdot \dots$



## Concept: Choices

---

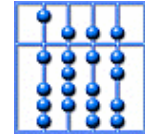


Purpose: Describe controllable alternatives of behaviors of a reactive system

Concept: Alternative choice of interactions  $a_1, a_2$  at state  $s$

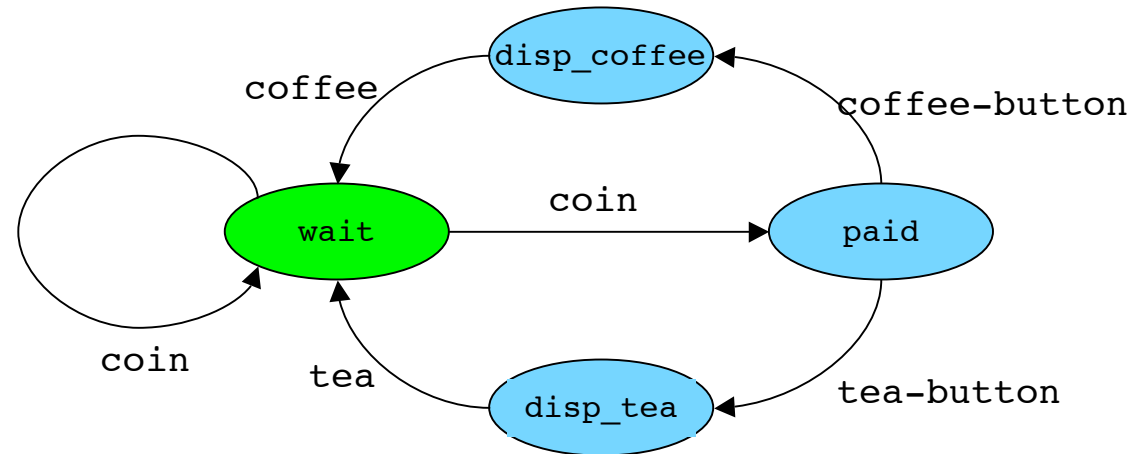
- $(s, a_1, s_1) \in T$
- $(s, a_2, s_2) \in T$

Example:  $(\text{paid}, \text{coffee-button}, \text{disp\_coffee}), (\text{paid}, \text{tea-button}, \text{disp\_tea}) \in T$



## Concept: Nondeterminism

---

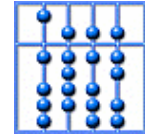


Purpose: Describe uncontrollable alternatives of behaviors of a reactive system

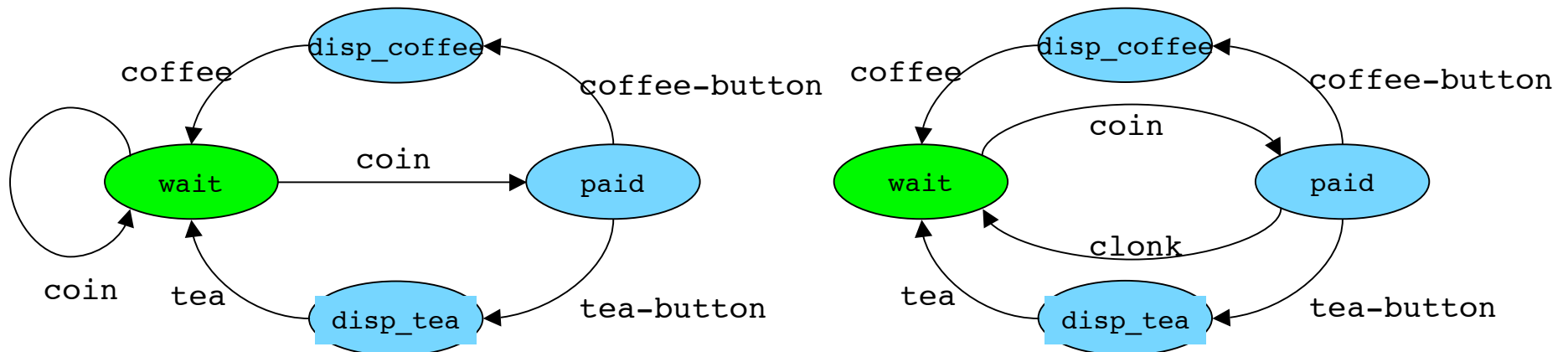
Concept: Alternative result of interaction  $a$  at state  $s$

- $(s, a, s_1) \in T$
- $(s, a, s_2) \in T$

Example:  $(\text{wait}, \text{coin}, \text{paid}), (\text{wait}, \text{coin}, \text{wait}) \in T$



## Choice vs Nondeterminism

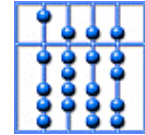


Alternative behavior of a system:

- Choice: Controlled by environment (user of a system)
- Nondeterminism: Controlled by system

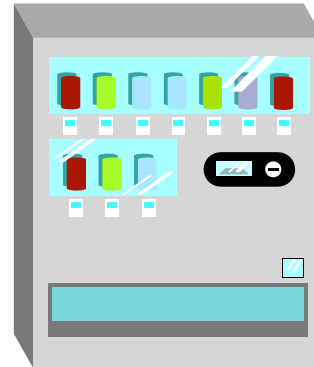
Point of view: Who controls an interaction?

- Symmetric view: System and environment must agree on action
- Asymmetric view: Distinction between input and output



## Concept: Input and Output

---

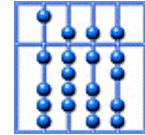


Purpose: Distinguish between actions of environment and actions of systems

Concept: Input actions  $I$  and output actions  $O$ , forming alphabet  $A$

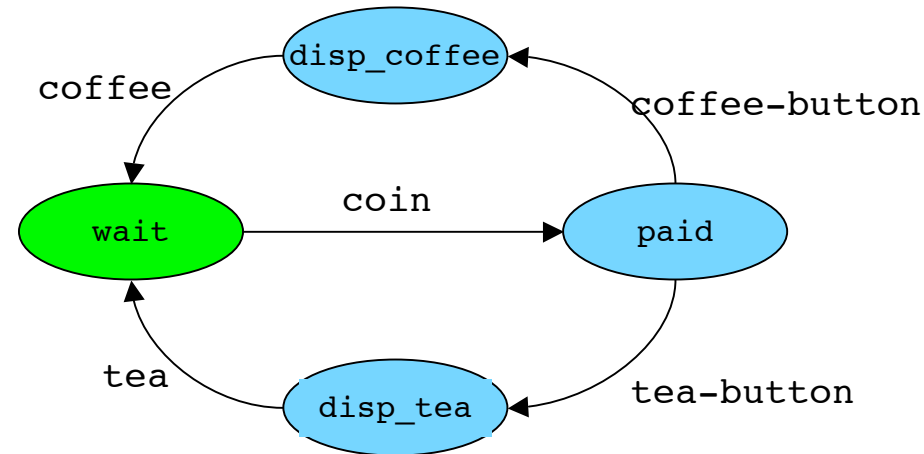
- Input  $I$ : Actions controlled by environment
- Output  $O$ : Actions controlled by system

Example:  $I = \{\text{coin, coffee-button, tea-button}\}$ ,  $O = \{\text{coffee, tea}\}$ ,  $A = I \cup O$



## Concept: Input and Output

---

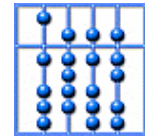


Adaption: Transition System Labeled with Input and Output Actions

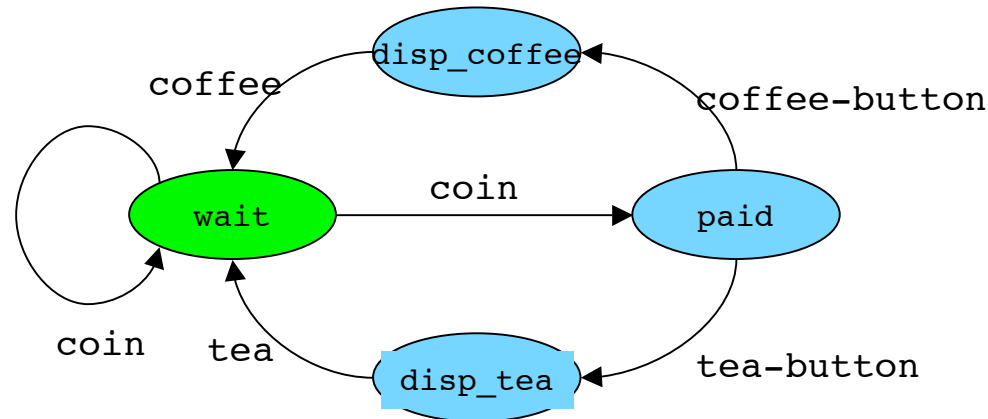
Concept: Labeled Transition System (S,s,I,O,T)

- Input I: Actions controlled by environment
- Output O: Actions controlled by system

Example:  $I = \{\text{coin, coffee-button, tea-button}\}$ ,  $O = \{\text{coffee, tea}\}$



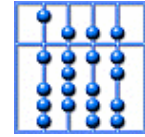
## 3.2 Summary: Modeling Reactivity



### 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  $(S, A, S_0, T)$



## 2.3 Modeling Concurrency: Synchronized Transition Systems

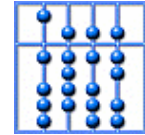
---

**Goal:** Define a model to describe the behavior of concurrently executing reactive systems

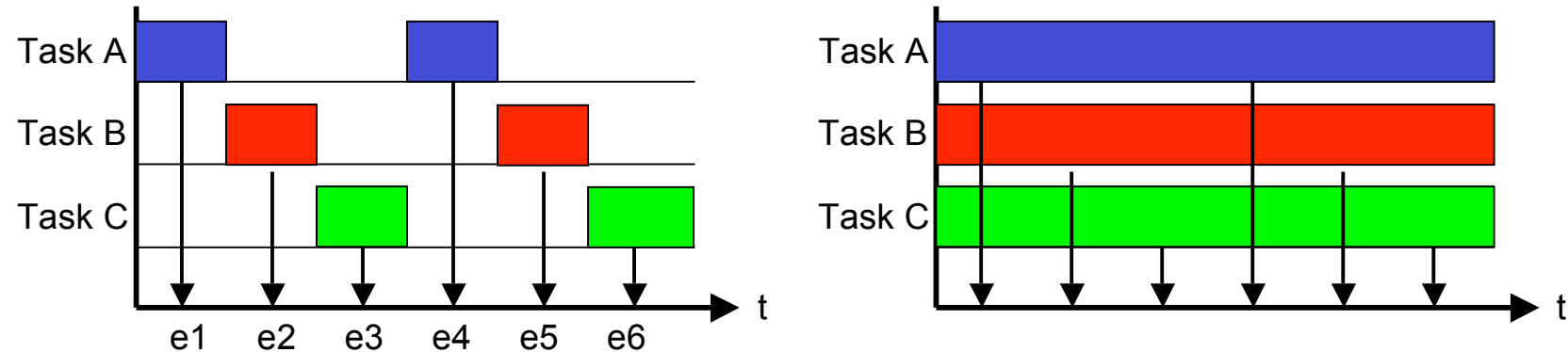
- Relevant: Address aspects evolving with synchronization
- General: Cover aspects independent of specific communication/synchronization techniques
- Abstract: Ignore aspects like communication bandwidth, execution speed

**Concept:** Shared/independent interaction, shared/independent transition, concurrent execution

**Model:** Synchronized Transition System



## Concept: Concurrency and Parallelism

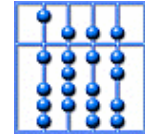


Purpose: Describing parallel executions of systems

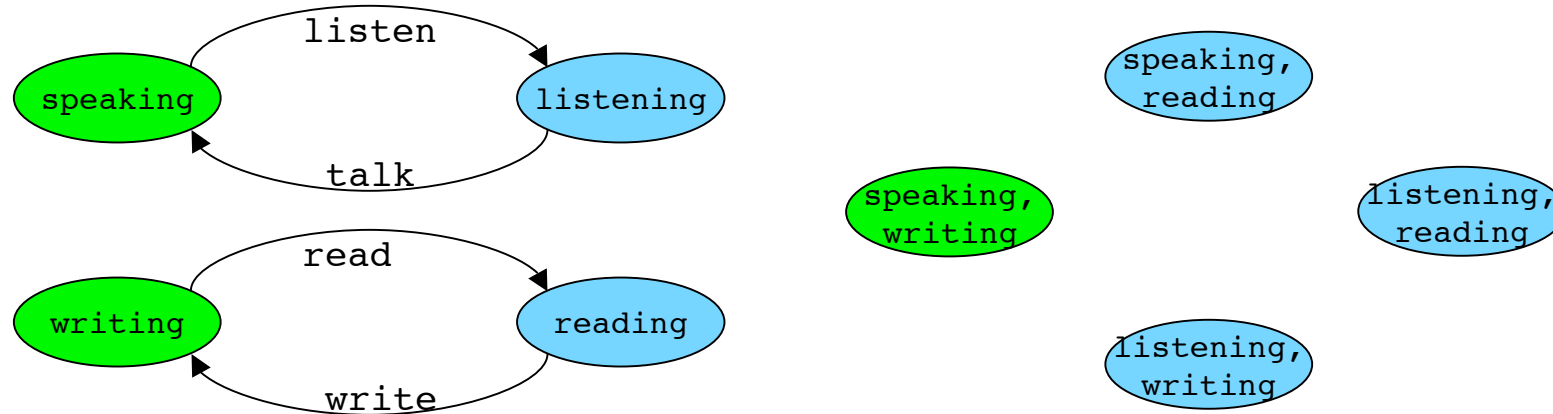
Concept: Simultaneous execution

- Concurrent: Logically simultaneous (simultaneous assuming sequential observation)
- Parallel: Physically simultaneous (simultaneous assuming instantaneous observation)

Example: Pre-emptive multi-tasking vs. multi-processor execution



## Concept: Concurrent Execution



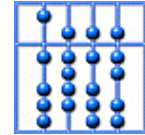
Purpose: Describing concurrent executions of systems

Concept: States of  $(S, A, S_0, T)$  and  $(S', A', S'_0, T')$  executed in parallel

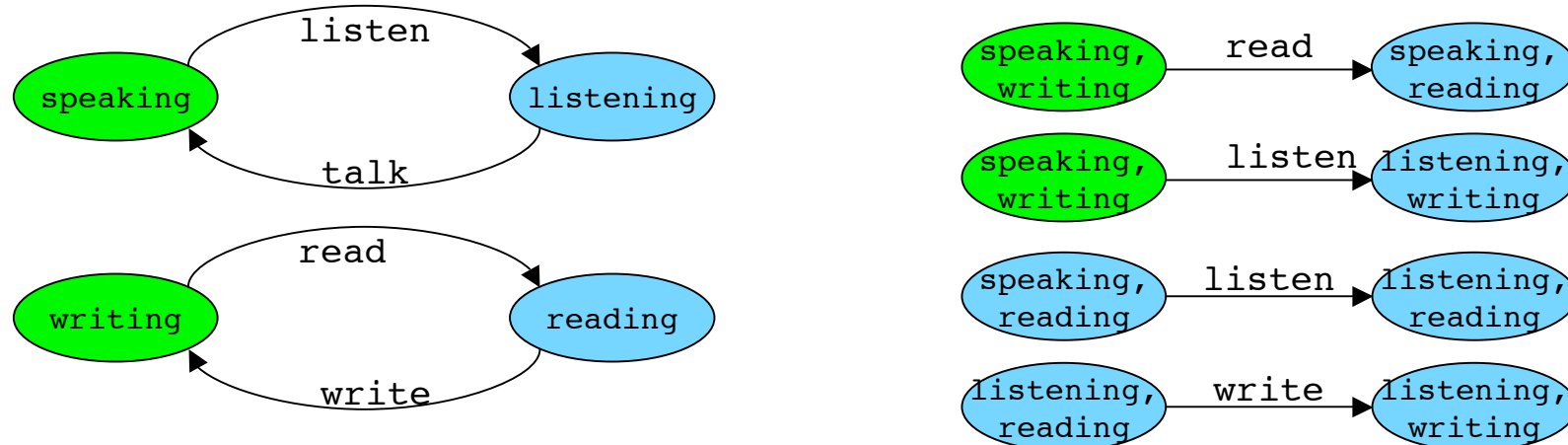
- Combination of all states of each LTA: Product state  $S \times S'$
- Initial states:  $S_0 \times S'_0$

Example:  $(\{\text{speaking}, \text{listening}\}, A, \{\text{speaking}\}, T)$  and  $(\{\text{writing}, \text{reading}\}, A', \{\text{writing}\}, T')$

- Product state:  $\{\text{speaking}, \text{listening}\} \times \{\text{reading}, \text{writing}\} = \{(\text{speaking}, \text{writing}), (\text{speaking}, \text{reading}), (\text{listening}, \text{writing}), (\text{listening}, \text{reading})\}$
- Initial state =  $\{\text{speaking}\} \times \{\text{writing}\} = \{(\text{speaking}, \text{writing})\}$



## Concept: Independent Interaction



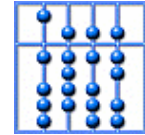
Purpose: Describing independent interactions of systems

Concept: Transitions for independent interactions of  $(S, A, S_0, T)$  and  $(S', A', S'_0, T')$

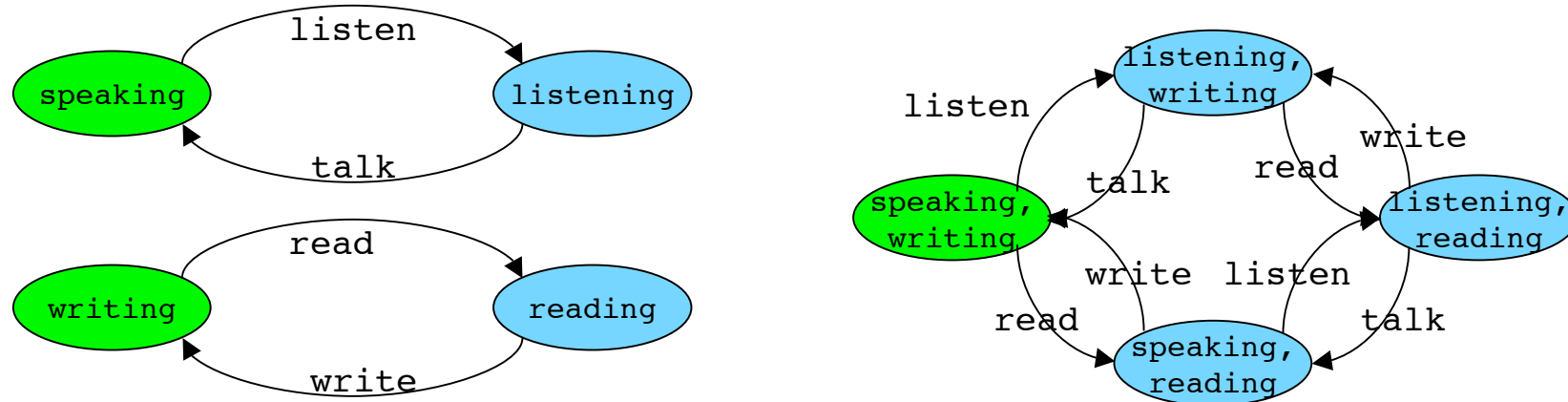
- $((s, s'), a, (t, s'))$  for each  $(s, a, t) \in T$ , and  $a \in A \setminus A'$
- $((s, s'), a', (s, t'))$  for each  $(s', a', t') \in T'$ , and  $a' \in A' \setminus A$

Example:

- $((\text{speaking}, \text{writing}), \text{read}, (\text{speaking}, \text{reading}))$
- $((\text{listening}, \text{reading}), \text{talk}, (\text{speaking}, \text{reading}))$



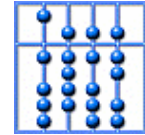
## Model: Product Automaton



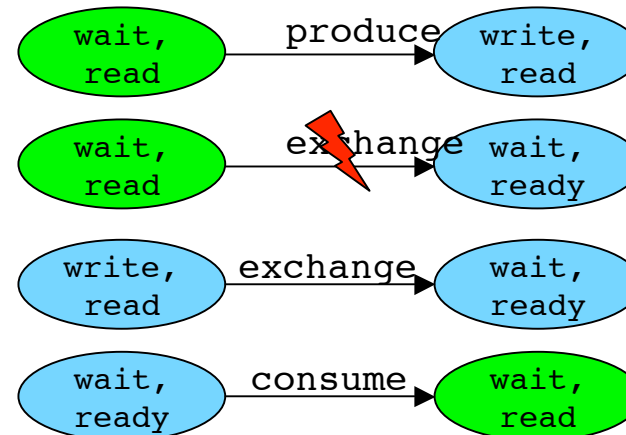
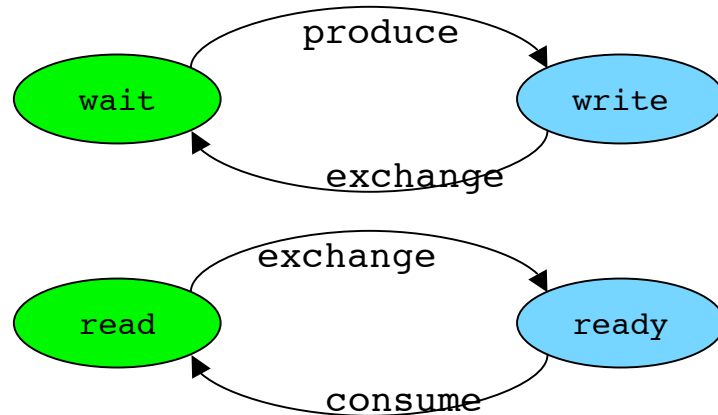
Purpose: Describing concurrent execution with independent interactions (i.e.,  $A \cap A' = \emptyset$ )

Model: Product Transition System of  $(S, A, S_0, T)$  and  $(S', A', S'_0, T')$ :

- States:  $S \times S'$
- Alphabet:  $A \cup A'$
- Initial states:  $S_0 \times S'_0$
- Transition relation  $T \parallel T'$ :
  - $((s, s'), a, (t, s'))$  for each  $(s, a, t) \in T$ , and  $a \in A \setminus A'$
  - $((s, s'), a', (s, t'))$  for each  $(s', a', t') \in T'$ , and  $a' \in A' \setminus A$



## Concept: Synchronized Interaction



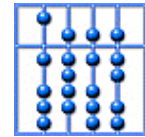
Purpose: Describing synchronized interactions of systems

Concept: Transitions for synchronized interactions of  $(S, A, S_0, T)$  and  $(S', A', S'_0, T')$

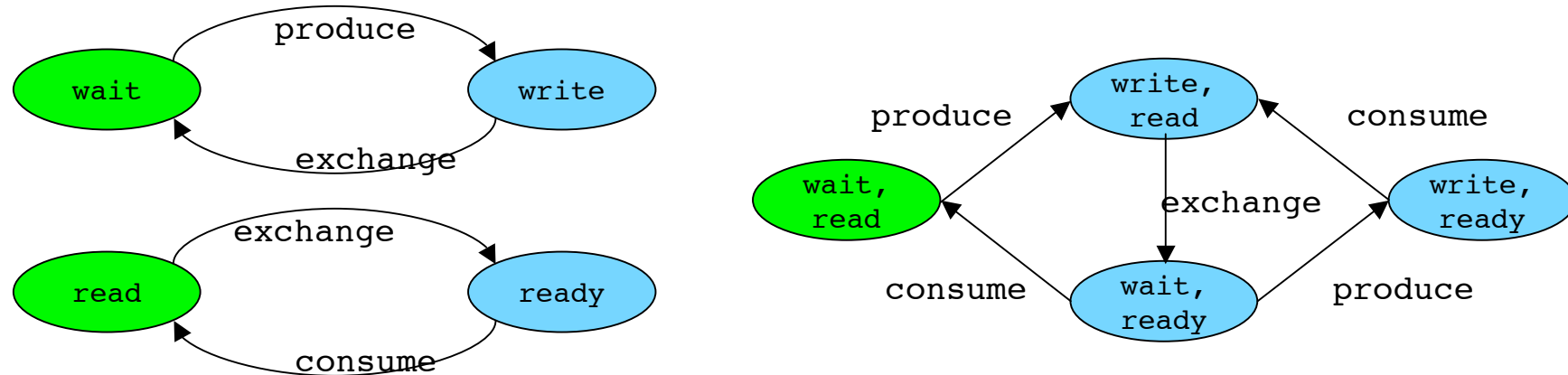
- $((s, s'), a, (t, t'))$  for each  $(s, a, t) \in T$ ,  $(s', a, t') \in T'$ , and  $a \in A \cap A'$

Example:

- $(\text{write}, \text{read}), \text{exchange}, (\text{wait}, \text{ready})$



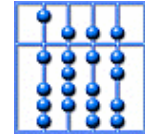
## Model: Synchronizing Product Transition System



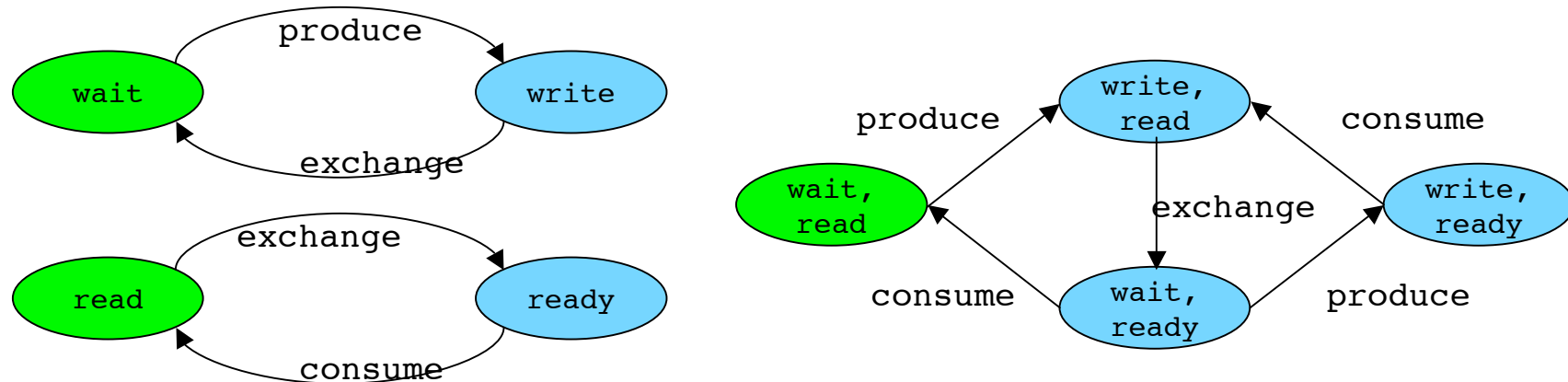
Purpose: Describing concurrent execution with synchronized interactions (i.e.,  $A \cap A' \neq \emptyset$ )

Model: Product Transition System of  $(S, A, S_0, T)$  and  $(S', A', S'_0, T')$ :

- States:  $S \times S'$
- Alphabet:  $A \cup A'$
- Initial states:  $S_0 \times S'_0$
- Transition relation  $T \parallel T'$ :
  - $((s, s'), a, (t, s'))$  for each  $(s, a, t) \in T$ , and  $a \in A \setminus A'$
  - $((s, s'), a', (s, t'))$  for each  $(s', a', t') \in T'$ , and  $a' \in A' \setminus A$
  - $((s, s'), a, (t, t'))$  for each  $(s, a, t) \in T$ ,  $(s', a, t') \in T'$ , and  $a \in A \cap A'$



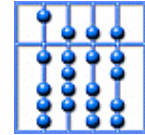
### 3.3 Summary: Modeling Concurrency



Concepts:

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

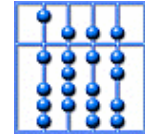
Model: Synchronized Product Automaton



## 2.4 Questions

---

1. What are the atomic actions of a Pascal/C/Lisp/Java program?
2. What are the simple interactions of a Java program?
3. Does an input/output distinction make sense in a hand-shake (message-synchronous) communication model?
4. What is the relation between the observation traces of the vending machine and the broken vending machine?
5. Is there a general principle behind the answer of question 4?
6. Is there a different machine with the same observation traces as the broken vending machine?
7. If so, is one of them more “benign” than the other, as seen from the user’s point of view?



## Questions

---

8. Can you tell - by looking at the LTS as introduced in 2.2 - whether a system is parallel or sequential?
9. What must be added to the model to distinguish between parallel and concurrent systems?
10. How does such a modified model affect the specification of the traffic light?
11. How can a trace of a subsystem be obtained from the trace of a composed system?
12. Can you tell - by looking at its traces - whether a system is parallel or sequential?
13. What must be added to obtain a more general model of traces, distinguishing between interleaved and parallel behavior?
14. What is the relationship between choice and parallelism concerning the model of LTS as introduced in 2.2?