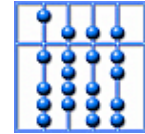


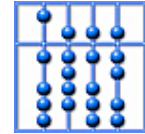
Specification of Distributed Systems

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

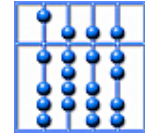
5. Data Flow Models

6. State-Based Models

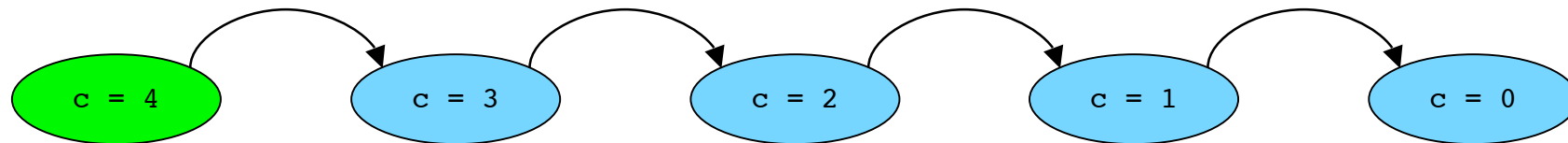
7. Coordination

8. Executions

9. Property Descriptions



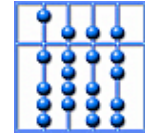
3.1 Summary: Modeling Computations



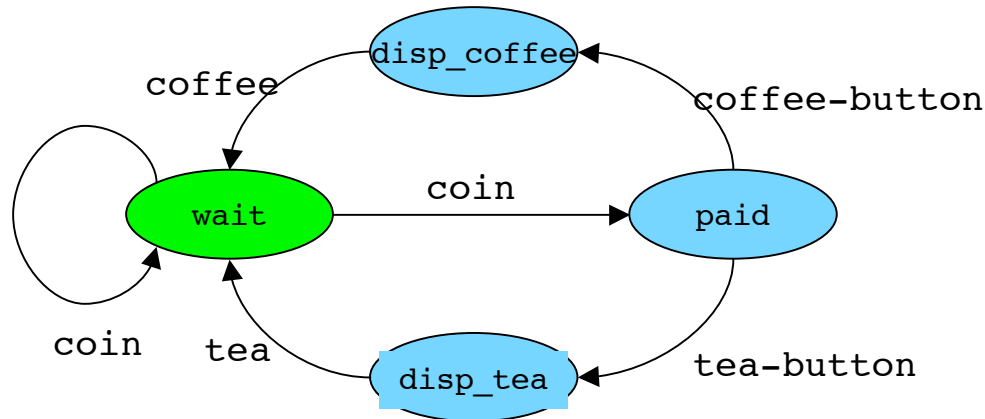
Concepts:

- 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)



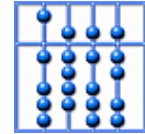
3.2 Summary: Modeling Reactivity



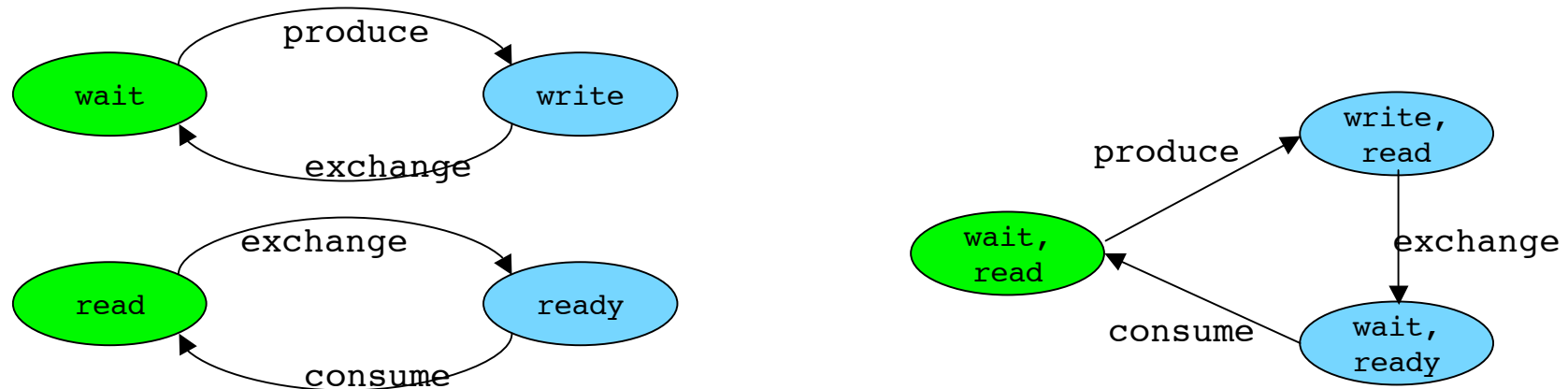
Concepts:

- 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)



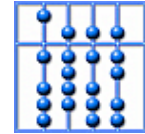
3.3 Summary: Modeling Concurrency



Concepts:

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

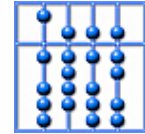
Model: Synchronized Product Automaton



Exercise 1

Define a timer that can be started (or resumed), stopped, and reset, and that will raise an alarm after 30 seconds if not stopped or reset.

- $S = \{ \text{stop}_0, \text{stop}_1, \dots, \text{stop}_{30}, \text{run}_0, \text{run}_1, \dots, \text{run}_{30} \}$
- $A = \{ \text{start}, \text{stop}, \text{tick}, \text{ring} \}$
- $S_0 = \{ \text{stop}_0 \}$
- $T = \{$
 $(\text{stop}_{30}, \text{start}, \text{run}_{30}), \dots, (\text{stop}_0, \text{start}, \text{run}_0),$
 $(\text{run}_{30}, \text{tick}, \text{run}_{29}), \dots, (\text{run}_1, \text{tick}, \text{run}_0),$
 $(\text{run}_0, \text{ring}, \text{stop}_0),$
 $(\text{run}_{30}, \text{stop}, \text{stop}_{30}), \dots, (\text{run}_0, \text{stop}, \text{stop}_0),$
 $(\text{run}_{30}, \text{reset}, \text{run}_{30}), \dots, (\text{run}_0, \text{reset}, \text{run}_{30}),$
 $(\text{stop}_{30}, \text{reset}, \text{stop}_{30}), \dots, (\text{stop}_0, \text{reset}, \text{stop}_{30}) \}$



Question 1

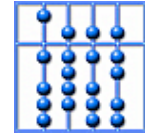
What are the atomic actions of a Pascal/C/Lisp/Java program?

Atomic actions:

- (Conceptually) instantaneous
- Guaranteed effect

Atomic actions are (generally) independent of the programming language but depend on the execution platform (and its notion of interleaving):

- Pascal/C: Atomic instructions of the operating system
- Java: Atomic instructions of the JVM



Question 2

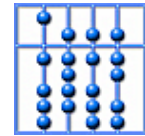
What are the simple interactions of a Java program/thread?

Simple interactions: Joint observation/action at the interface

- Interface of a Java program/thread: Methods
- (Joint) observations/actions: (Shared) objects

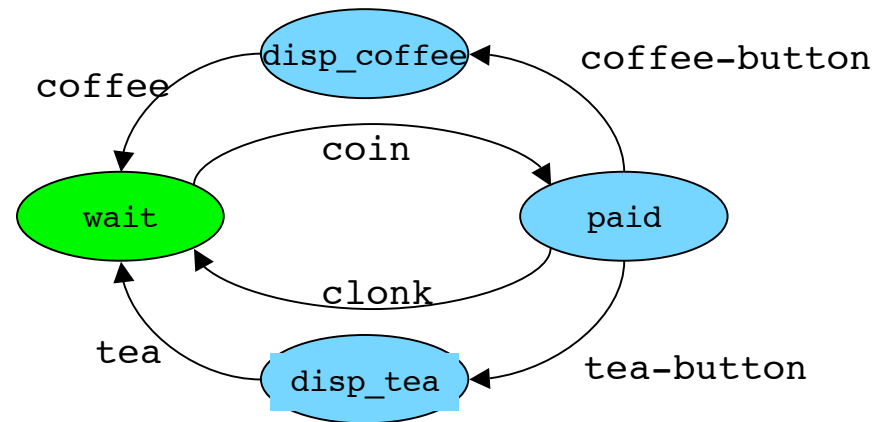
Procedural languages (including most OO languages) do not support synchronization at their interfaces but use synchronization elements:

- Implicit synchronization: Passive objects
- Explicit synchronization: Synchronized passive objects (monitors)



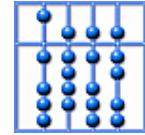
Question 3

Does an input/output distinction make sense in a hand-shake (message-synchronous) communication model?



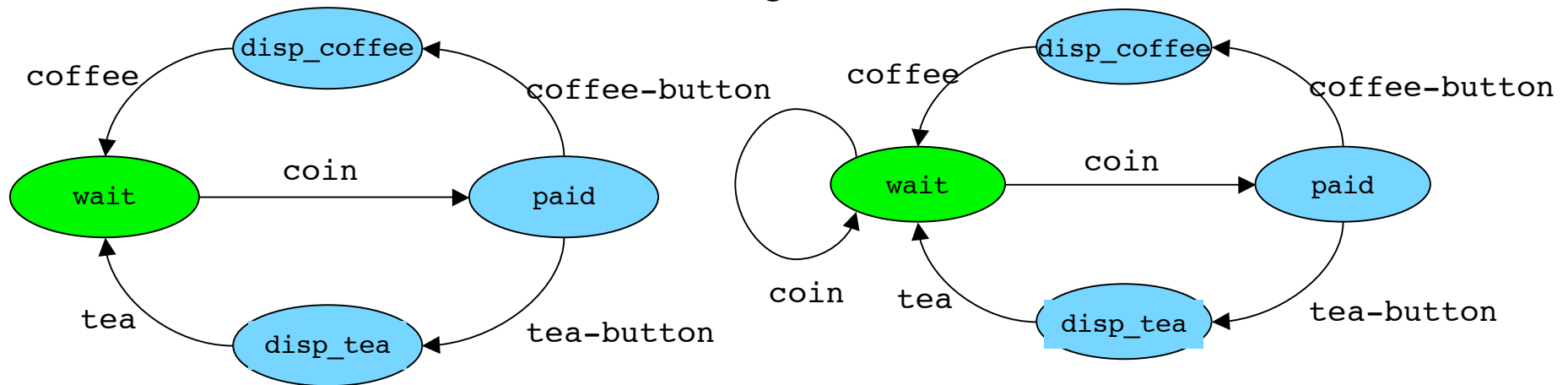
Noisy broken vending machine:

- Intuitively: Input is controlled by the environment, output by the system
 - Inputs: coin, coffee-button, tea-button
 - Outputs: clonk, tea, coffee
- Formally: Interaction is controlled by both parties
 - Compose noisy machine with a customer who accepts tea or coffee, but no clonk
 - No deadlock, but clonk cannot be executed

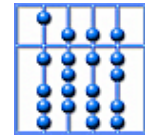


Question 4

What is the relation between the observation traces of the vending machine and the broken vending machine?

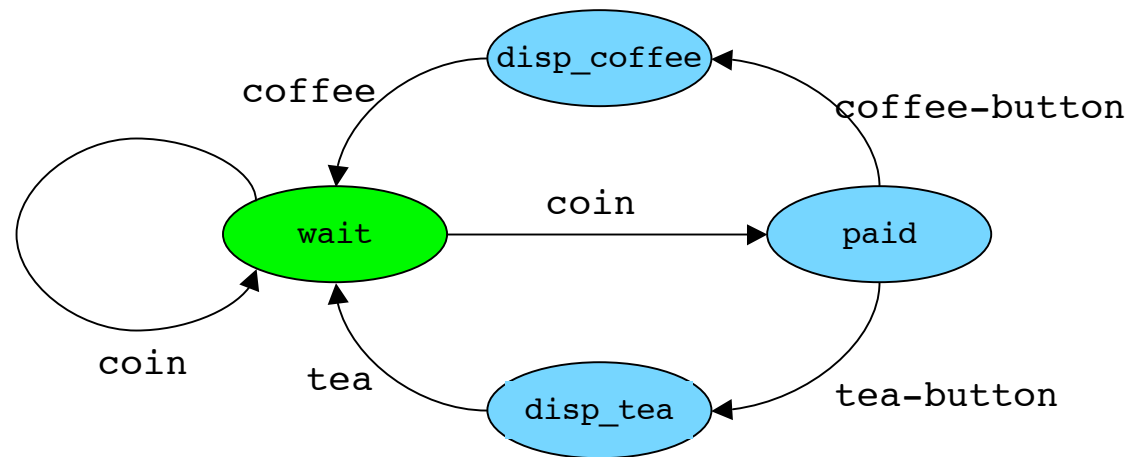


- Execution traces of vending machine LTS1:
(wait • coin • paid • {c-but • di_cof • cof, t-but • di_tea • tea})^{*} and any prefix
- Observation trace of vending machine : (coin • {c-but • cof, t-but • tea})^{*} and any prefix
- Execution traces of broken vending machine LTS2:
(wait • (coin • wait)^{*} • coin • paid • {c-but • di_cof • cof, t-but • di_tea • tea})^{*} and any prefix
- Observation trace: (coin • coin^{*} • {c-but • cof, t-but • tea})^{*} and any prefix
- Relation: Traces(LTS1) ⊆ Traces(LTS2)

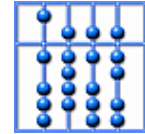


Question 5

Is there a general principle behind the answer of question 4?

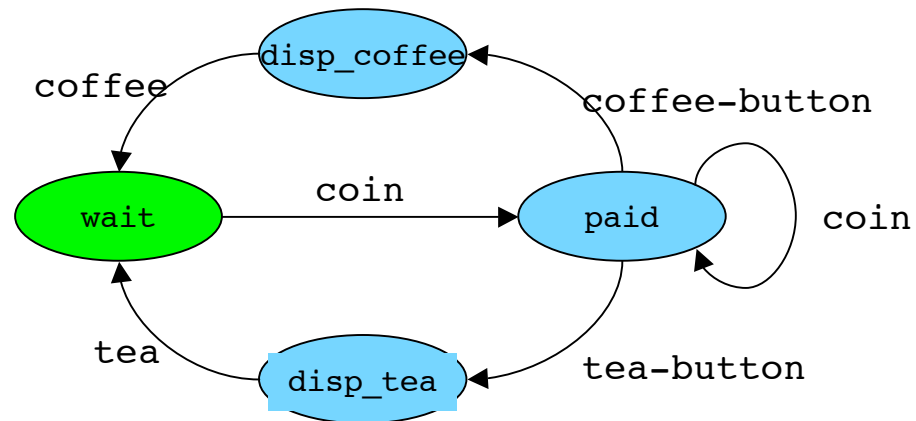


- According to our definition of (system-controlled) nondeterminism: Additional nondeterminism is caused by a branching transition with an identical label
- Therefore: Additional execution/observation trace starting at this branch
- Relation between a LTS1 with less nondeterminism than LTS2: $\text{Traces}(\text{LTS1}) \subseteq \text{Traces}(\text{LTS2})$
- Remark: Does the converse hold? What about choices?



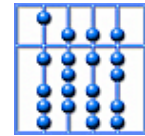
Question 6

Is there a different machine with the same observation traces as the broken vending machine?



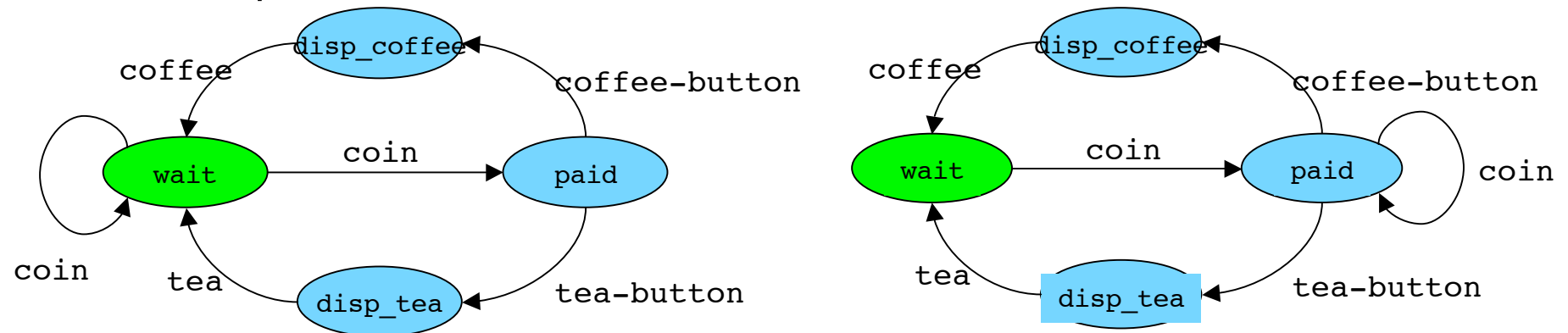
Strange vending machine:

- Execution traces of this LTS:
 $(\text{wait} \cdot \text{coin} \cdot \text{paid} \cdot (\text{coin} \cdot \text{paid})^* \cdot \{\text{c-but} \cdot \text{di_cof} \cdot \text{cof}, \text{t-but} \cdot \text{di_tea} \cdot \text{tea}\})^*$ as well as any prefix thereof
- Observation trace: $(\text{coin} \cdot \text{coin}^* \cdot \{\text{c-but} \cdot \text{cof}, \text{t-but} \cdot \text{tea}\})^*$ as well as any prefix thereof

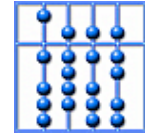


Question 7

If so, is one of them more “benign” than the other, as seen from the user’s point of view?

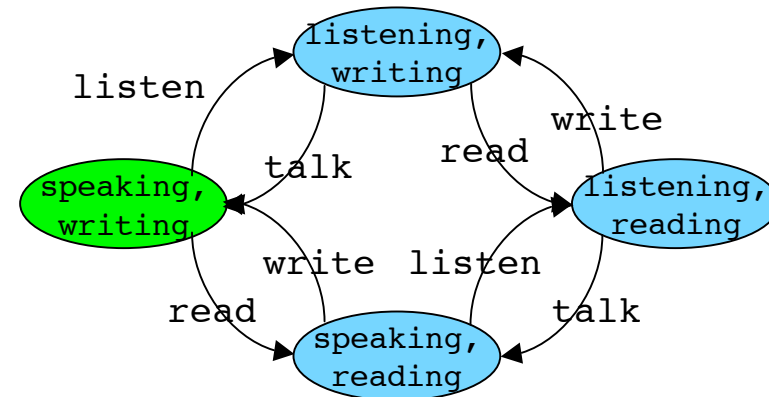
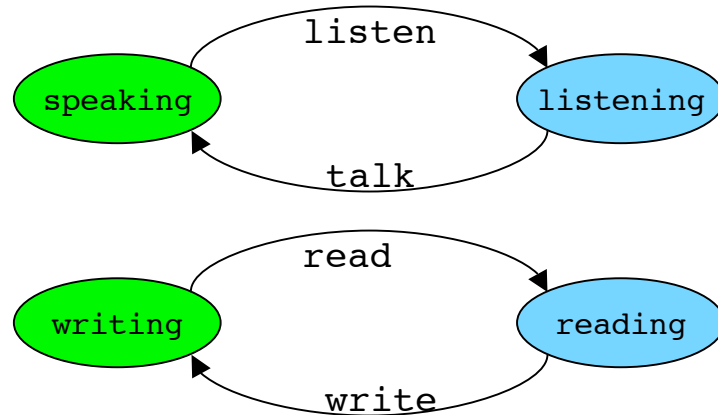


- Difference of strange to broken vending machine
 - Broken vending machine: Outcome of inserting the coin is controlled by the system (either state wait or state paid)
 - Strange vending machine: Coin and button are controlled by the user of the vending machine
- Therefore: user of strange vending machine can choose to enter more coins, but is not obliged to do so



Question 8

Can you tell - by looking at the LTS as introduced in 2.2 - whether a system is parallel or sequential?

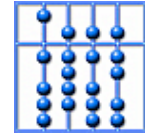


Parallel composition:

- Transforms two LTS into a single LTS
- Uses arbitrary sequentialization ('interleaving') for independent interactions, joint sequentialization for shared interactions

Therefore: We cannot distinguish whether LTS was obtained through parallel composition

Especially: $LTS \parallel LTS = LTS$



Question 9

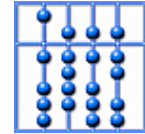
What must be added to the model to distinguish between parallel and concurrent systems?

Restriction: The standard LTS allows only one (atomic) interaction per transition

Extension: Allow complex interactions per transition

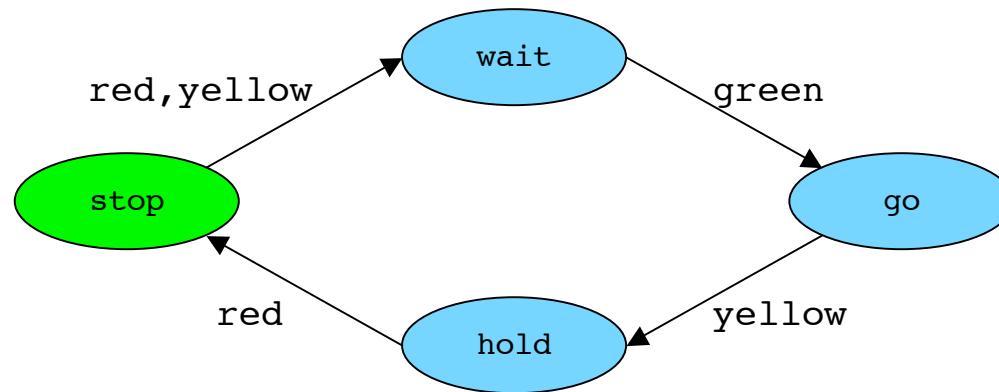
A possible formalization: Labeled Transition System (S, A, S_0, T)

- 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 $T \subseteq S \times 2^A \times S$



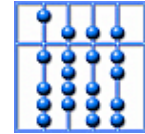
Question 10

How does such a modified model affect the specification of the traffic light?



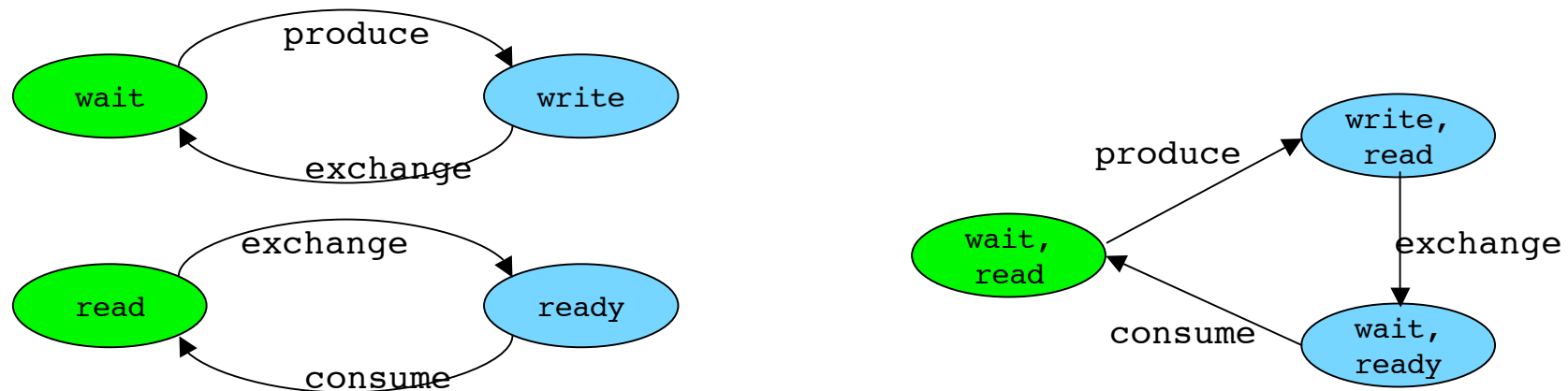
Modified Model:

- $S = \{ \text{stop, wait, go, hold} \}$
- $A = \{ \text{green, yellow, red} \}$
- $S_0 = \{ \text{stop} \}$
- $T = \{ (\text{stop}, \{ \text{red, yellow} \}, \text{wait}), (\text{wait}, \{ \text{green} \}, \text{go}), (\text{go}, \{ \text{yellow} \}, \text{hold}), (\text{hold}, \{ \text{red} \}, \text{stop}) \}$

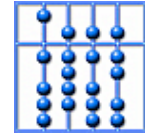


Question 11

How can a trace of a subsystem be obtained from the trace of a composed system?

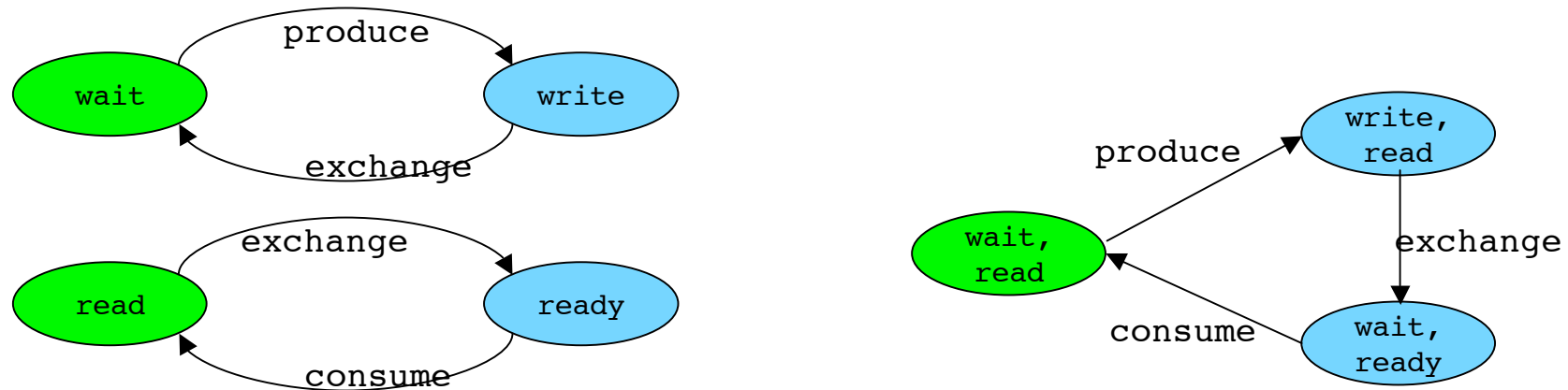


- Observation Traces:
 - Producer: $(\text{produce} \cdot \text{exchange})^*$ and any prefix
 - Consumer: $(\text{exchange} \cdot \text{consume})^*$ and any prefix
 - System: $(\text{produce} \cdot \text{exchange} \cdot \text{consume})^*$ and any prefix
- Obtaining the traces:
 - Producer: remove all “consume” actions from the system observations
 - Consumer: remove all “produce” actions from the system observations

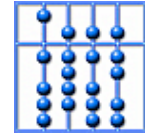


Question 11 (continued)

How can a trace of a subsystem be obtained from the trace of a composed system?

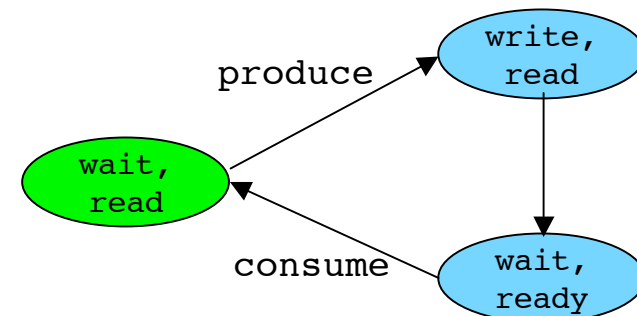
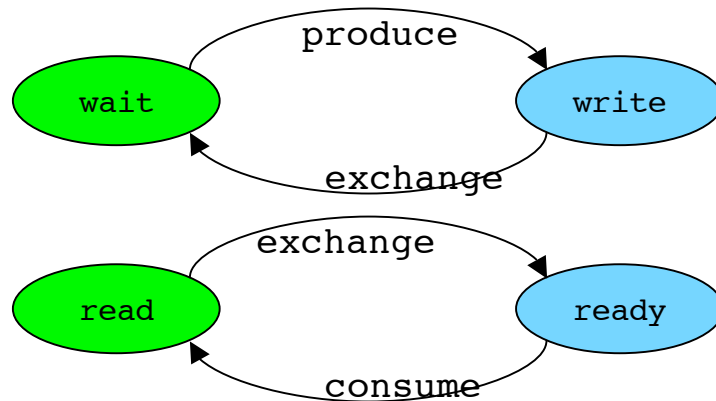


- Execution Traces:
 - Producer: (wait • produce • write • exchange)* and any prefix
 - Consumer: (read • exchange • ready • consume)* and any prefix
 - System: ((wait,read) • produce • (write,read) • exchange • (write,ready) • consume)* and any prefix
- Obtaining the traces:
 - Producer: Remove “consume” • state pairs, remove second elements of states
 - Producer: Remove “produce” • state pairs, remove second elements of states



Question 12

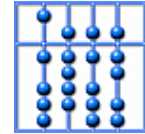
Can you tell - by looking at its traces - whether a system is parallel or sequential?



As discussed in Question 8:

- We cannot distinguish whether LTS was obtained through parallel composition
- Special example: $LTS \parallel LTS = LTS$
- Since Traces can be constructed from an LTS, this is already a counter example

Additional example: Both LTS generate the trace set (produce • exchange • consume)* and any prefix



Question 13

What must be added to obtain a more general model of traces, distinguishing between interleaved and parallel behavior?

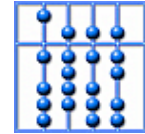
Restriction: The standard definition of a trace allows only one (atomic) interaction per time instance

Extension: Allow complex interactions per time instance

A (simple) possible formalization: Traces of $(2^A)^*$

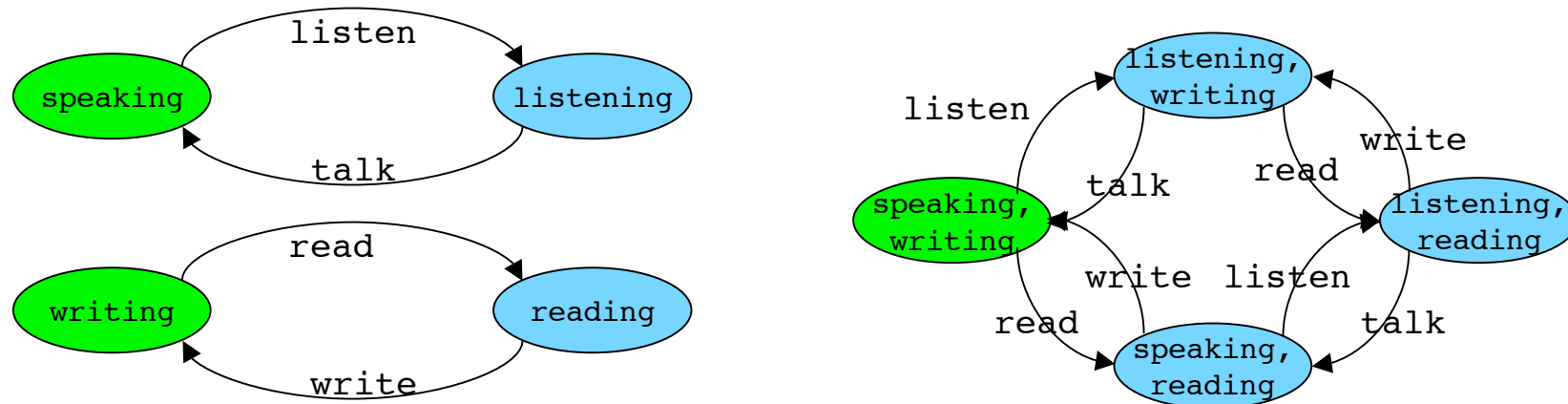
Example: $\{\text{red,yellow}\} \cdot \{\text{green}\} \cdot \{\text{yellow}\} \cdot \{\text{red}\} \cdot \{\text{red,yellow}\} \cdot \dots$

More general: Partial orders of interactions



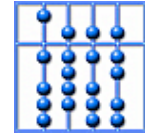
Question 14

What is the relationship between choice and parallelism concerning the model of LTS as introduced in 2.2?



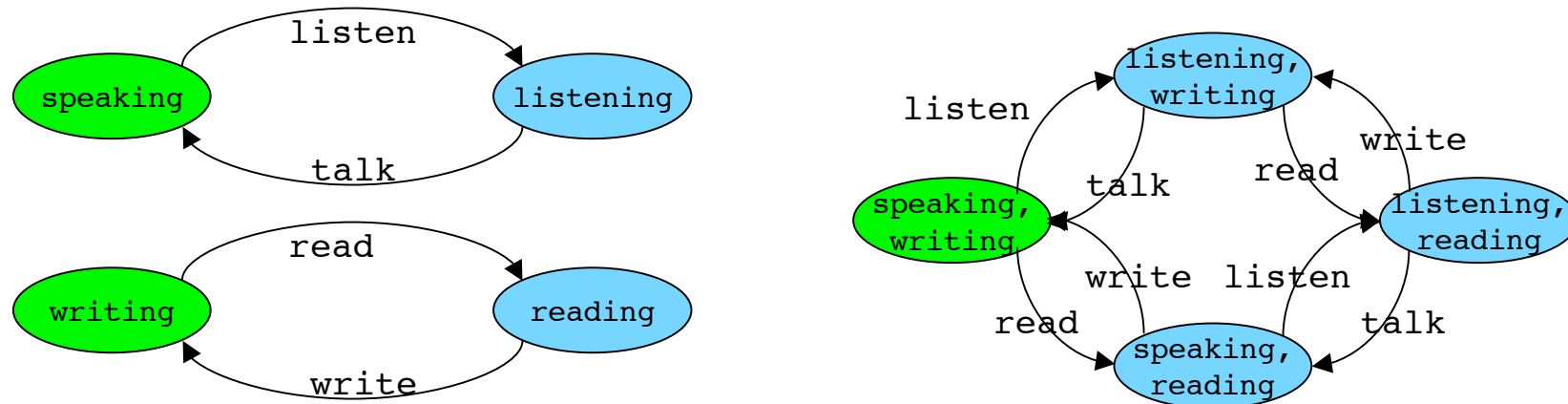
Relation: Independent parallelism can be substituted by arbitrary sequentialization and vice versa

- Independent parallelism: $(\text{speaking} \xrightarrow{\text{listen}} \text{listening}) \parallel (\text{writing} \xrightarrow{\text{read}} \text{reading})$
- Arbitrary sequentialization:
 $(\text{speaking, writing}) \xrightarrow{\text{listen}} (\text{listening, writing}), (\text{speaking, writing}) \xrightarrow{\text{read}} (\text{speaking, reading}),$
 $(\text{listening, writing}) \xrightarrow{\text{read}} (\text{listening, reading}), (\text{speaking, reading}) \xrightarrow{\text{listen}} (\text{listening, reading}),$



Question 15

What is relation between the 'execution speed' of two independent parallel processes?



Asynchronous synchronization: Interleaving of actions

- Arbitrary sequentialization: Either process may proceed at each step
- Independent parallelism: Only one process may proceed at each step
- Unfair execution: infinite takeover of one process

$(\text{speaking, writing}) \xrightarrow{\text{listen}} (\text{listening, writing}) \xrightarrow{\text{talk}} (\text{speaking, writing}) \xrightarrow{\text{listen}} (\text{listening, writing}) \xrightarrow{\text{talk}} \dots$