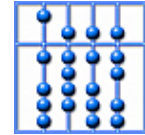


Specification of Distributed Systems

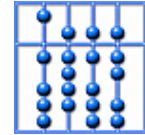
Dr. Bernhard Schätz
Leopold-Franzens Universität Innsbruck
Sommersemester 2005



Overview

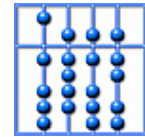
1. Introduction
2. Basics: Behavior, Interaction, Concurrency
3. Coroutines
4. Communicating Processes
5. Data Flow Models
6. Coordination
7. Executions
8. State-Based Model
9. Property Descriptions

10. Implementation

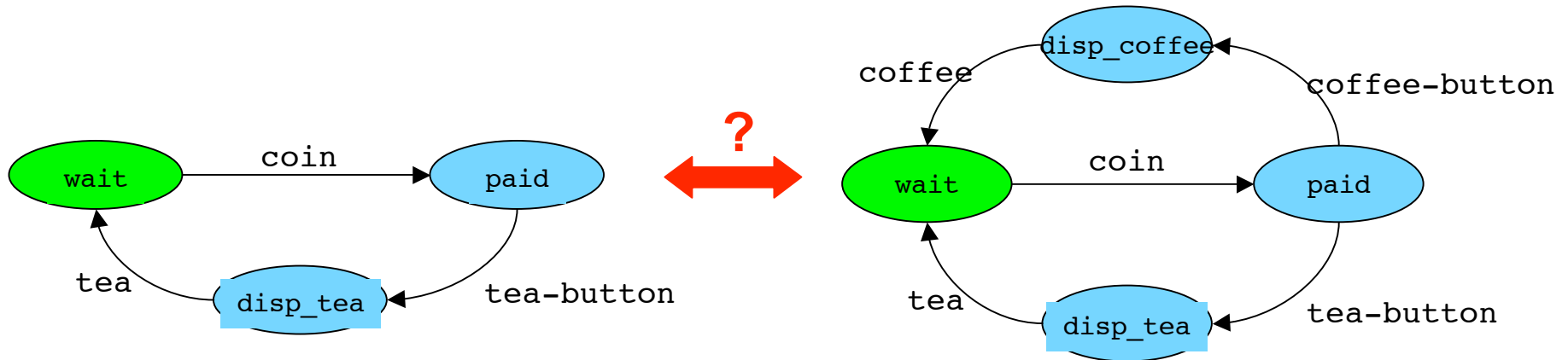


Overview

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

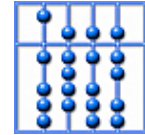


Motivation: Implementation



Implementation relation: Relating abstract and concrete system

- Concrete system respects properties of abstract system
- Concrete system resolves open issues of abstract system



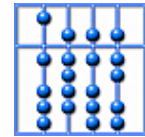
10.1 Implementation: Operational Verification

Goal: Define a technique to verify the implementation relation of systems

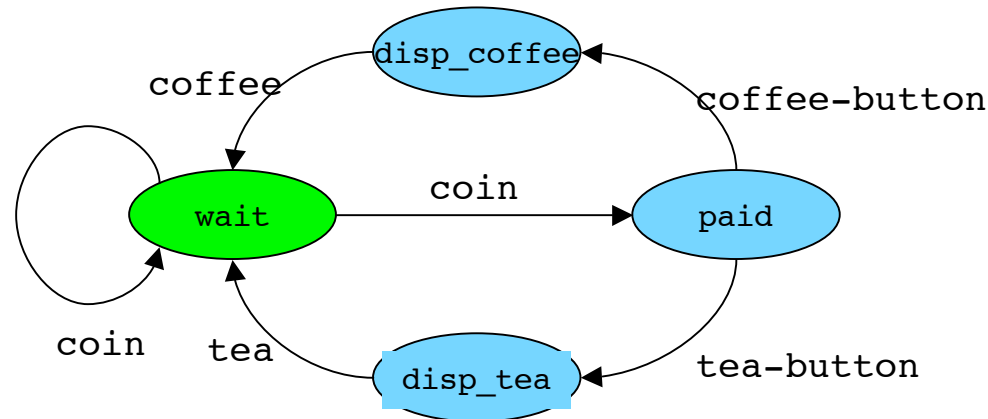
- Relevant: Address aspects concerning offered choices
- General: Cover aspects of independent of specific form of implementation
- Abstract: Ignore aspects like improvement of choices

Concept: Simulation

Model: Labeled Transition Systems



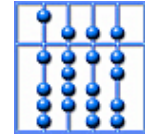
Recap: Modeling Reactivity



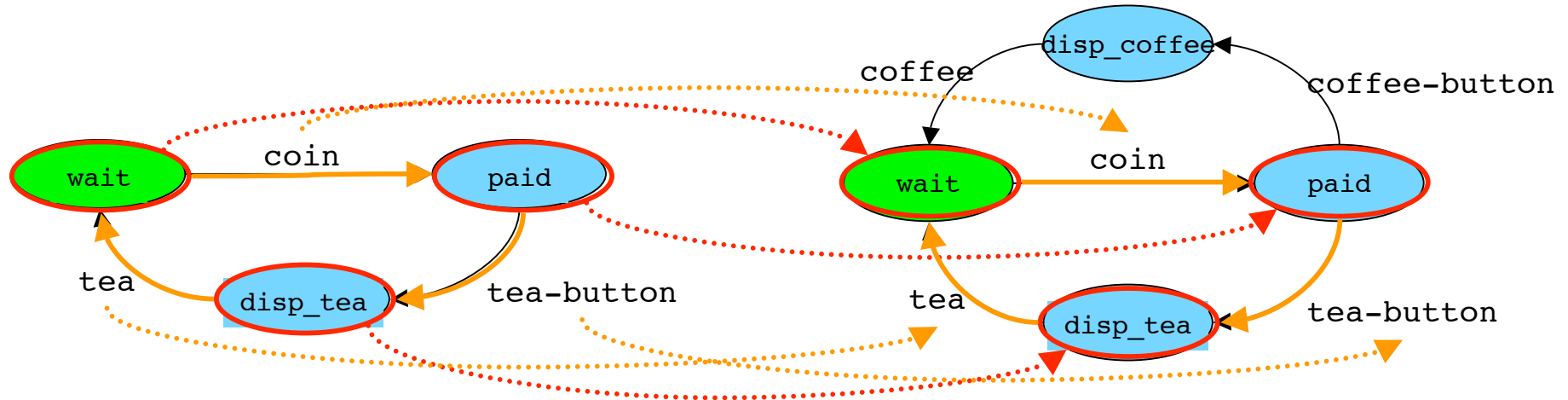
Concepts:

- 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)



Concept: Simulation

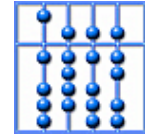


Purpose: Relate states of the system offering the same choice of interactions

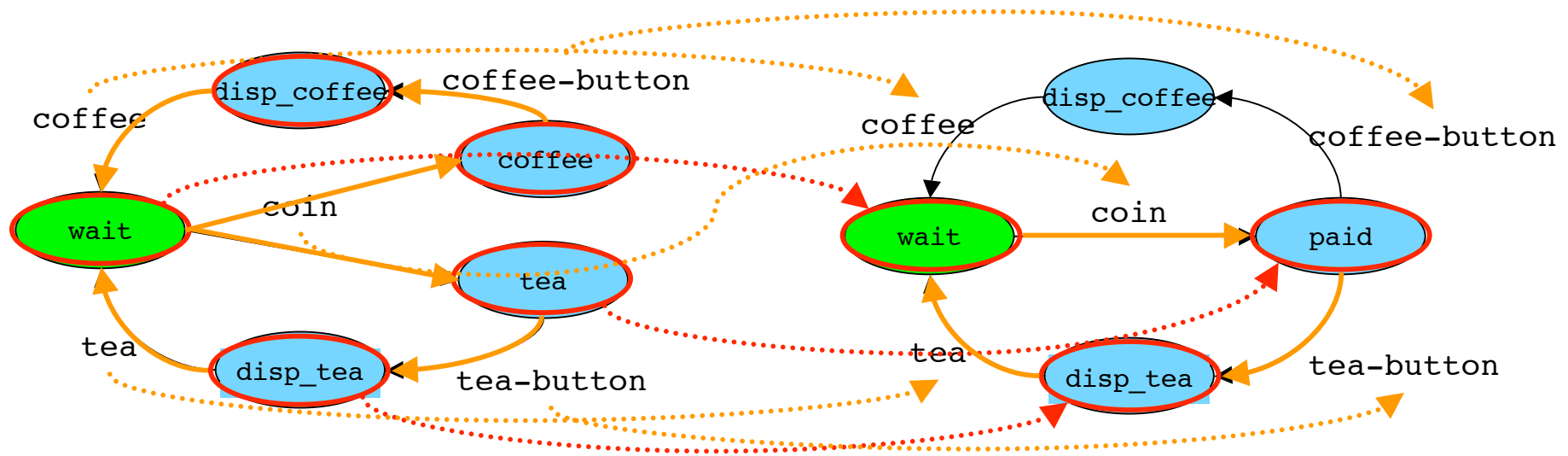
Concept: Simulation relation R between systems (S, A, S_0, T) and (S', A, S'_0, T')

- Relates abstract and concrete labeled transition systems with same alphabet
- Relates respective states $R \subseteq S \times S'$
- Ensures compatibility of offered choices in related states

Example: $R = \{ (wait, wait'), (paid, paid'), (disp_tea, disp_tea) \}$



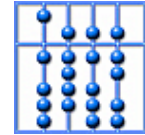
Definition: Simulation



Definition: Simulation relation R between systems (S, A, S_0, T) and (S', A, S'_0, T')

- Relates respective states $R \subseteq S \times S'$
- Ensures compatibility of offered choices in related states:
 - If: (s, s') in R and (s, a, t) in T
 - Then: some t' with (s', a, t') in T' and (t, t') in R
- Relates systems:
 - (s, s') in R for all s in S_0 and s' in S'_0

Example: $R = \{ (wait, wait'), (coffee, paid'), (tea, paid'), (disp_tea, disp_tea) \}$



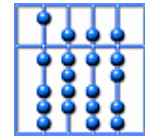
10.2 Implementation: Algebraic Verification

Goal: Define a technique to verify the equivalence relation of systems

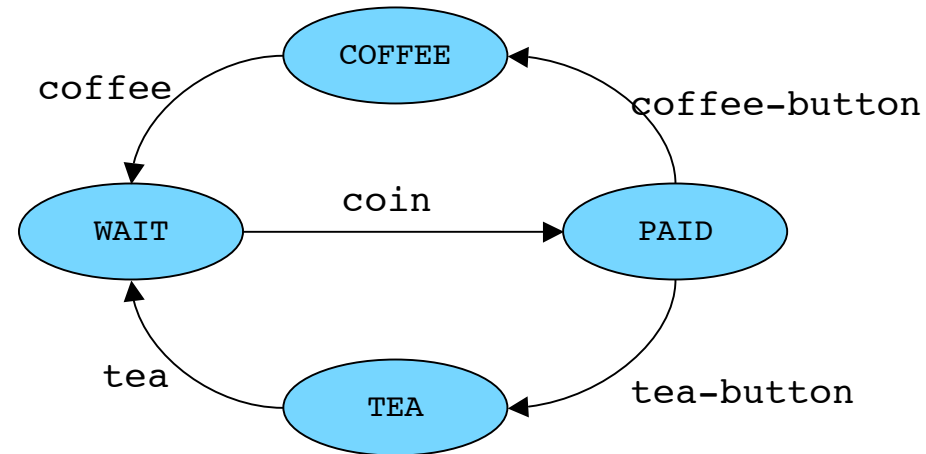
- Relevant: Address aspects concerning possible choices
- General: Cover aspects of independent of form of implementation
- Abstract: Ignore aspects like improvements of choices

Concept: Equivalence

Model: Process terms

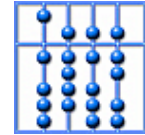


Recap: Process Terms



Concepts:

- Process term: Term describing structure of process
- Prefix: $a \rightarrow P$
- Choice: $P \mid Q$
- Condition: $c \triangleright P$
- Parallel Composition: $P \parallel Q$
- Hiding: $P \setminus H$



Concept: Equivalence

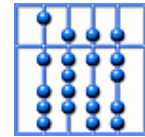
Purpose: Relate process terms exhibiting the same behavior

Concept: Equivalence $P = Q$

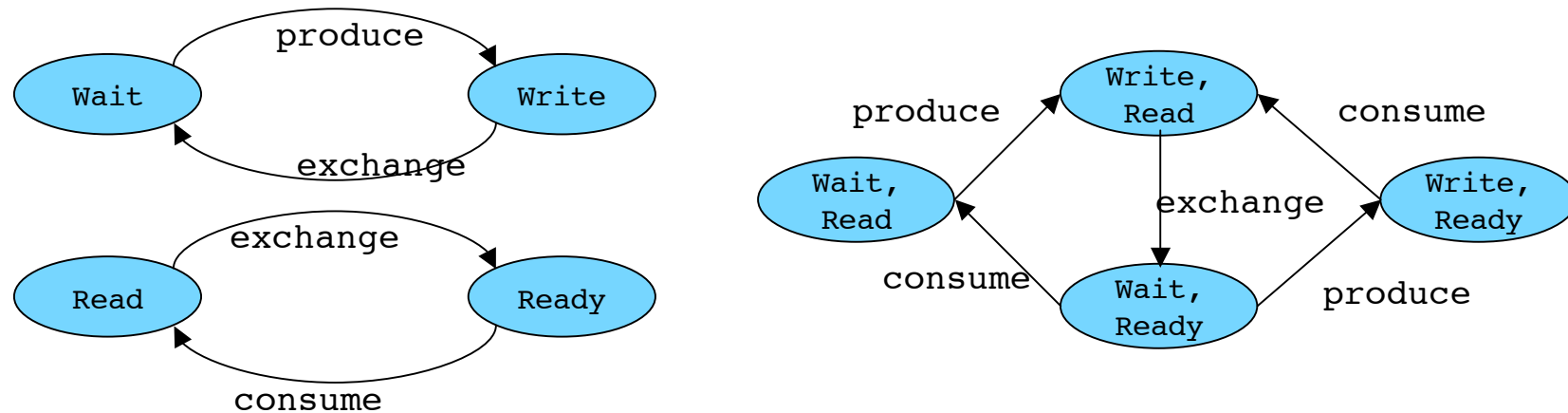
- Relates process terms P and Q
- Allows unfolding of process term
- Supports equational reasoning over structure of process terms

Examples:

- $P \mid \text{STOP} = P$
- $\text{false} \triangleright P = \text{STOP}$
- $P \parallel Q = Q \parallel P$
- $P \parallel (Q \parallel R) = (P \parallel Q) \parallel R$
- $P \parallel \text{STOP}_A = \text{STOP}_A$ (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)
- $\text{STOP}_A \setminus H = \text{STOP}_{A \setminus H}$
- $(a \rightarrow P) \setminus \{ a \} = P \setminus \{ a \}$



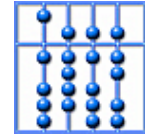
Concept: Equational Reasoning



Purpose: Establish equivalence of process terms

Concept: Equational Reasoning

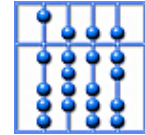
- Basic equivalences: Application of basic equivalence relations
- Stepwise reasoning: $P = Q, Q = R \Rightarrow P = R$
- Inductive reasoning: $P = F(P) \Rightarrow P = \mu Q. F(Q)$



Example: Inductive Reasoning

- $M = \mu P. Mch(P)$
- $M(P) = coin \rightarrow ((tea-button \rightarrow tea \rightarrow P) \mid (coffee-button \rightarrow coffee \rightarrow P))$
- $C = \mu Q. Cst(Q)$
- $Cst(Q) = coin \rightarrow tea-button \rightarrow tea \rightarrow Q$

- Inductive reasoning:
 - Stepwise reasoning: $M \parallel C$
 - = $\mu P. Mch(M) \parallel \mu Q. Cst(Q)$
 - = $Cst(\mu P. Mch(P)) \parallel Cst(\mu Q. Mch(Q))$
 - = $Mch(M) \parallel Cst(C)$
 - = $(coin \rightarrow ((tea-button \rightarrow tea \rightarrow M) \mid (coffee-button \rightarrow coffee \rightarrow M)))$
 - = $coin \rightarrow (((tea-button \rightarrow tea \rightarrow M) \mid (coffee-button \rightarrow coffee \rightarrow M)))$
 - = $coin \rightarrow tea-button \rightarrow ((tea \rightarrow M) \parallel (tea \rightarrow C)) =$
 - = $coin \rightarrow tea-button \rightarrow tea \rightarrow (M \parallel C)$
 - Inductive reasoning: $M \parallel C = \mu Q. Cst(Q)$
 - Stepwise reasoning: $M \parallel C = C$



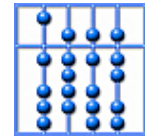
10.3 Implementation: Denotational Verification

Goal: Define a technique to verify the implementation relation of systems

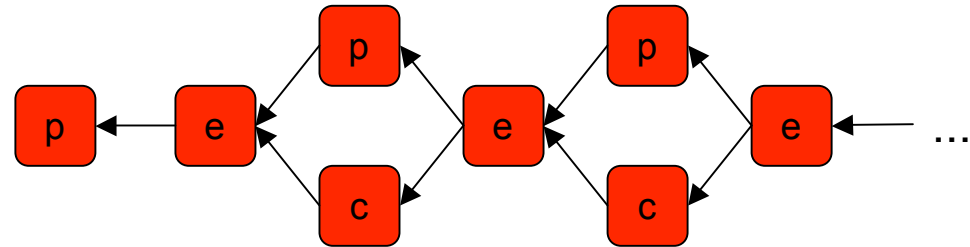
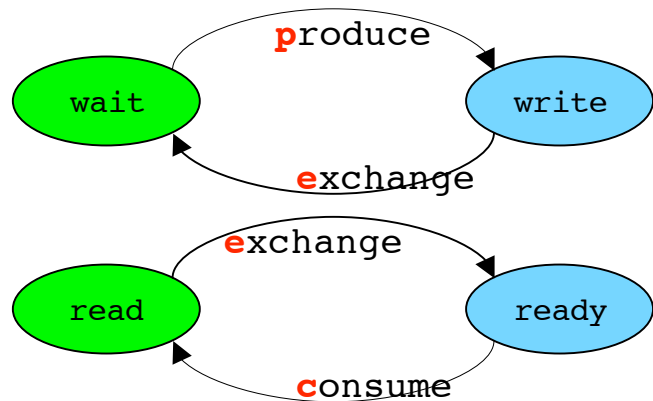
- Relevant: Address aspects concerning possible behaviors
- General: Cover aspects of independent of form of implementation
- Abstract: Ignore aspects like improvements of behaviors

Concept: Implication

Model: Behaviors



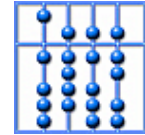
Recap: Event Structures



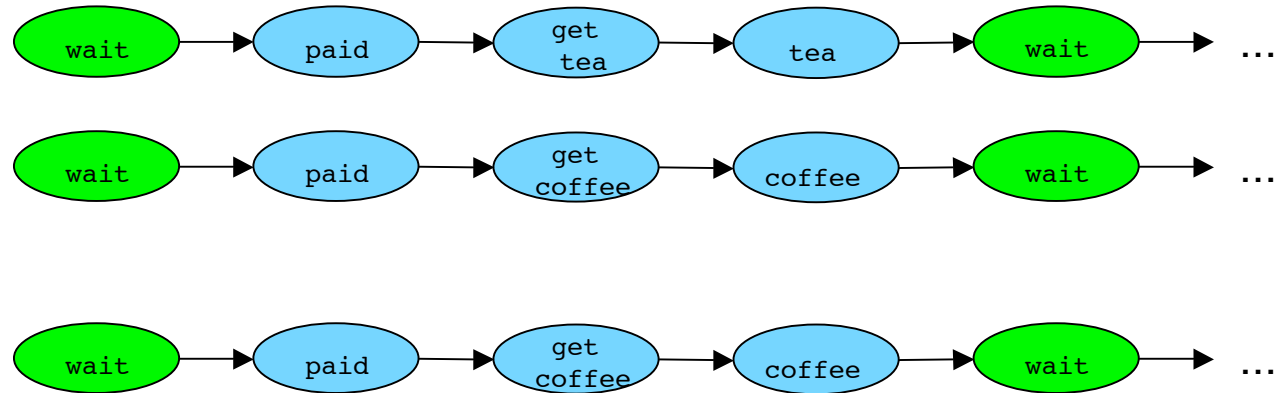
Concepts:

- Interaction: Synchronization between processes
- Labeled event: Occurrence of an interaction
- Causal order: Dependency in order of occurring interactions
- Event trace: Structure of labeled events ordered by causal order
- Behavior: Set of event traces of a system

Model: Event structures



Concept: Refinement

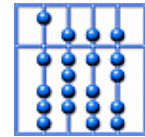


Purpose: Relate sets of possible behaviors of systems

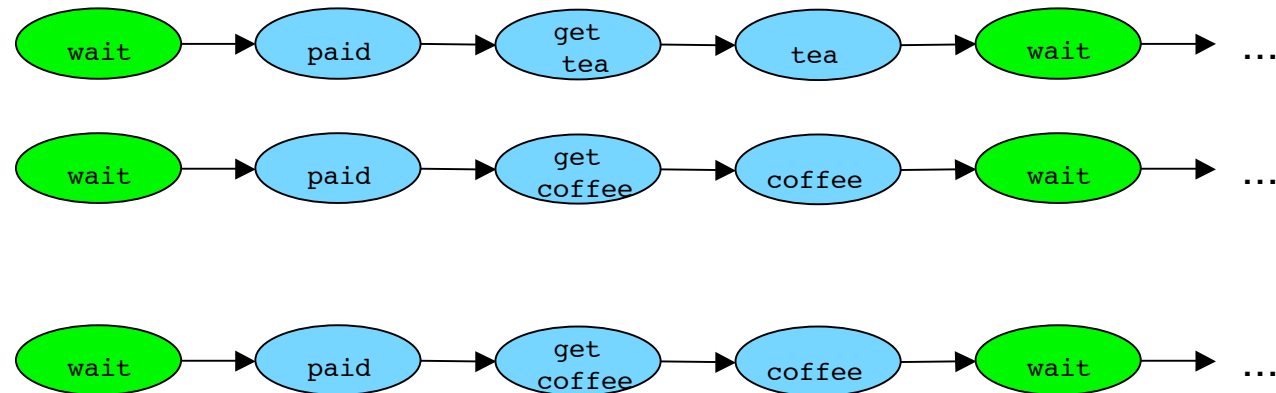
Concept: Refinement between abstract and concrete systems

- Relates sets of behaviors with same alphabet
- Models: (Timed, untimed, structured) observations traces, event structures
- Ensures that concrete system is “more deterministic” than abstract system

Example: $R = \{\text{coin} \cdot \text{tea-button} \cdot \text{tea} \cdot \text{coin} \cdot \dots, \text{coin} \cdot \text{coffee-button} \cdot \text{coffee} \cdot \text{coin} \cdot \dots\} \subseteq \{\text{coin} \cdot \text{coffee-button} \cdot \text{coffee} \cdot \text{coin} \cdot \dots\}$



Definition: Refinement



Purpose: Relate sets of possible behaviors of systems

Definition: Refinement between abstract and concrete systems

- Intuition: Abstract system is more non-deterministic as concrete system
- Model-level: Behavior of concrete system \subseteq behavior of abstract systems
 - Message-asynchronous systems: Traces, event structures
 - Message-synchronous systems: Failure sets
- Specification level: $\text{Spec}_{\text{abstract}} \leftarrow \text{Spec}_{\text{concrete}}$