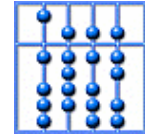


Specification of Distributed Systems

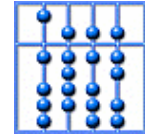
Dr. Bernhard Schätz
Leopold-Franzens Universität Innsbruck
Sommersemester 2005



Overview

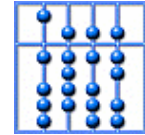
1. Introduction
2. Basics: Behavior, Interaction, Concurrency
3. Coroutines
4. Communicating Processes
5. Data Flow Models
6. Coordination
7. Executions
8. State-Based Model

9. Property Descriptions



Overview

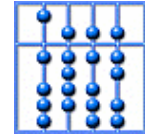
1. Introduction
2. Basics: Behavior, Interaction, Concurrency
3. Coroutines
4. Communicating Processes
5. Data Flow Models
6. Coordination
7. Executions
8. State-Based Model
9. Property Descriptions
 1. Temporal Properties
 2. Safety and Liveness Properties



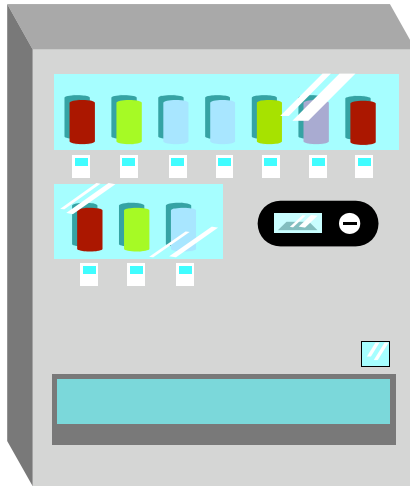
Specifications

Classes of behavioral specifications:

- **Constructive:** Specification of an (abstract) implementation
 - Purpose: Description capable of generating all possible observations about the behavior of the system
 - Models: e.g., Transitions systems, Kripke structures, process algebras
 - Notations: Transition Diagrams, process terms
- **Descriptive:** Specification in form of observable behavior
 - Purpose: Description of required/allowed/forbidden observations about behavior of the system
 - Models: e.g., Traces, event structures
 - Notations: e.g., Sequential interaction diagrams, temporal predicates



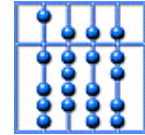
Motivation: Specifying Temporal Properties



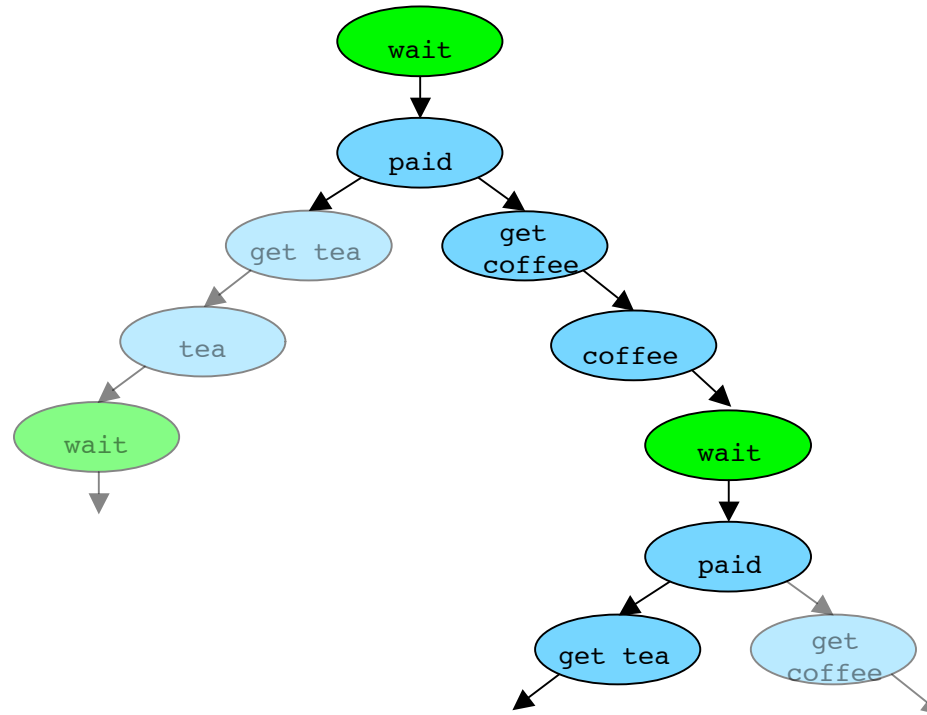
If at some point in time a coin is in the coin slot
and at the next moment the tea button is pressed
then eventually tea will be supplied in the dispenser slot

Propositions over of state-based systems:

- Propositions over states
- Propositions over executions



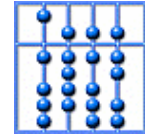
Recap: Execution



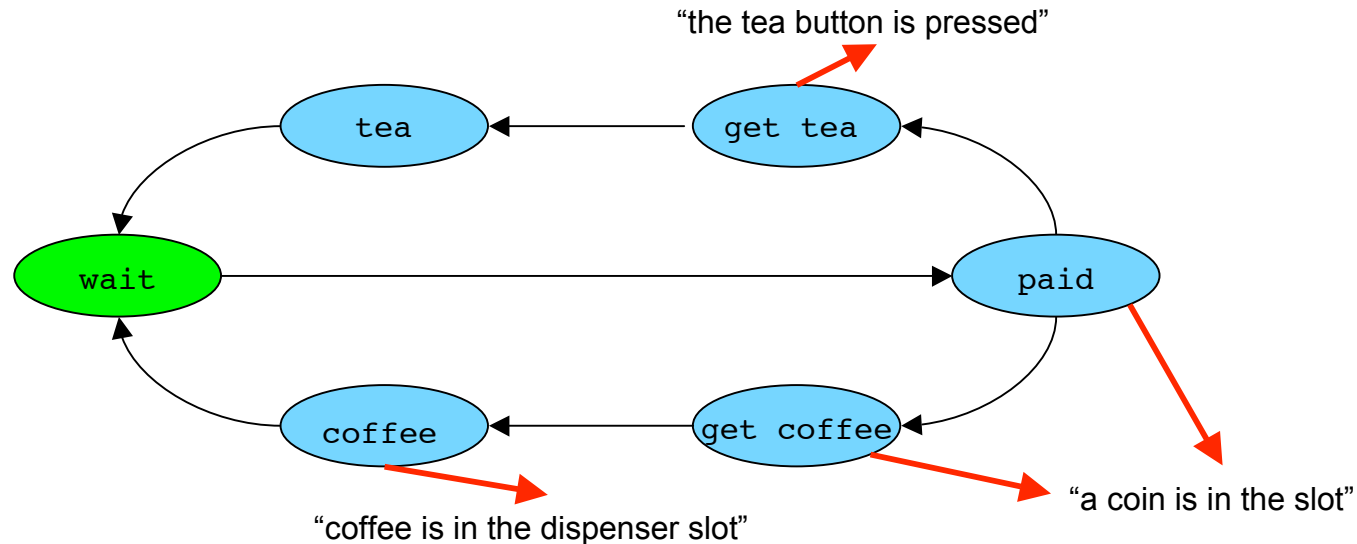
Purpose: Unfolding of state transitions of a system

Concept: Executions

- Tree: Branches of alternative transitions of the system
- Sequence: Single path of execution of a system



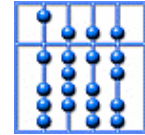
Recap: Kripke Structure



Purpose: Model the change between valid propositions through the changes of a system

Model: Kripke structure (S, S_0, T, P, O)

- S : States of the system
- S_0 : Initial states
- T : Transition relations
- P : Propositions
- O : Observation relations



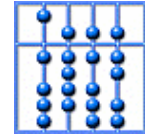
9.1 Describing Properties: Temporal Logics

Goal: Define a notation to describe a observations over complete executions of systems

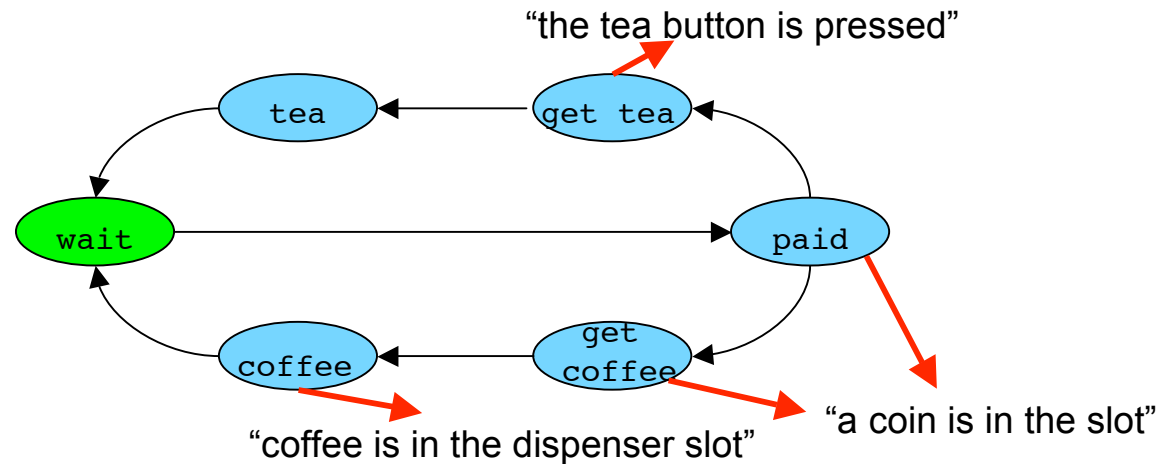
- Relevant: Address aspects evolving with unfolding in time
- General: Cover aspects of independent of specific model of computation
- Abstract: Ignore aspects like durations of execution steps or activities

Concept: Observation, Timed Proposition, Execution (Path)

Notation: Linear Temporal Logic, Computation Tree Logic



Concept: Atomic Properties

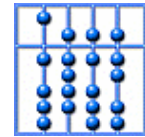


Purpose: Describe *properties valid at a specific point of time*

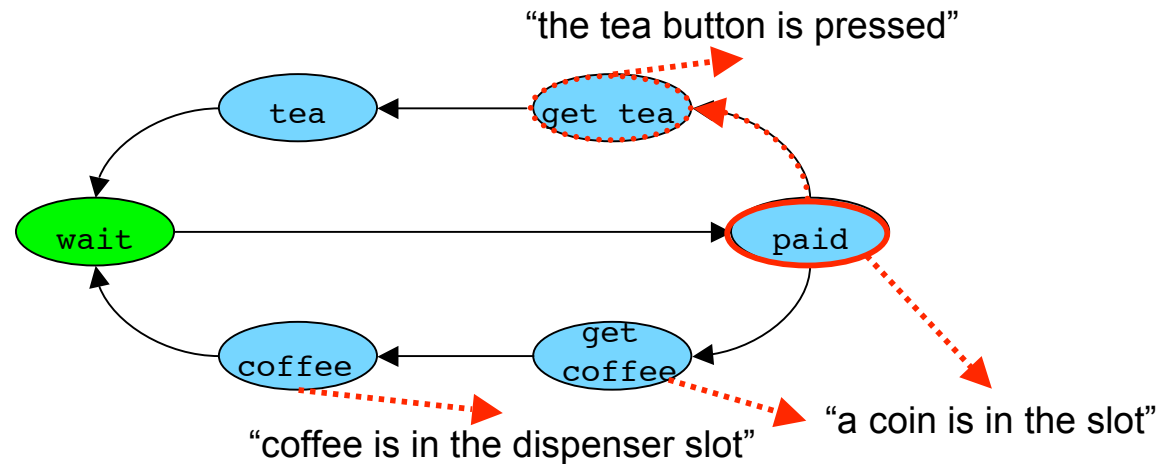
Concept: Atomic Property (Current State Property)

- Atomic: Contains no sub-property
- Observable: Either true or false
- Instant: References current point of time

Example: “Coin is in the coin slot”, “Tea is in the dispenser slot”



Concept: Next Property

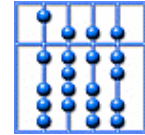


Purpose: Describe *property holding at the next point in time*

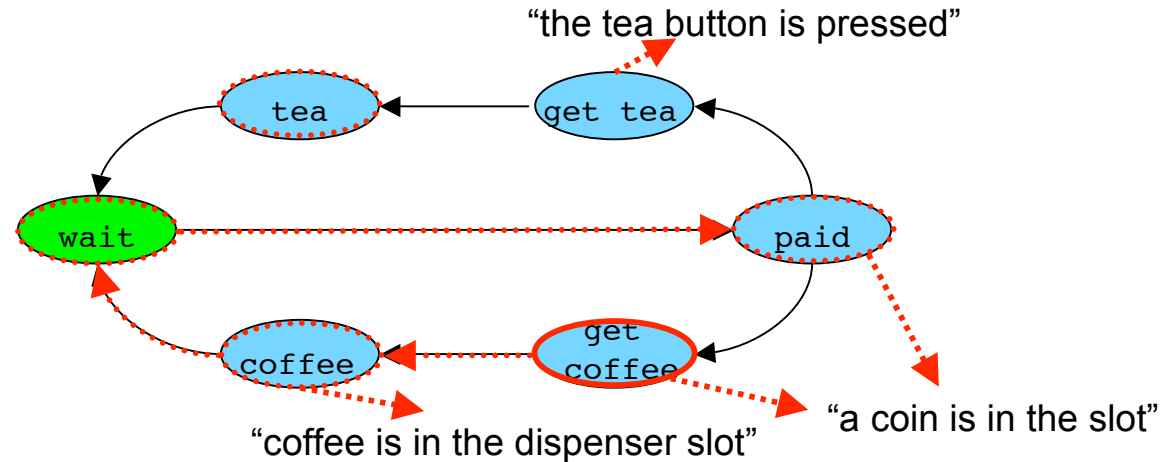
Concept: Next State Property

- Notation: $\bigcirc P$ ("Circle P"), $X P$ ("Next P")
- Interpretation: At the next point in time

Example: "The tea is pressed at the next step" \bigcirc "The tea button is pressed"



Concept: Future Property

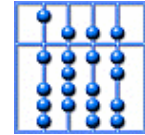


Purpose: Describe a *property that will eventually hold a future point in time*

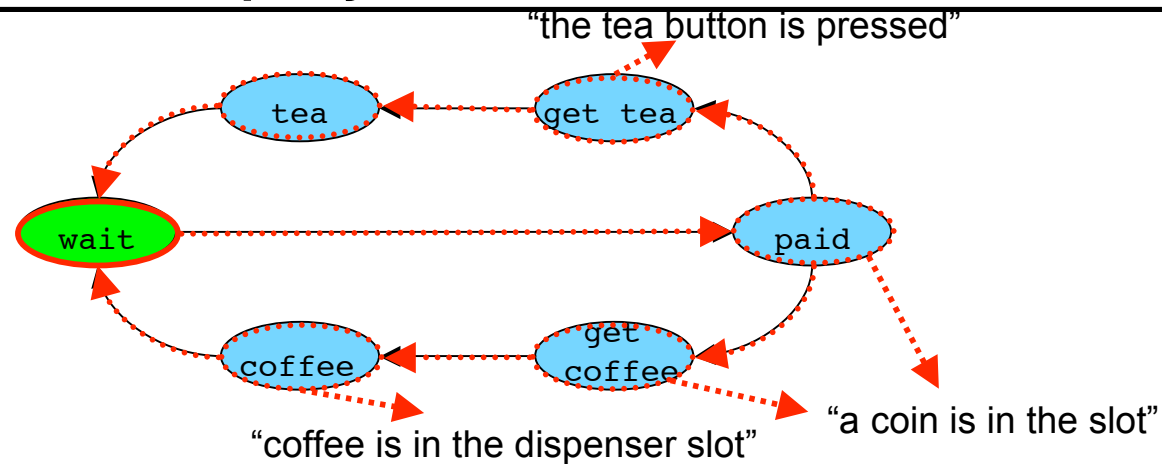
Concept: Future State Property

- Notation: $\diamond P$ ("Diamond P"), $F P$ ("Finally P")
- Interpretation: At some point in time after the current time point, P will hold

Example: "A coin is eventually inserted", \diamond "Coin is in the coin slot"



Concept: Global Property

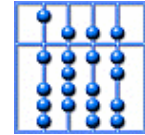


Purpose: Describe *property holding for all future points in time*

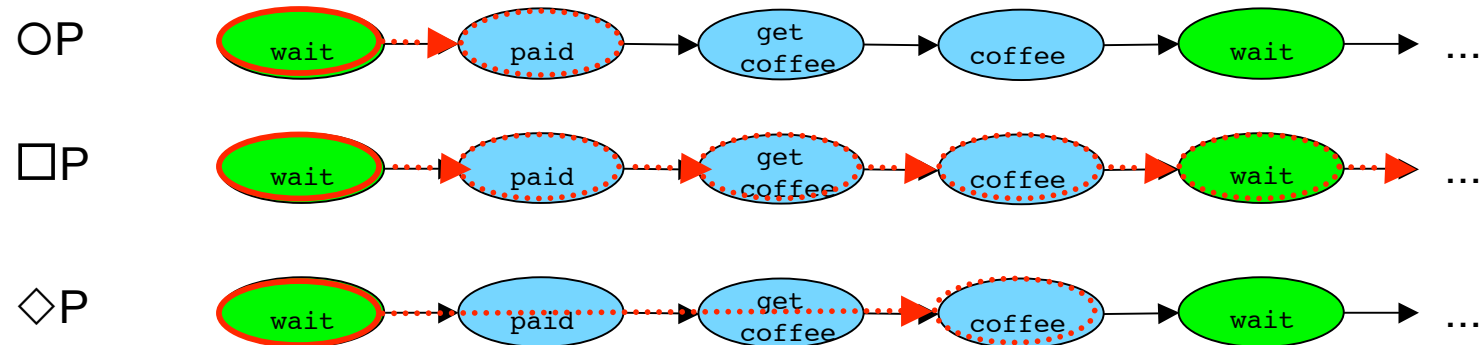
Concept: Global State Property

- Notations: $\Box P$ ("Box P"), or $G P$ ("Globally P")
- Interpretation: Starting at the current time point, P always holds

Example: "Tea is always dispensed next to selecting tea", \Box ("Tea button pressed" \Rightarrow \Diamond "Tea in dispenser")



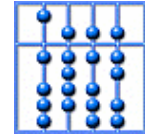
Concept: Linear Time Property



Purpose: Describe *properties holding for a complete execution*

Concept: Linear Time Property

- Model: Sequential execution $E = s_0 \cdot s_1 \cdot s_2 \cdot s_3 \cdot \dots$ of a Kripke structure
- Interpretation: Property holds for execution E
 - P holds for E : $P \in O(s_0)$
 - OP holds for E : P holds for execution $s_1 \cdot s_2 \cdot s_3 \cdot \dots$
 - $\square P$ holds for E : P holds for execution $s_i \cdot s_{i+1} \cdot s_{i+2} \cdot \dots$ for all $i \geq 0$
 - $\diamond P$ holds for E : P holds for execution $s_i \cdot s_{i+1} \cdot s_{i+2} \cdot \dots$ for some $i \geq 0$
- Validity of P : Property holds for all possible executions starting in initial state



Notation: LTL

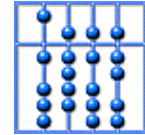
Purpose: Provide a notation *to describe observations about execution sequences*

Notation: Linear Time Logics (LTL) propositions P, Q

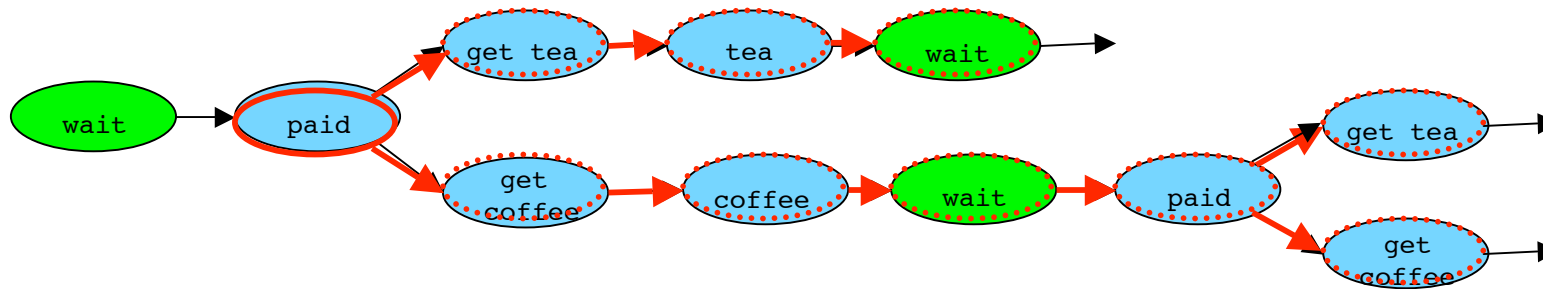
- Basic formulae: Atomic propositions, true, false
- Propositional operators: $\neg P$, $P \wedge Q$, $P \vee Q$, $P \Rightarrow Q$
- Some temporal operators:
 - $\bigcirc P$ / $X P$: P holds in the next state
 - $\square P$ / $G P$: P always holds
 - $\diamond P$ / $F P$: P holds eventually
 - $P U Q$: P holds until Q
 - $P \rightarrow Q$: P leads to Q

Note: Operators may be expressed by each other, e.g.

- $\diamond P = \neg \square \neg P$
- $\square P = P U \text{false}$
- $P \rightarrow Q = (P U Q) \wedge \diamond Q$



Concept: Universal Property

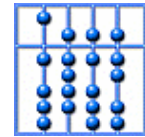


Purpose: Describe *properties for all possible paths of execution*

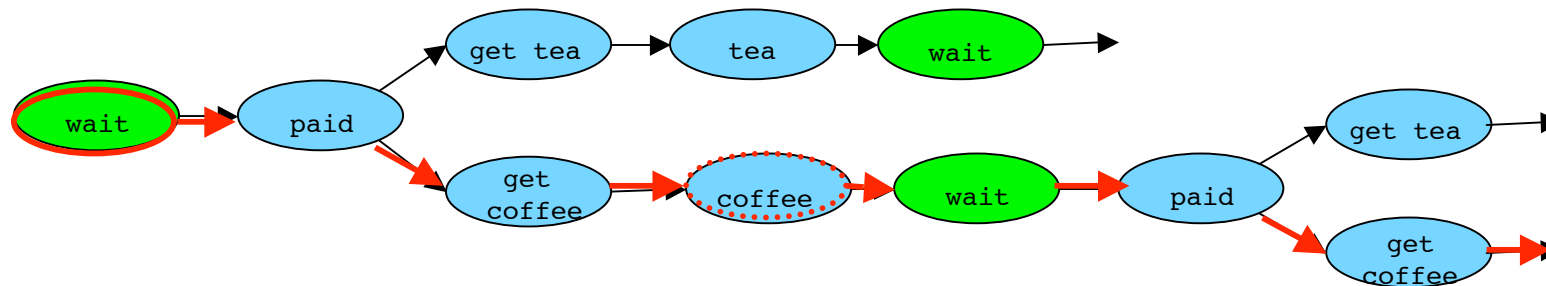
Concept: Universal Path Property

- Notation: $A \ P$ (“All P”)
- Interpretation: P holds for all possible paths starting at the current state

Example: “For all alternative executions, either tea or coffee is selected”,
 $A \ (X \ (\text{“Coffee button is pressed”} \vee \ \text{“Tea button is pressed”}))$



Concept: Existential Property

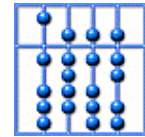


Purpose: Describe *properties for at least one path of execution*

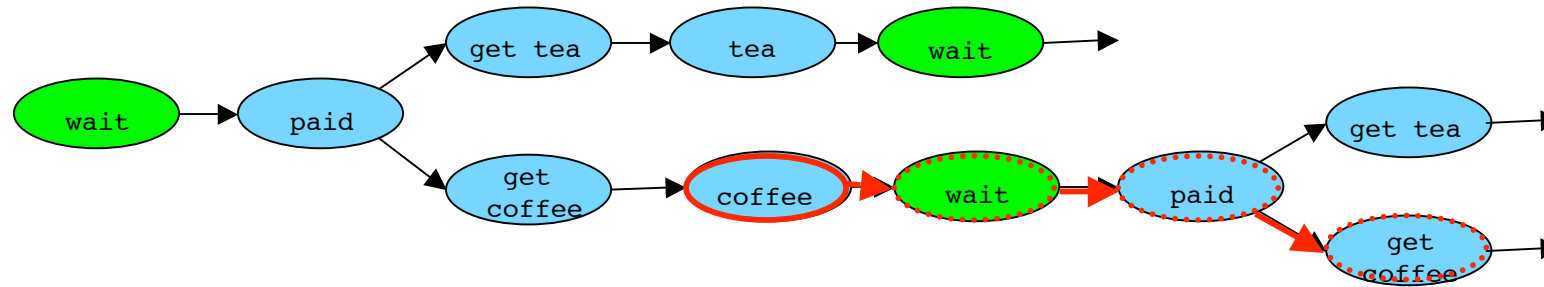
Concept: Existential Path Property

- Notation: $E P$ (“Some P ”)
- Interpretation: Starting at the current time point, P holds for at least one path

Example: “During at least one execution coffee is eventually dispensed”,
 $E (F \text{ “Coffee is in the dispenser slot”})$



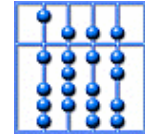
Concept: Path Properties



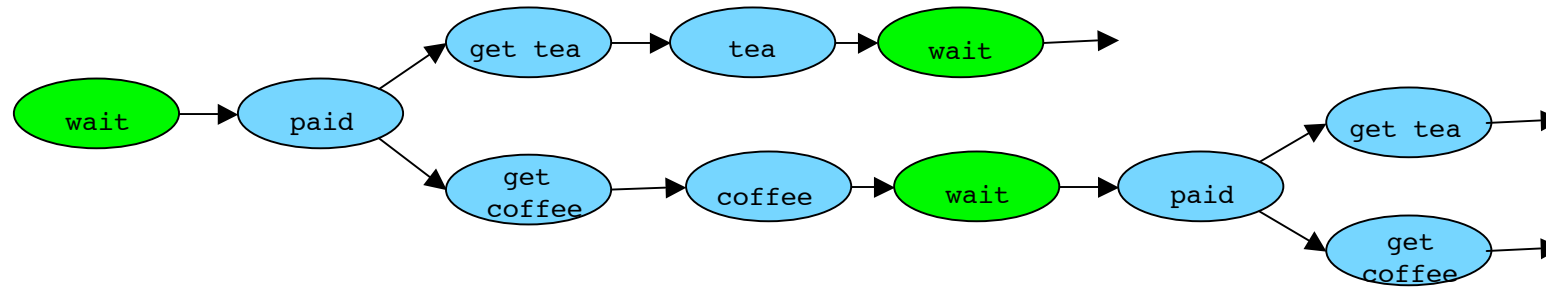
Purpose: Describe *property holding for an execution path*

Concept: Path property

- Model: Sequential execution $E = s_0 \cdot s_1 \cdot s_2 \cdot s_3 \cdot \dots$ of a Kripke structure
- Interpretation: Property holds for execution E
 - P holds for E : State property P holds for s_0
 - $X P$ holds for E : Path property P holds for execution $s_1 \cdot s_2 \cdot s_3 \cdot \dots$
 - $G P$ holds for E : Path property P holds for execution $s_i \cdot s_{i+1} \cdot s_{i+2} \cdot \dots$ for all $i \geq 0$
 - $F P$ holds for E : Path property P holds for execution $s_i \cdot s_{i+1} \cdot s_{i+2} \cdot \dots$ for some $i \geq 0$



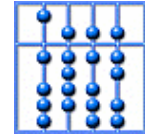
Concept: State Property



Purpose: Describe *properties holding for a complete execution tree*

Concept: State Property

- Model: State s_i of a Kripke structure
- Interpretation: Property holds for state s_i
 - P holds at s_i : $P \in O(s_i)$
 - $A P$ holds at s_i : Path property P holds for all executions starting in s_i
 - $E P$ holds at s_i : Path property P holds for some execution starting in s_i
- Validity of P concerning system S : Property holds in initial state s_0 of system S



Notation: CTL*

Purpose: Provide a notation *to describe observations about execution sequences*

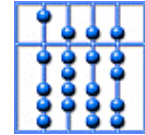
Notation: Computation Tree Logics (LTL) propositions P, Q

- Basic formulae: Atomic propositions, true, false
- Propositional operators: $\neg P, P \wedge Q, P \vee Q, P \Rightarrow Q$
- State operators:
 - $E P$: P holds for some path starting in the current state
 - $A P$: P holds for all paths starting in the current state
- Path operators:
 - $X P$: P holds in the next state
 - $G P$: P always holds
 - $F P$: P holds eventually
- Additional path operators
 - $P U Q$: P holds until Q
 - $P \rightarrow Q$: P leads to Q

Note: Operators may be expressed by each other

- See LTL
- $E P = \neg A \neg P$

Note: CTL: Subset allowing only pairs of State/Path operators (e.g., $E F, A G$)



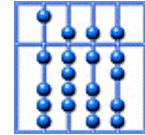
9.2 Describing Properties: Safety and Liveness

Goal: Define classes of properties common in specifications of discrete reactive systems

- Relevant: Address aspects arising during proofs
- General: Cover aspects independent of specific model of computation
- Abstract: Ignore aspects like duration actions

Concept: Safety, liveness

Model: Executions



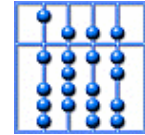
Concept: Safety Property

Purpose: Describe properties *restricting the occurrence of unwanted actions*

Concept: Safety property

- Intuition: “Nothing bad ever occurs”
- Proof principle: The violation of a safety property can be identified upon execution of a specific action

Example: “The vending machine never dispenses coffee and tea at the same time”



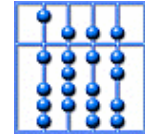
Definition: Safety Property

Purpose: Classify *safety properties*

Definition: Safety property

- Intuition: If a safety condition is violated, this violation can be detected after a finite number of steps
- Formalization: Safety property S over complete execution traces
 - Violation: $\neg S(s_0 \cdot s_1 \cdot s_2 \cdot s_3 \cdot \dots)$
 - Violating step: $s_0 \cdot s_1 \cdot s_2 \cdot s_3 \cdot \dots \cdot s_k$
 - Cannot be mended: $\neg S(s_0 \cdot s_1 \cdot s_2 \cdot s_3 \cdot \dots \cdot s_k \cdot s'_{k+1} \cdot s'_{k+2} \dots)$

Example: “Tea is supplied after tea is selected”



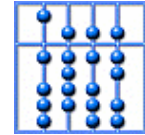
Concept: Liveness Property

Purpose: Describe properties *enforcing the occurrence of wanted actions*

Concept: Liveness property

- Intuition: “Something good eventually happens”
- Proof principle: Violation of liveness property can only be detected after looking at a complete (infinite) execution

Example: “The vending machine eventually supplies coffee after selecting the coffee-button”



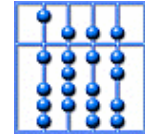
Definition: Liveness Property

Purpose: Formalize *liveness properties*

Definition: Liveness Property

- Intuition: Any execution with finite number of steps can be extended to a life execution
- Formalization: Liveness property L over complete execution traces
 - Finite execution step: $s_0 \cdot s_1 \cdot s_2 \cdot s_3 \cdot \dots \cdot s_k$
 - Can be extended: $L(s_0 \cdot s_1 \cdot s_2 \cdot s_3 \cdot \dots \cdot s_k \cdot s'_{k+1} \cdot s'_{k+2} \dots)$

Example: “Either tea or coffee is eventually supplied”



Application: Temporal Pattern

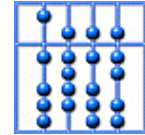
Safety pattern:

- Invariance: A violation of a law never occurs = $AG \neg \text{violated}$
- Triggering: A response is never issued unless triggered = $AG (\neg \text{response} \cup \text{trigger})$
- Bounded response: A response is issued k steps after the trigger = $AG (\text{trigger} \Rightarrow O^k \text{ response})$

Liveness pattern:

- Fulfillment: The required property is eventually established = EF property
- Weak fairness: Continuously enabled actions are eventually performed = $(GF (\neg \text{enabled} \vee \neg \text{performed}))$
- Strong fairness: Infinitely often enabled actions are eventually performed = $(GF \text{ enabled} \Rightarrow GF \text{ performed})$

Notation: Each observation property can be described as a combination of a safety and a liveness property



9 Questions

1. Exercise: Define a Kripke structure for the producer/consumer system describing which component is currently busy consuming or producing.
2. Exercise: Extend the Kripke structure to describe that either producer or consumer might get stuck during production or consumption.
3. What are corresponding logical forms of the following properties:
 - In every execution, the producer is always either busy or ready
 - In every execution, if producer and consumer are ready they will be busy in the next step
 - In every execution, the consumer will not eventually always be ready
 - There is an execution with the producer eventually always being busy
4. Which of the above properties hold
 - In the system of Exercise 1?
 - In the system of Exercise 2?
5. Express the following property in CTL: “It is possible that the producer eventually never becomes ready”. Is it equivalently expressible in LTL?
6. Express the following property in LTL: “In every execution, the consumer will eventually always be ready”. Is it equivalently expressible in CTL?
7. Express the following property in CTL*: “It is possible that the consumer is always finally ready”. Is it equivalently expressible in LTL? In CTL?