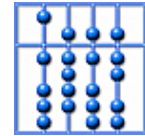


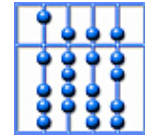
Vorlesung Specification of Distributed Systems

Dr. Bernhard Schätz
Leopold-Franzens Universität Innsbruck
Sommersemester 2005



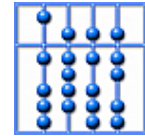
Overview

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



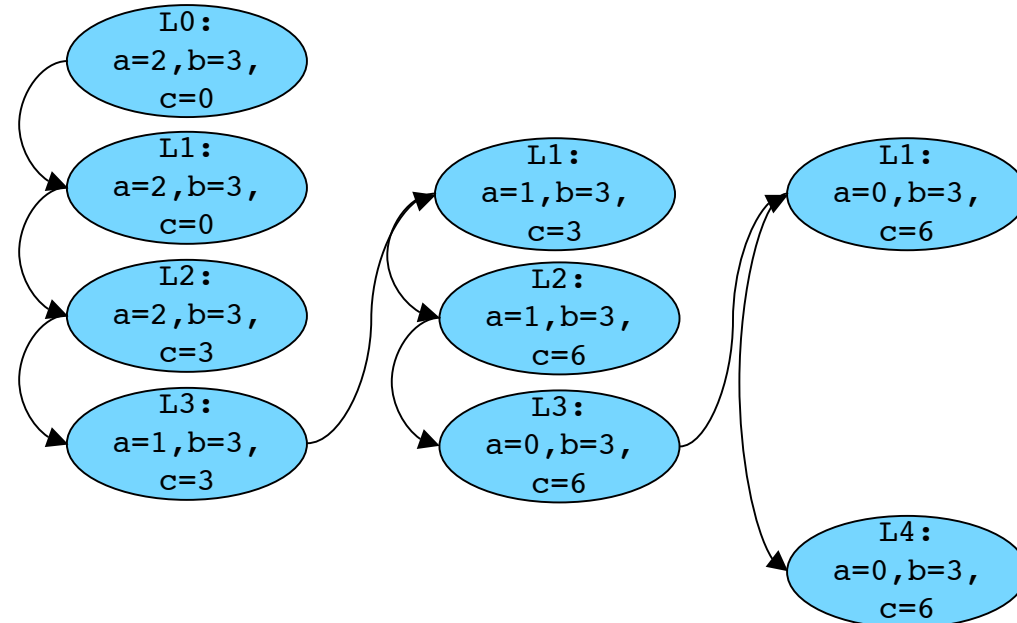
Overview

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



Recap: Computations - Control State/Data State

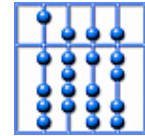
```
L0:  int a=2,b=3,c=0;
L1:  while(a>0) do {
L2:      c = c + b;
L3:      a = a - 1; }
L4:
```



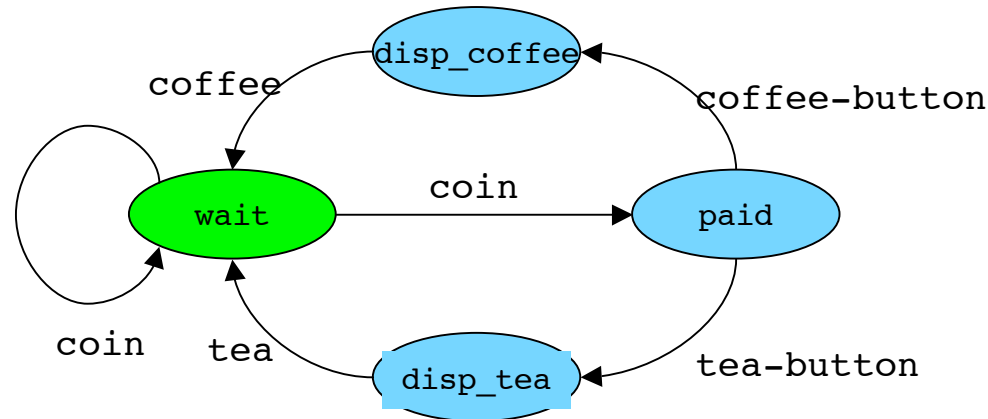
Concepts:

- 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: Extended Transition System (S, S_0, T)



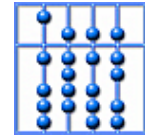
Recap: Modeling Reactivity



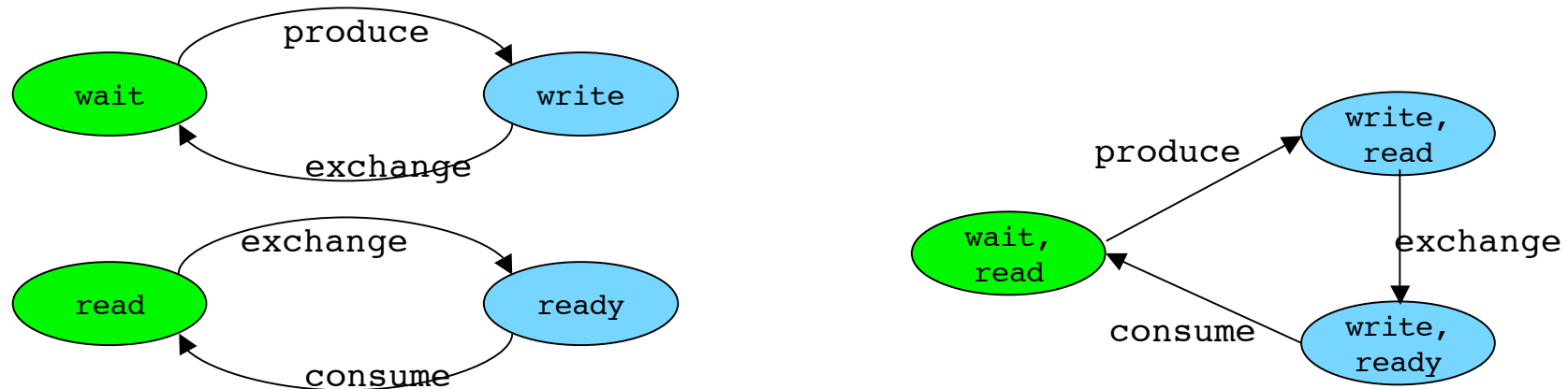
Concepts:

- 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)



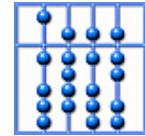
Recap: Modeling Concurrency - Synchronized Execution



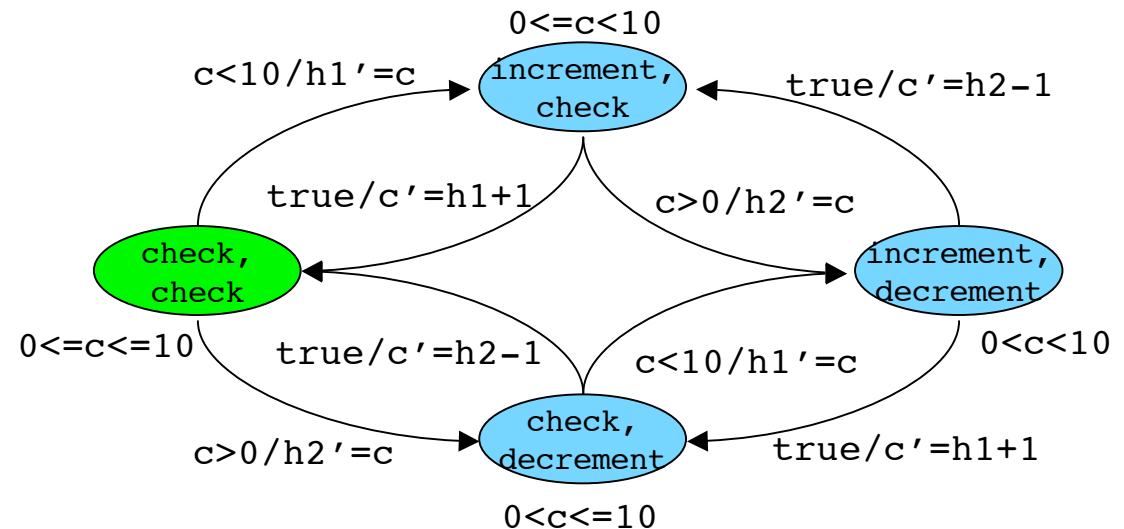
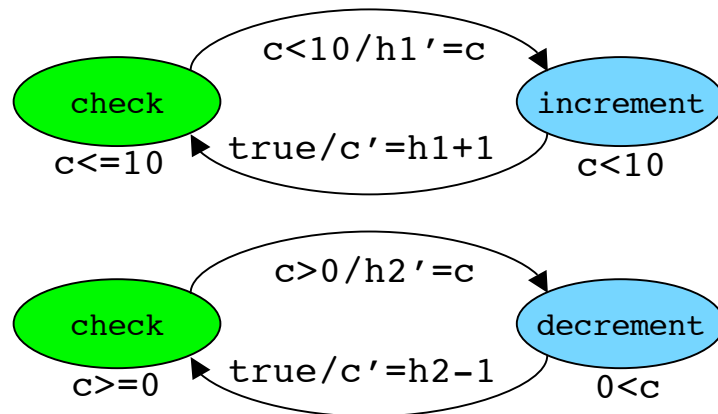
Concepts:

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

Model: Synchronized Product Automaton



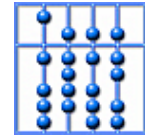
Recap: Modeling Shared Memory



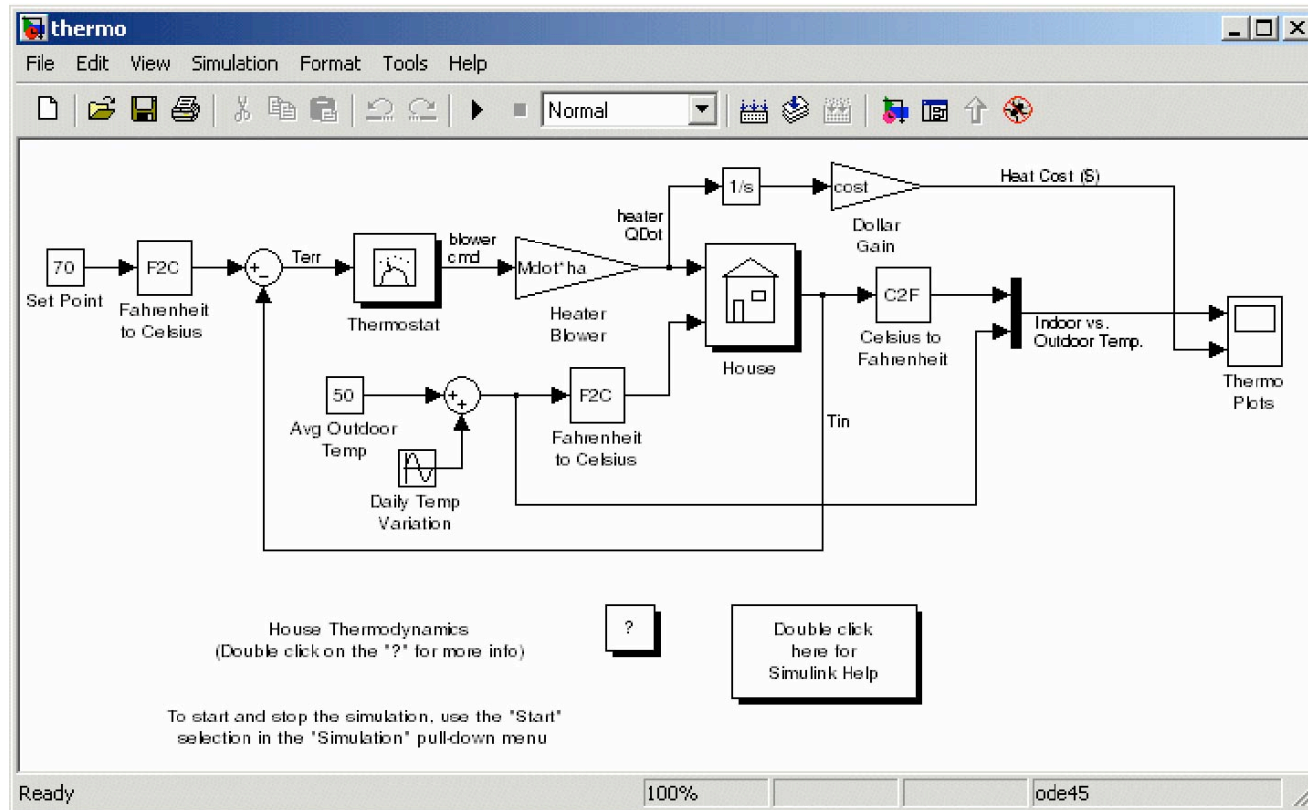
Concepts:

- Shared Memory: Product space with joint global state and independent local and control state
- Implicit interaction: Interleaved execution of interactions

Model: Combined Extended Transition System



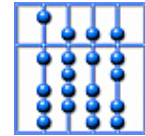
Motivation: Signal Flow



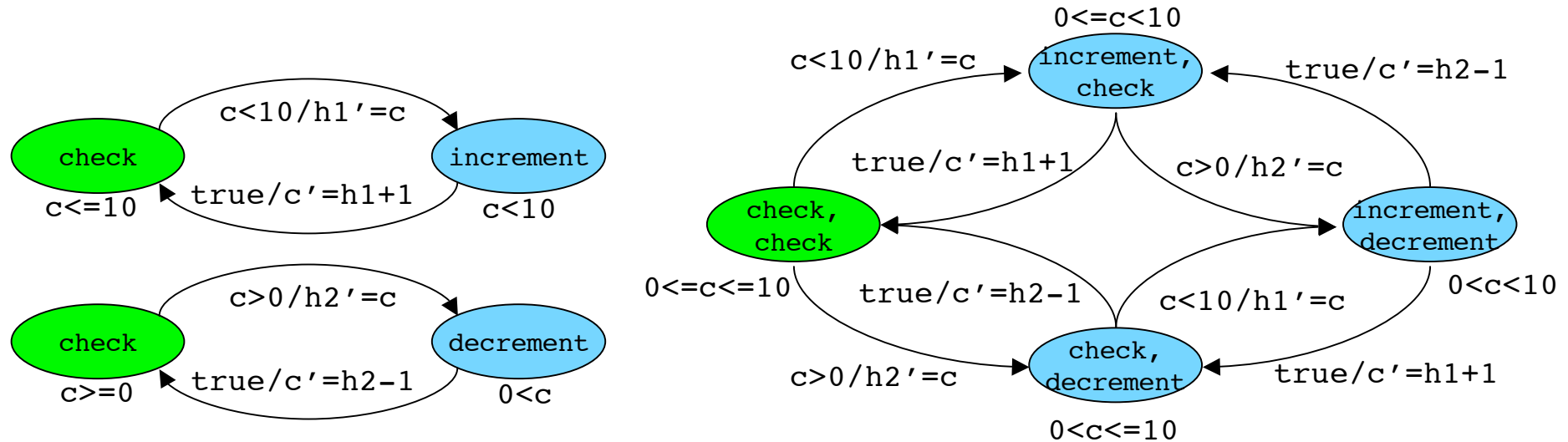
Source:
Matlab/Simulink Tutorial

Modular Development in Electrical Engineering:

- System constructed by composition of components
- Components communicate by exchanging signal

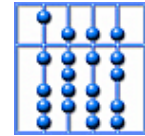


Motivation: Shared Memory vs Data Flow

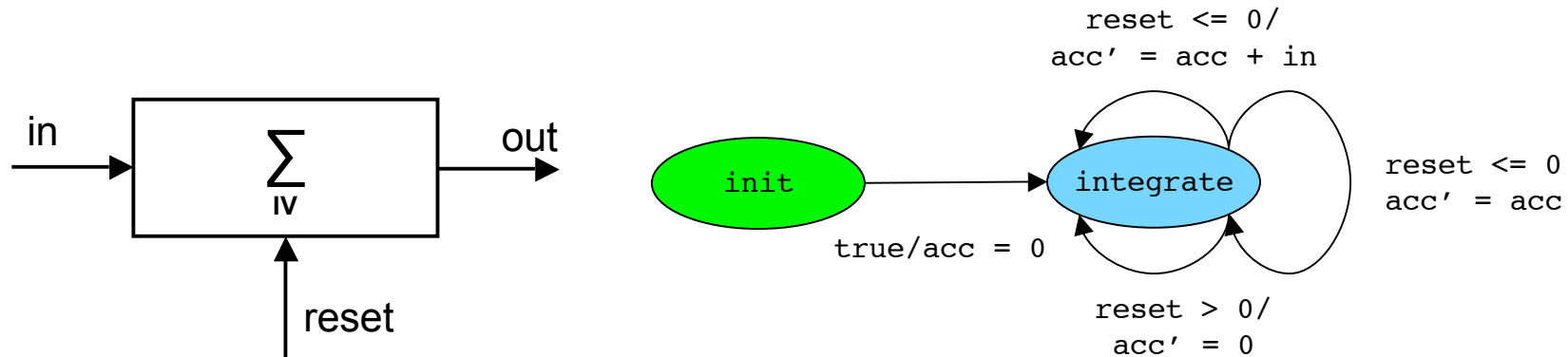


Shared Memory:

- Implicit Communication: No interface
- Problem: Integrity may be lost:
 - An update of the consumer may be lost if the producer is slower
 - An update of the producer may be lost if the consumer is slower

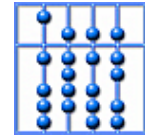


Motivation: Shared Memory vs. Data Flow



Shared Memory:

- Implicit Communication: No synchronization
- Problem: Communication may be lost:
 - A possible input is lost if the sender is faster
 - A possible output is missed if the integrator is faster



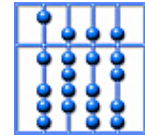
4.1 Signal-Based Interaction: Synchronized Models

Goal: Define a model to describe the behavior of locally distributed data flow systems

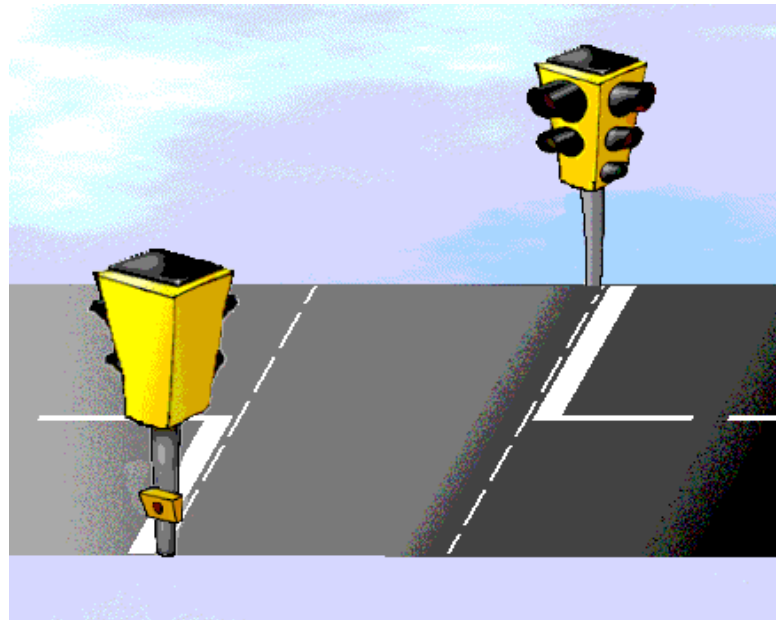
- Distinction between system and environment
- Explicit communication by signals
- Extended transitions by input and output

Concept: Interface, signal, synchronized execution, synchronized communication, message-asynchronous communication

Model: Synchronized Extended Finite Automata



Concept: Interface

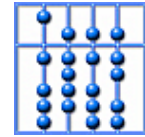


Purpose: Distinguish between system and environment

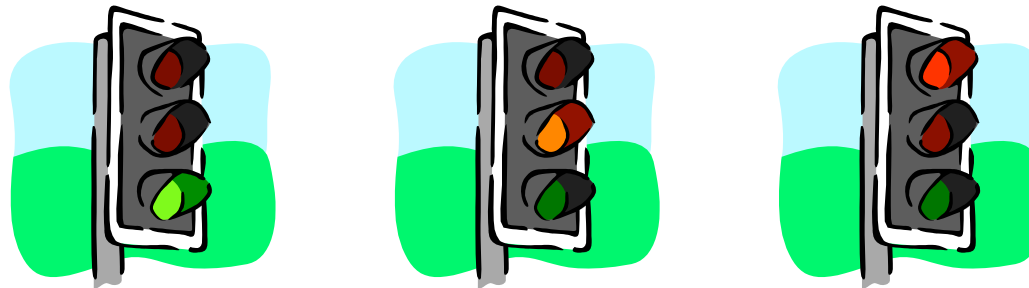
Concept: Interface, port

- Interface: Shared part of system and environment
- Port: Part of interface allowing data flow between system and environment

Example: Ports: Button, Indicator, Traffic Lights, Pedestrian Lights



Concept: Signal

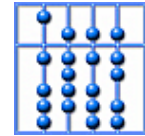


Purpose: Describing flow of information between system and environment

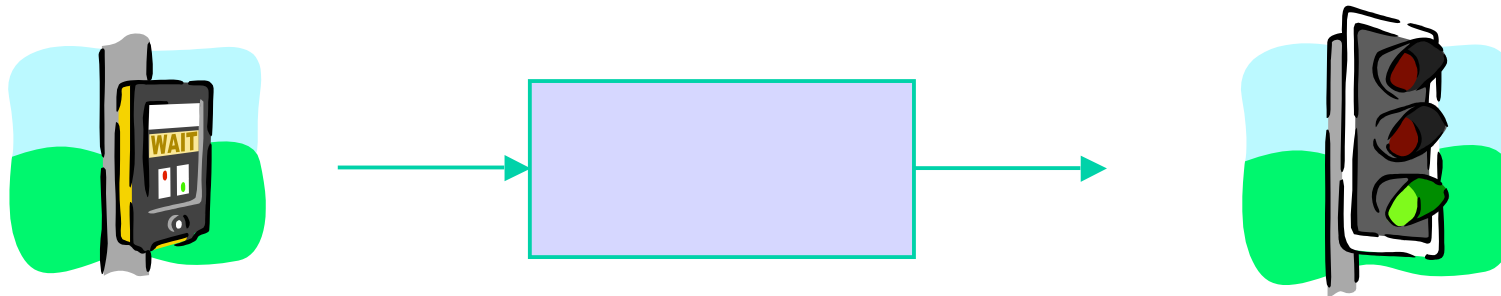
Concept: Signal = observation of flow of information over time

- Type of signal: Possible values of a signal
- History of signal: Value of signal for all instances of time

Example: Type of signal of traffic light : “Red”, “Yellow”, “Green”,
“RedYellow”



Concept: Input and Output



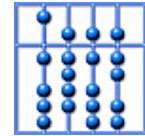
Purpose: Describing directed data flow

Concept: Input and output

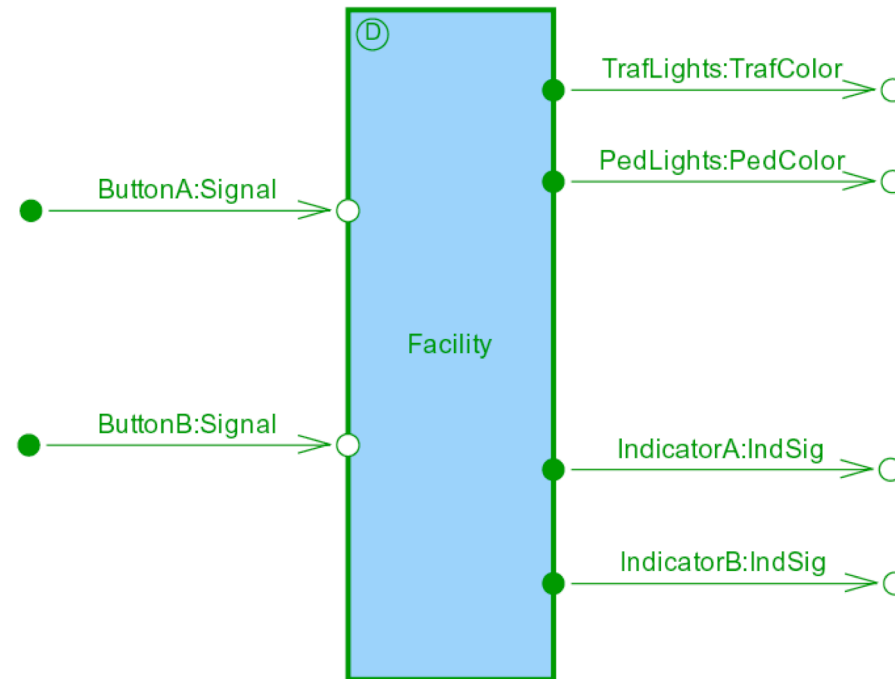
- Input signal: Signal controlled by environment
- Output signal: Signal controlled by system

Example:

- Input ports: ButtonA, ButtonB
- Output ports = TrafLights, PedLights, IndicatorA, IndicatorB



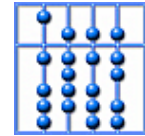
Notation: Structure Diagrams



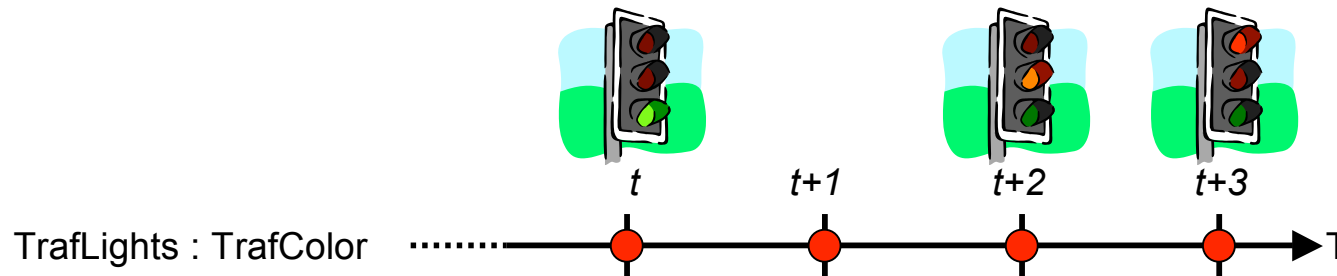
Structure Diagrams: Description of interface of system or component

- Name of system/component (e.g., Facility)
- Name/type of ports (e.g., TrafLights : TrafColor)

Variants: Capsule Diagrams (ROOM), Data Flow Diagrams (Rhapsody)



Concept: Discrete Signal

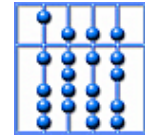


Purpose: Describing clocked executions of systems

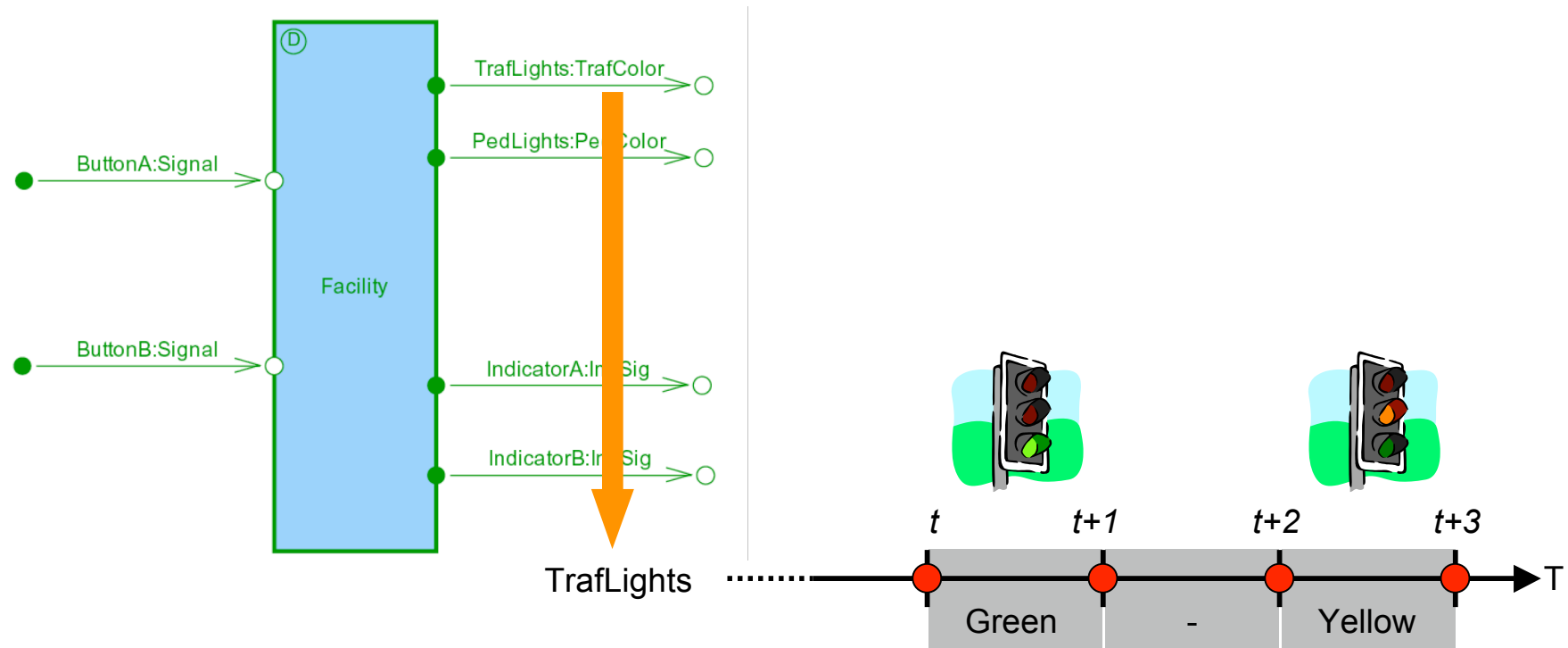
Concept: Discrete-time signal = signal changes at defined time steps

- Presence: Signal defined at current time step according to type of signal
- Absence: Signal undefined at current time step

Example: Signal $\text{TrafLights} \in \text{TrafColor} \cup \{-\}$ (“-” = absence of signal)

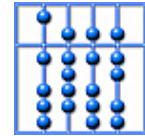


Example: Signal History

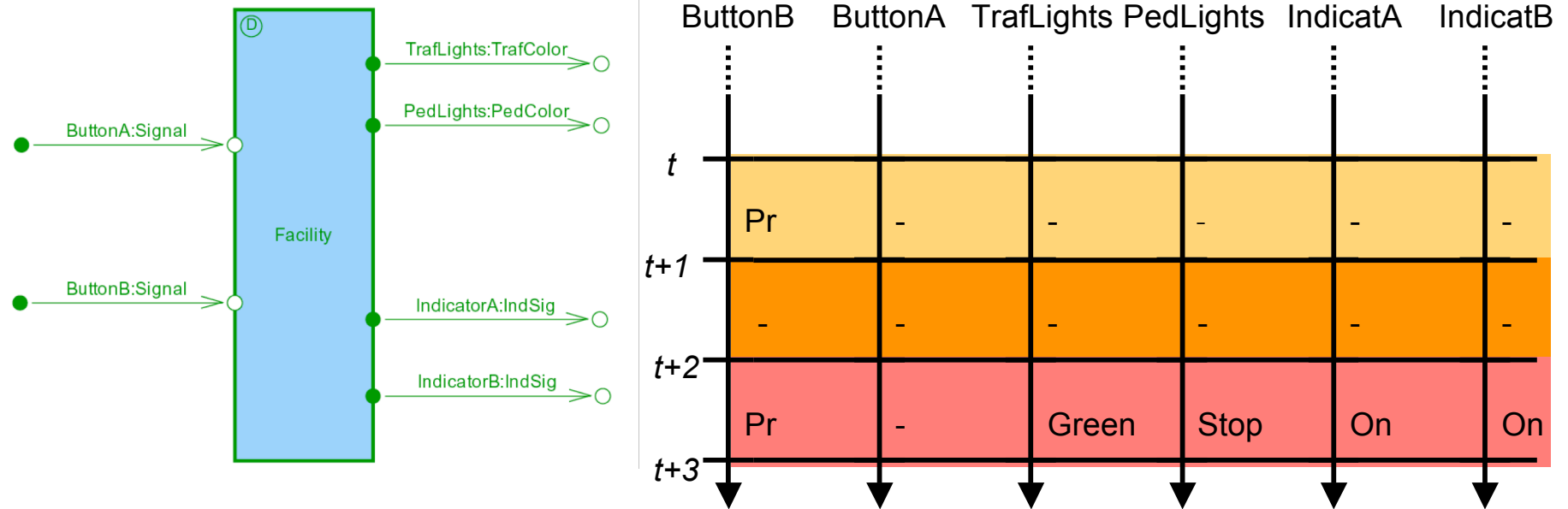


Signal history of discrete signal: Signal unchanged between time steps

- (Infinite) sequence of values (including “-” for absence)
- Example: ... • Green • - • Yellow • ...



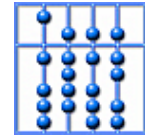
Concept: Clocked Execution



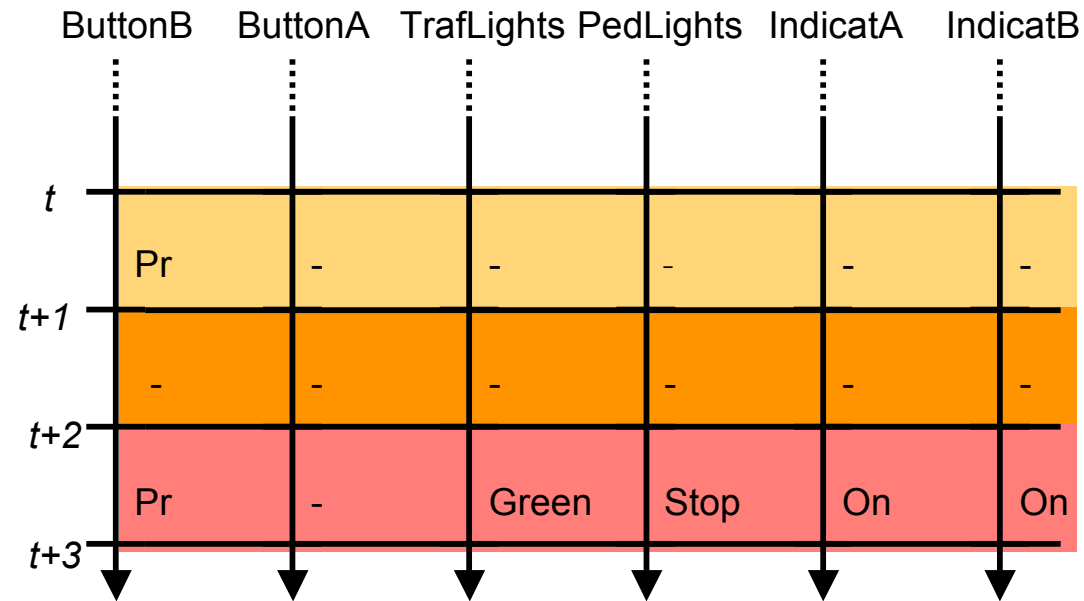
Purpose: Describing discrete execution steps

Concept: Clocked execution = discrete execution cycle

- Read all input ports simultaneously
- Perform execution step
- Write all output ports simultaneously



Concept: Clocked Observation Trace

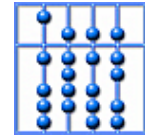


Purpose: Describing signals at interfaces caused by discrete execution steps

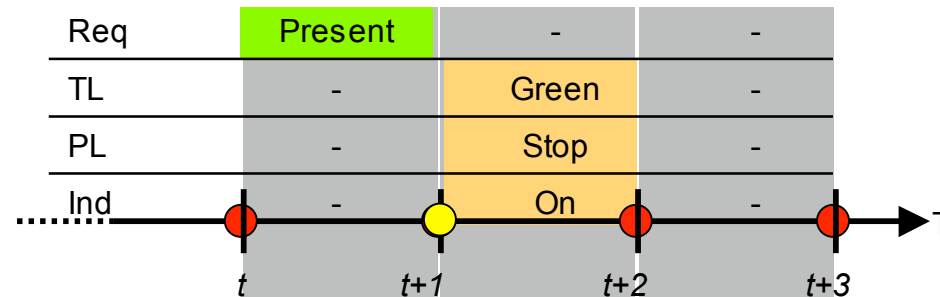
Concept: Clocked observation trace

- Trace = Sequence of simultaneous interactions
- Interaction = Combination of current signal values

Example: ... • (Present,-,-,-,-) • (-,-,-,-,-) • (Present,-,Green,Stop,On,On) • ...



Concept: Sending/Receiving Signals



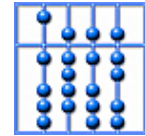
Purpose: Distinguishing signals from variables

Concept: Send/receive actions for interface ports at current time step

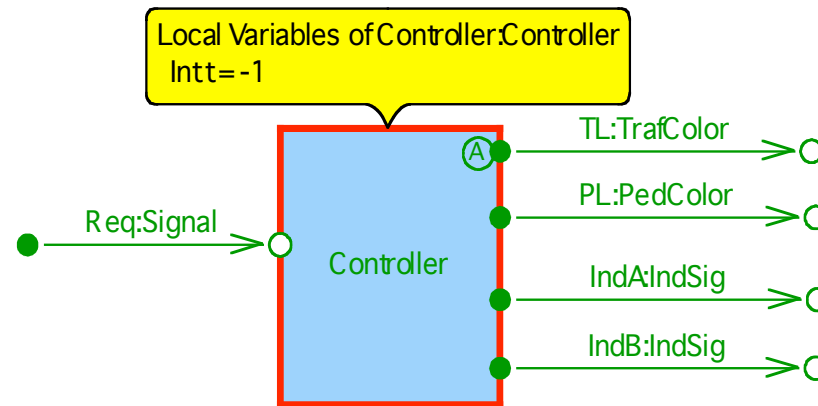
- Send action: Write signal on output port after current time step
- Receive action: Read signal on input port after current time step

Note: Operational interpretation:

- Unsent signals are undefined/absent

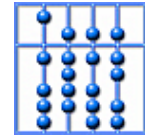


Model: Extended Transition System

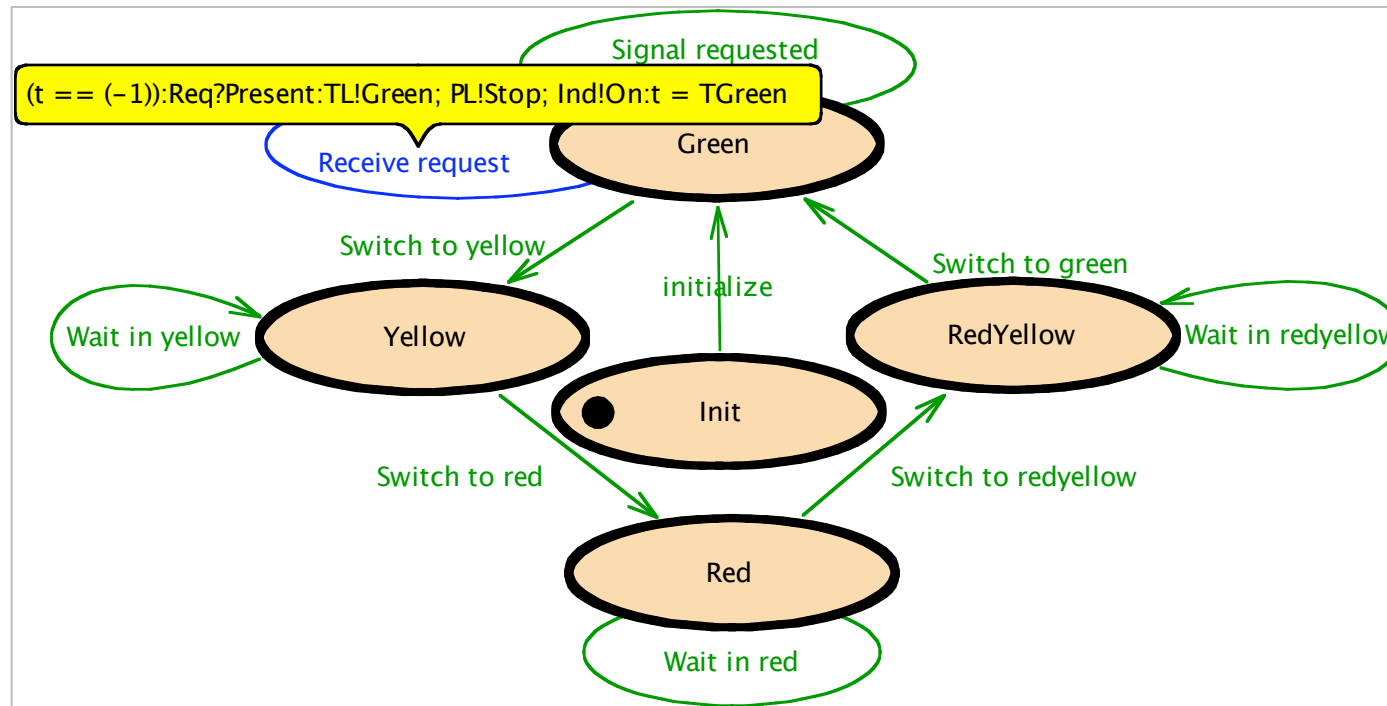


Model: State space $S \subseteq L \times V_1 \times \dots \times V_k \times I_1 \times \dots \times I_m \times O_1 \times \dots \times O_n$

- Control State L (e.g., Init, Green, Yellow, Red, RedYellow)
- Local Variables V_1, \dots, V_k (e.g., $t:\text{Int}$)
- Input State: Input Signals I_1, \dots, I_m (e.g., $\text{Req:Signal} \cup \{-\}$)
- Output State: Output Signals O_1, \dots, O_n (e.g., $\text{TL:TrafColor} \cup \{-\}$, $\text{PL:PedColor} \cup \{-\}$)

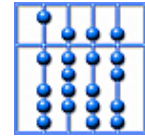


Notation: Transition

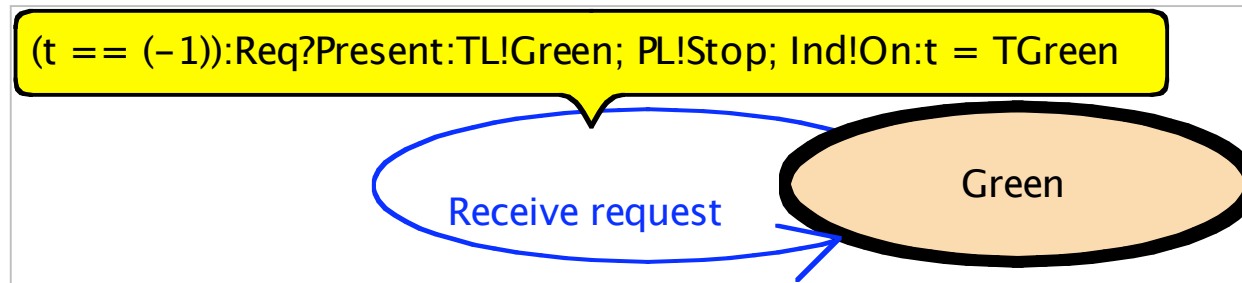


Describing atomic actions:

- Pre condition: $t == -1$
- Input action: $Req?Present$
- Output action: $TK!Green; PL!Stop; Ind!On$
- Post condition: $T = TGreen$ (short for „ $T' = TGreen$ “)



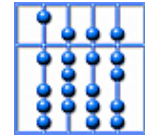
Model: Transition



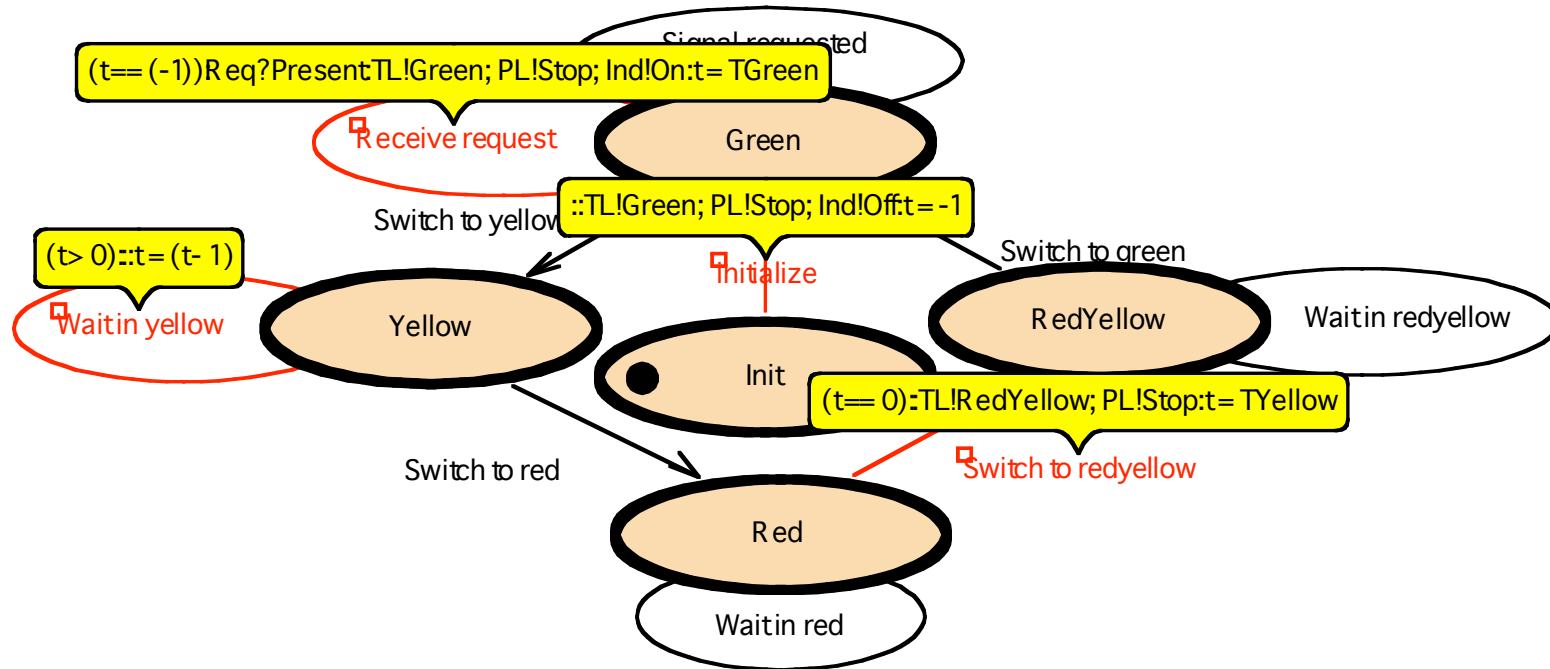
Pre-State			Post-State				
State	t	Req	TL	PL	Ind	t	State
Green	-1	Present	Green	Stop	On	TGreen	Green

Model: Extended Transition System

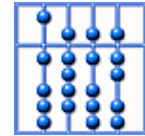
- Pre-State: Control State, Data State, Input Ports
- Post-State: Output Ports, Data State, Control State



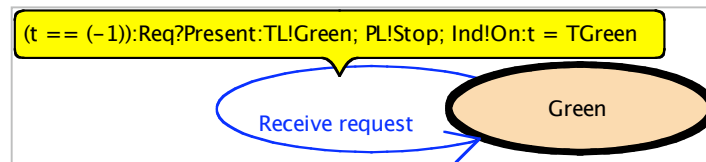
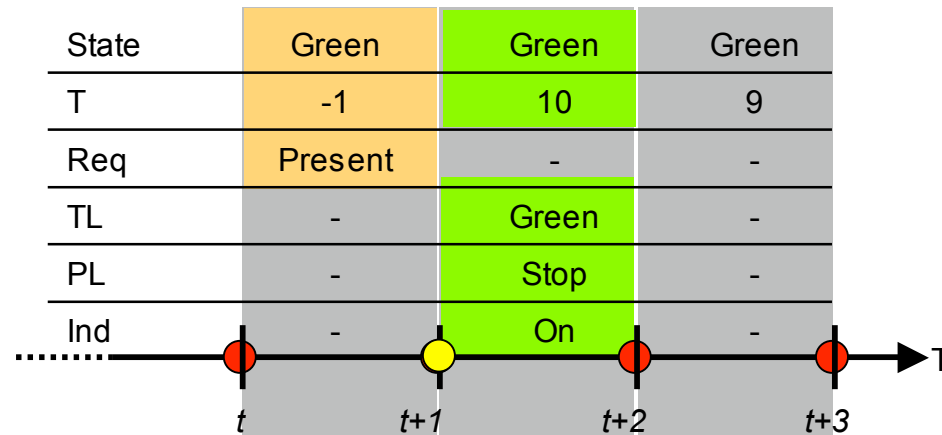
Example: Transition



Pre-State			Post-State				
State	t	Req	TL	PL	Ind	t	State
Init			Green	Stop	Off	-1	Green
Green	-1	Present	Green	Stop	On	TGreen	Green
Yellow	>0		-	-	-	t-1	Yellow
Red	0		RedYellow	Stop	-	TYellow	RedYellow



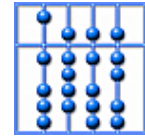
Example: Transition



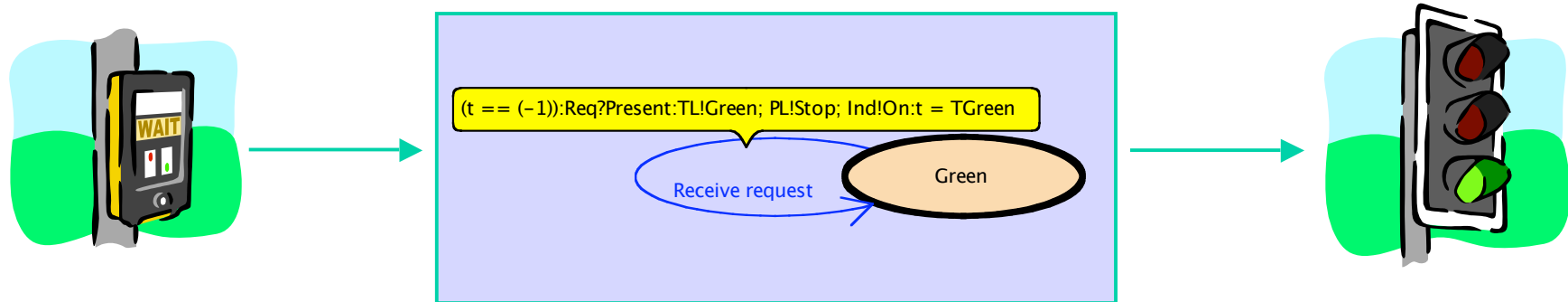
Pre-State			Post-State				
State	t	Req	TL	PL	Ind	t	State
Green	-1	Present	Green	Stop	On	TGreen	Green

Execution of Transitions: Application of Pre-State/Post-State Relation

- Pre-State: Control State, Data State, Input Ports
- Post-State: Output Ports, Data State, Control State



Concept: Message-Asynchronous Communication

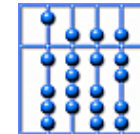


Purpose: Describing communication via non-blocking data flow

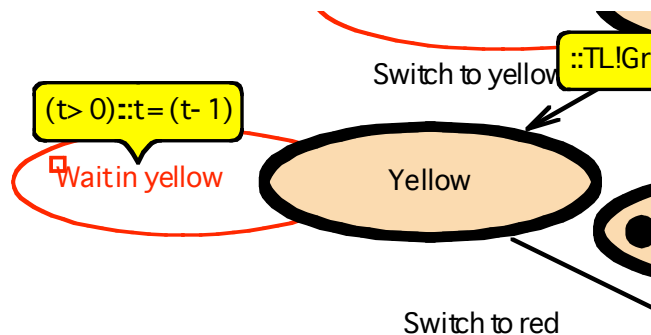
Concept: Message-asynchronous communication = Send actions never block

Note: Signals are transient = Unread signals are lost

Example: Environment may output "Req!Present" while $t \neq -1$ in state Green



Model: Input-Enabled Transition Systems

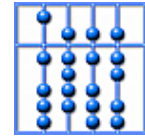


Pre-State			Post-State				
State	t	Req	TL	PL	Ind	t	State
Yellow	>0	-	-	-	-	t-1	Yellow
Yellow	==0	-	Red	Go	Off	0	Red
Yellow	>0	Present	Yellow	-	-	T-1	Yellow

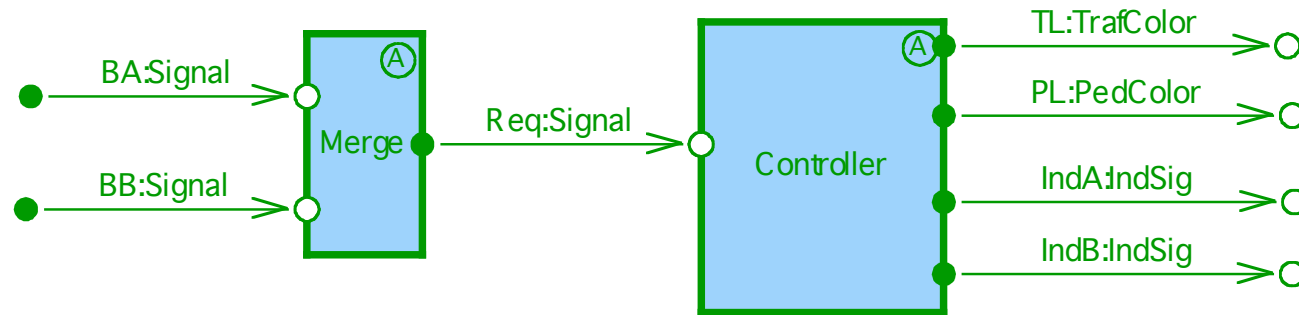
Purpose: Model non-blocking of input actions of environment

Model: Input-enabled ETS

- Input-complete transition relation
- Transition relation contains a transition for each possible pre-state



Concept: Composition

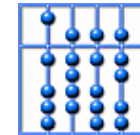


Purpose: Describing systems composed of concurrent components

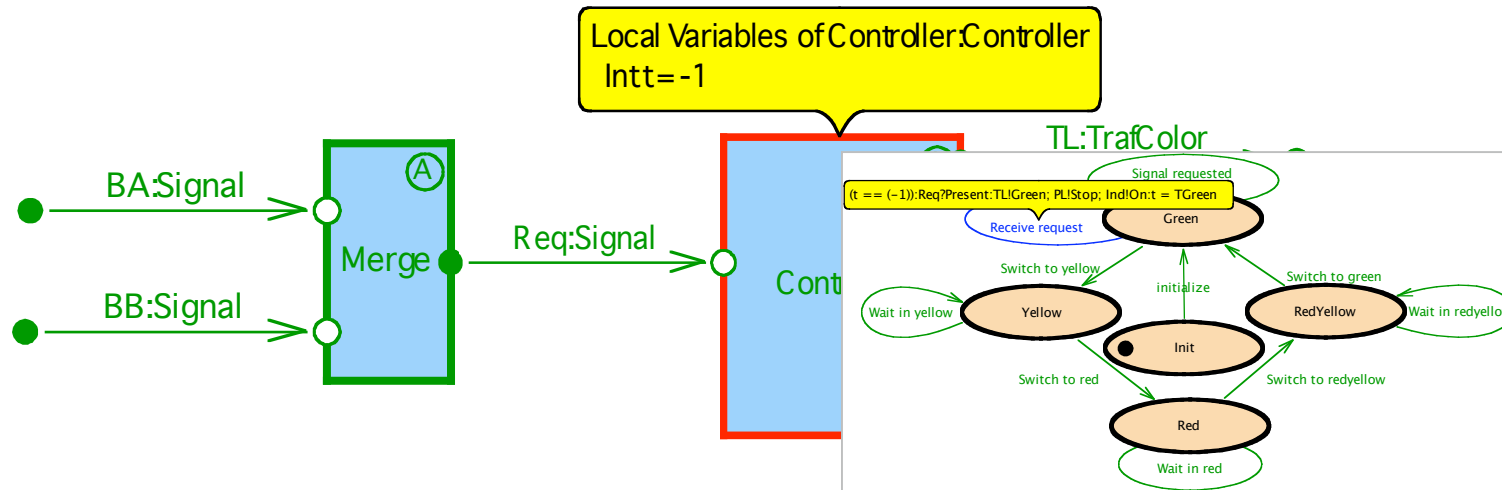
Concept: Composition = Parallel execution of components

- Execution: Components are synchronously executed
- Communication: Data flow
 - Output ports are connected to input ports by directed channels
 - Signals of output ports are immediately available at input ports

Example: Channel “Req” connects output port of Merge with input port of Controller



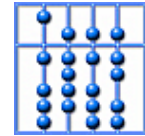
Model: Synchronized Transition Systems



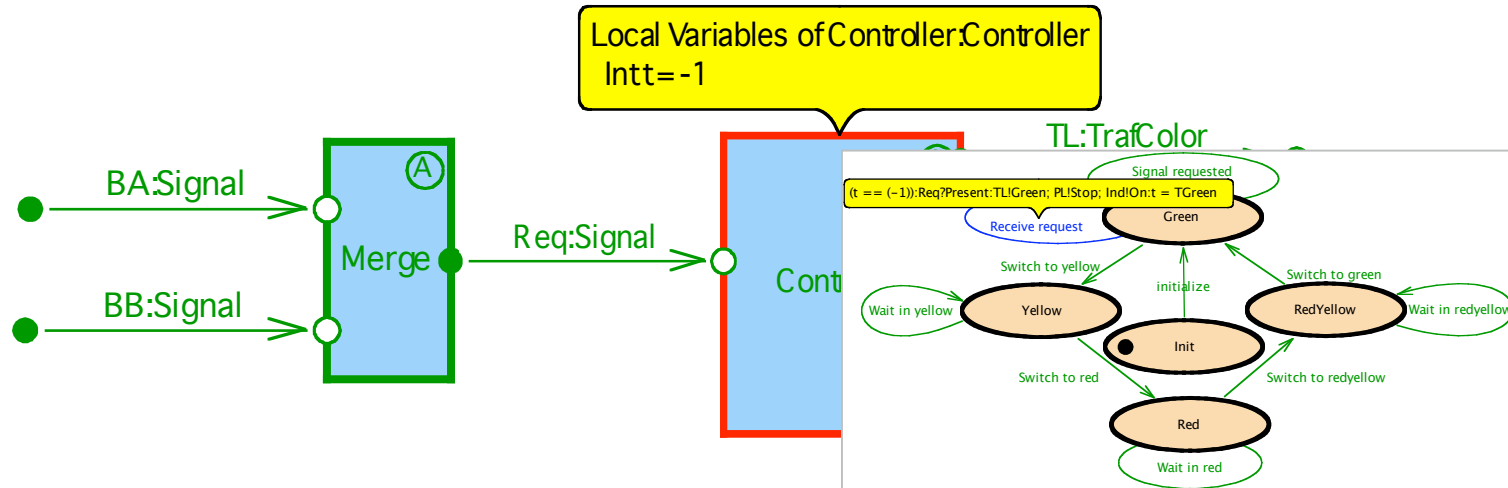
Model: State space of synchronized extended transition systems:

- $S_1 \subseteq L_1 \times V_{1,1} \times \dots \times V_{1,k} \times I_{1,1} \times \dots \times I_{1,m} \times O_{1,1} \times \dots \times O_{1,n}$
- $S_2 \subseteq L_2 \times V_{2,1} \times \dots \times V_{2,r} \times I_{2,1} \times \dots \times I_{2,s} \times O_{2,1} \times \dots \times O_{2,t}$
- Shared Ports: $I_{1,1} = O_{2,1}, \dots, I_{1,v} = O_{2,v}, I_{2,1} = O_{1,1}, \dots, I_{2,w} = O_{1,w}$
- Product space $S \subseteq$

$$\begin{array}{l}
 L_1 \times L_2 \times \\
 V_{2,1} \times \dots \times V_{2,k} \times V_{2,1} \times \dots \times V_{2,r} \times O_{1,1} \times \dots \times O_{1,w} \times O_{2,1} \times \dots \times O_{2,v} \times \\
 I_{1,v+1} \times \dots \times I_{1,m} \times I_{2,w+1} \times \dots \times I_{2,s} \times \\
 O_{1,w+1} \times \dots \times O_{1,n} \times O_{2,v} \times \dots \times O_{2,n}
 \end{array}$$



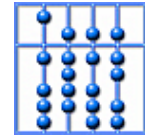
Model: Synchronized Transition Systems



Purpose: Describing clocked execution of communicating components

Model: Synchronized Extended Transition System:

- Synchronization on shared ports
- Synchronized transitions of **both** $(S_1, S_{1,0}, T_1)$ and $(S_2, S_{2,0}, T_2)$
- $((l_1, l_2, v_1, v_2, h_1, h_2, i_1, i_2, o_1, o_2), (l'_1, l'_2, v'_1, v'_2, h'_1, h'_2, i'_1, i'_2, o'_1, o'_2))$ for each
 - $((l_1, v_1, i_1, h_1, o_1), (l'_1, v'_1, i'_1, h'_1, o'_1)) \in T_1$
 - $((l_2, v_2, i_2, h_2, o_2), (l'_2, v'_2, i'_2, h'_2, o'_2)) \in T_2$



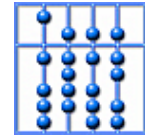
4.1 Summary: Synchronized Models

Concepts:

- Data Flow: Communication by exchange of signals/messages
- Interface: Description of shared input/output ports
- Signal: Atomic flow of information over time
- Message-asynchronous communication: Non-blocking data flow
- Synchronized Composition: Simultaneous execution of components

Model:

- Synchronized Extended Finite Automata



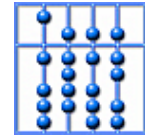
4.2 Immediate Data Flow: Perfectly-Synchronized Models

Goal: Define a model to describe the behavior of low-level systems

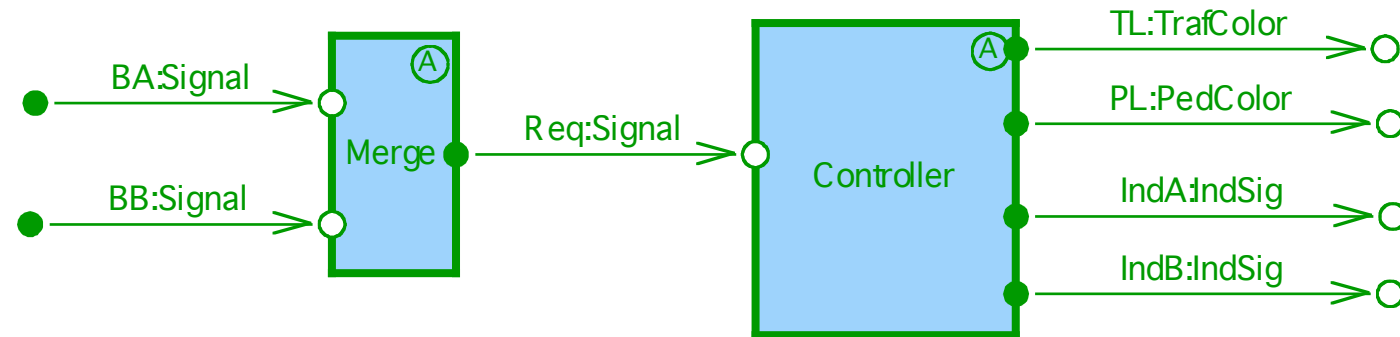
- Distinction between system and environment
- Explicit undelayed communication by signals
- Extended immediate transitions by input and output

Concept: Perfectly synchronized communication

Model: Synchronized Extended Finite Automata

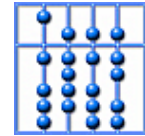


Motivation: Immediate Reaction

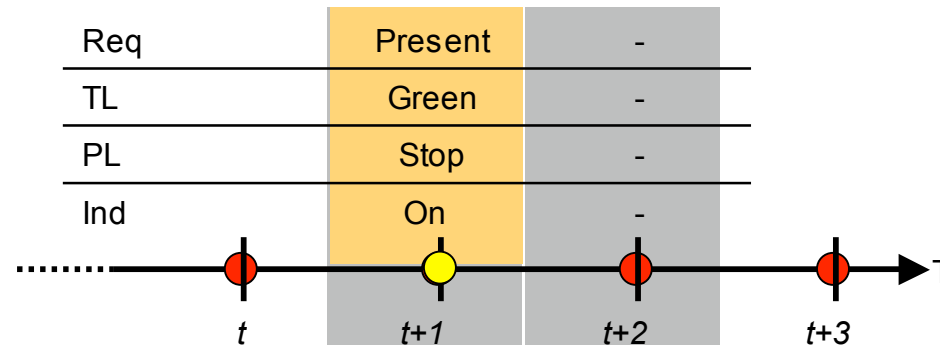


Observation: Synchronized Execution Cycle

- Action at current time step considers
 - input before current time step
 - output after current time step
- Reaction delayed for on time step



Concept: Immediate Communication



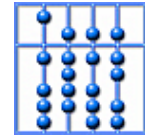
Purpose: Describing undelayed communication

Concept: Send/receive actions for interface ports

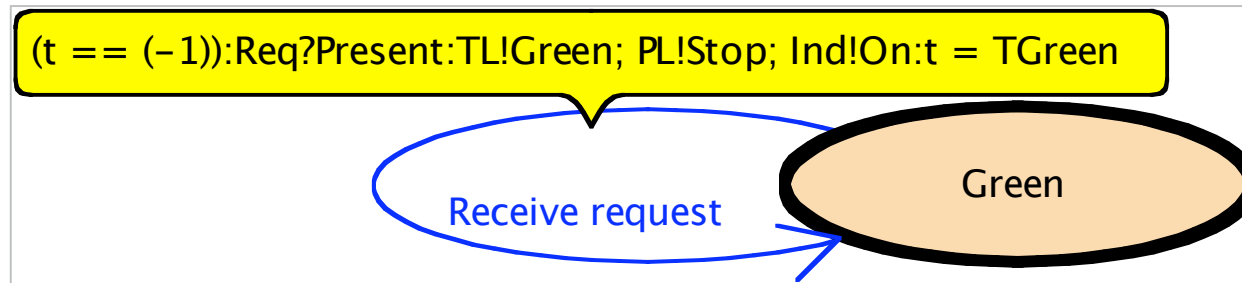
- Send action: Write signal on output port at current time step
- Receive action: Read signal on input port at current time step

Note: Immediate communication:

- Output immediately available as input



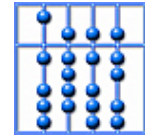
Model: Transition



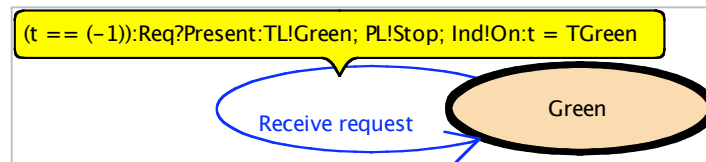
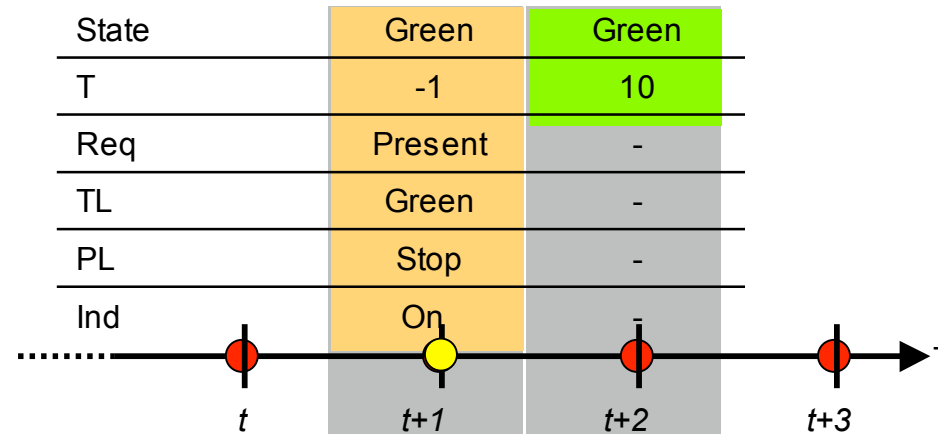
Pre State						Post State	
State	t	Req	TL	PL	Ind	t	State
Green	-1	Present	Green	Stop	On	TGreen	Green

Model: Extended Transition System

- Pre-State: Control State, Data State, Input Ports, Output Ports
- Post-State: Data State, Control State



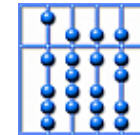
Example: Immediate Transition



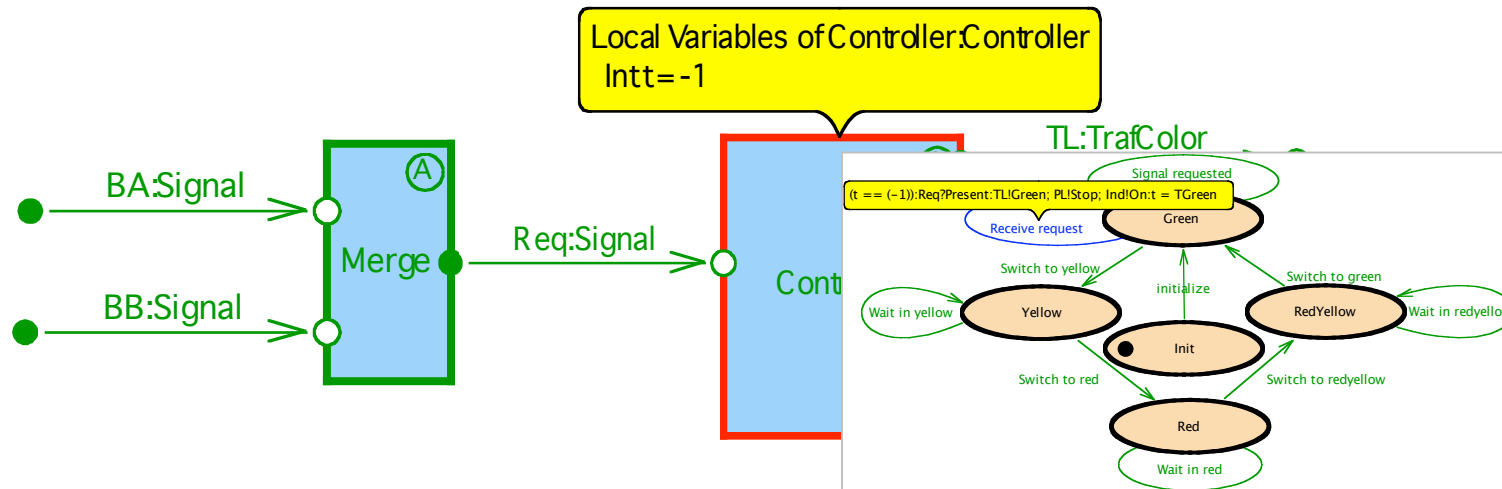
Pre State						Post State	
State	t	Req	TL	PL	Ind	t	State
Green	-1	Present	Green	Stop	On	TGreen	Green

Execution of Transitions: Application of Pre-State/Post-State Relation

- Pre-State: Control State, Data State, Input Ports
- Post-State: Output Ports, Data State, Control State



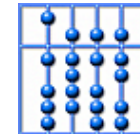
Model: Perfectly Synchronized Transition Systems



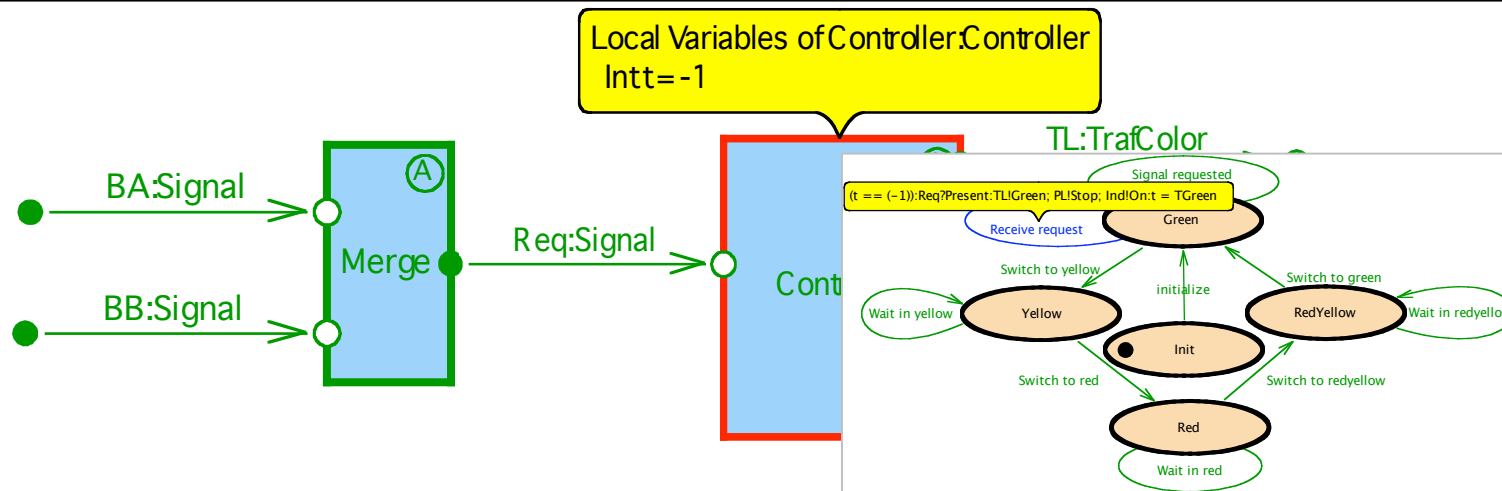
Model: State space of perfectly synchronized extended transition systems:

- $S_1 \subseteq L_1 \times V_{1,1} \times \dots \times V_{1,k} \times I_{1,1} \times \dots \times I_{1,m} \times O_{1,1} \times \dots \times O_{1,n}$
- $S_2 \subseteq L_2 \times V_{2,1} \times \dots \times V_{2,r} \times I_{2,1} \times \dots \times I_{2,s} \times O_{2,1} \times \dots \times O_{2,t}$
- Shared Ports: $I_{1,1} = O_{2,1}, \dots, I_{1,v} = O_{2,v}, I_{2,1} = O_{1,1}, \dots, I_{2,w} = O_{1,w}$
- Product space $S \subseteq$

$$\begin{array}{l}
 L_1 \times L_2 \times \\
 V_{2,1} \times \dots \times V_{2,k} \times V_{2,1} \times \dots \times V_{2,r} \times \\
 I_{1,v+1} \times \dots \times I_{1,m} \times I_{2,w+1} \times \dots \times I_{2,s} \times \\
 O_{1,w+1} \times \dots \times O_{1,n} \times O_{2,v} \times \dots \times O_{2,n}
 \end{array}$$



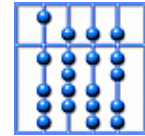
Model: Perfectly Synchronized Transition Systems



Purpose: Describing execution of immediately communicating components

Model: Perfectly synchronized Extended Transition System:

- Synchronization on shared ports (transition-local variables)
- Synchronized transitions of **both** $(S_1, S_{1,0}, T_1)$ and $(S_2, S_{2,0}, T_2)$
- $((l_1, l_2, v_1, v_2, i_1, i_2, o_1, o_2), (l'_1, l'_2, v'_1, v'_2, i'_1, i'_2, o'_1, o'_2))$ for each
 - $((l_1, v_1, i_1, h_1, o_1), (l'_1, v'_1, i'_1, h'_1, o'_1)) \in T_1$
 - $((l_2, v_2, i_2, h_2, o_2), (l'_2, v'_2, i'_2, h'_2, o'_2)) \in T_2$



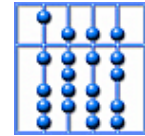
4.2 Summary: Perfectly Synchronized Models

Concepts:

- Data Flow: Communication by exchange of signals/messages
- Interface: Description of shared input/output ports
- Signal: Atomic flow of information over time
- Message-asynchronous communication: Non-blocking data flow
- Immediate communication: Output immediately available as input
- Synchronized composition: Simultaneous execution of components

Model:

- Perfectly-Synchronized Extended Finite Automata



4.3 Questions

1. Exercise: Build an integrator using an adder and a doubler, both with the synchronous and the perfectly-synchronous model.
2. Exercise: Construct a three-step execution trace for both models.
3. Does the synchronous model lead to satisfactory solution? If not, what is a possible remedy?
4. Does the perfectly synchronous model lead to satisfactory solution? If not, what is the cause for the problem?
5. Can you define a structural criterion avoiding this problem?
6. Can you define a syntactical criterion on the product automaton avoiding this problem?
7. What is the difference between the synchronization strategies of LTS, ETS; and STS?
8. How does this affect fairness of execution?
9. Can you tell whether a STS is sequential or parallel by looking at its observation traces?
10. What are the strongest assumptions that can be made about an input port?