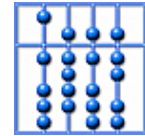


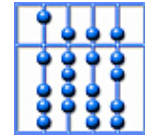
Vorlesung Specification of Distributed Systems

Dr. Bernhard Schätz
Leopold-Franzens Universität Innsbruck
Sommersemester 2005



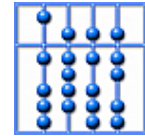
Overview

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

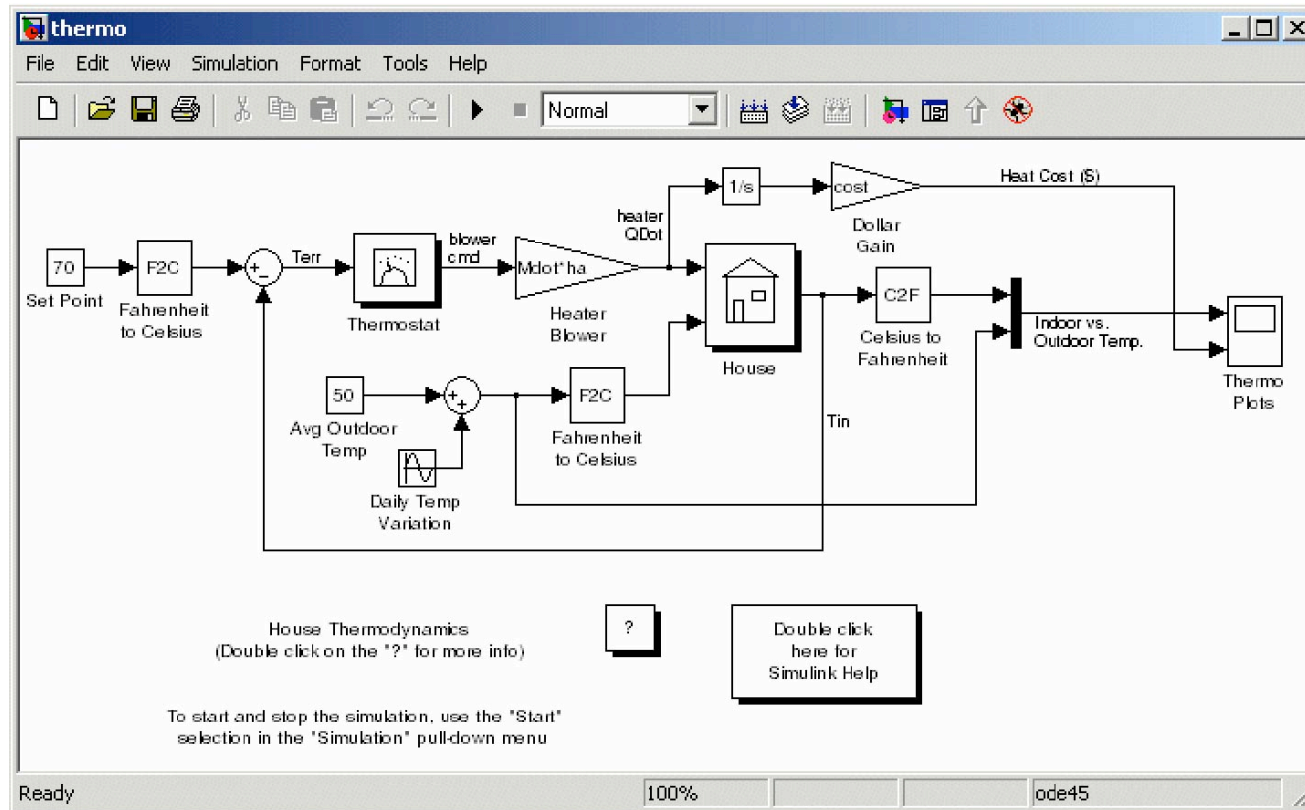


Overview

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



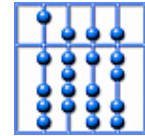
Motivation: Signal Flow



Source:
Matlab/Simulink Tutorial

Modular Development in Electrical Engineering:

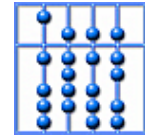
- System constructed by composition of components
- Components communicate by exchanging signal



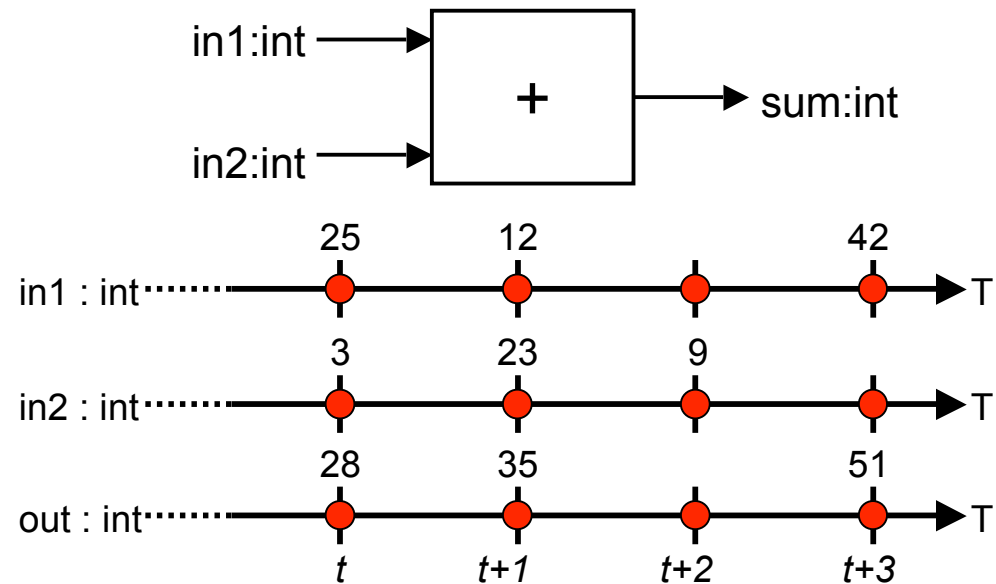
Recap: Models

Classes of models:

- **Operational model:** Specification in form of an abstract machine
 - Concepts: State, (labeled) transition, transition system
 - Verification: (Bi-)Simulation
 - Example: (Timed) Automata
- **Denotational model:** Specification in form of observable behavior
 - Concepts: Interface, event, (observation) trace,
 - Verification: Behavior Inclusion
 - Example: Trace-based models
- **Algebraic model:** Specification in form of syntactic formulae
 - Concepts: Process, action, choice,
 - Verification: Term equivalence
 - Example: (T)CSP



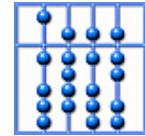
2.3 Recap: Modeling Behavior



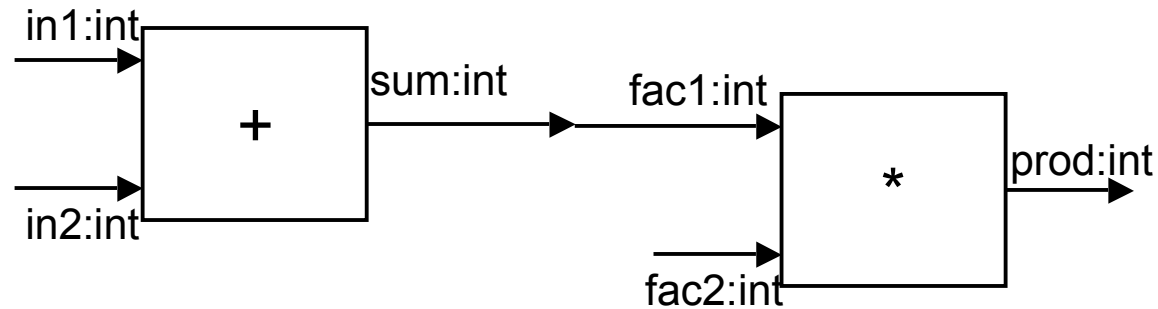
Concepts:

- Interface: Shared part of system and environment
- Signal: Time-based description of flow of information
- Message: Event-based description of flow of information
- (Observation) Trace: Sequence of interactions at interface
- Behavior: Set of (observation) traces

Model: Sets of timed/untimed traces



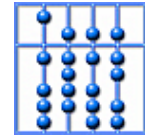
Recap: Modeling Communication



Concepts:

- Connected Interfaces: Shared parts of components
- Independent interaction: Disjoint observations about traces
- Synchronized Interaction: Common observations about traces

Model: Composed sets of (observation) traces



4.2 Buffered Data Flow: Asynchronous Models

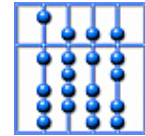
Goal: Apply denotational models to formalize notations for the behavior of high-level systems

- Distinction between system and environment
- Explicit sending and receiving of signals
- Decoupling of sender and receiver by non-blocking output

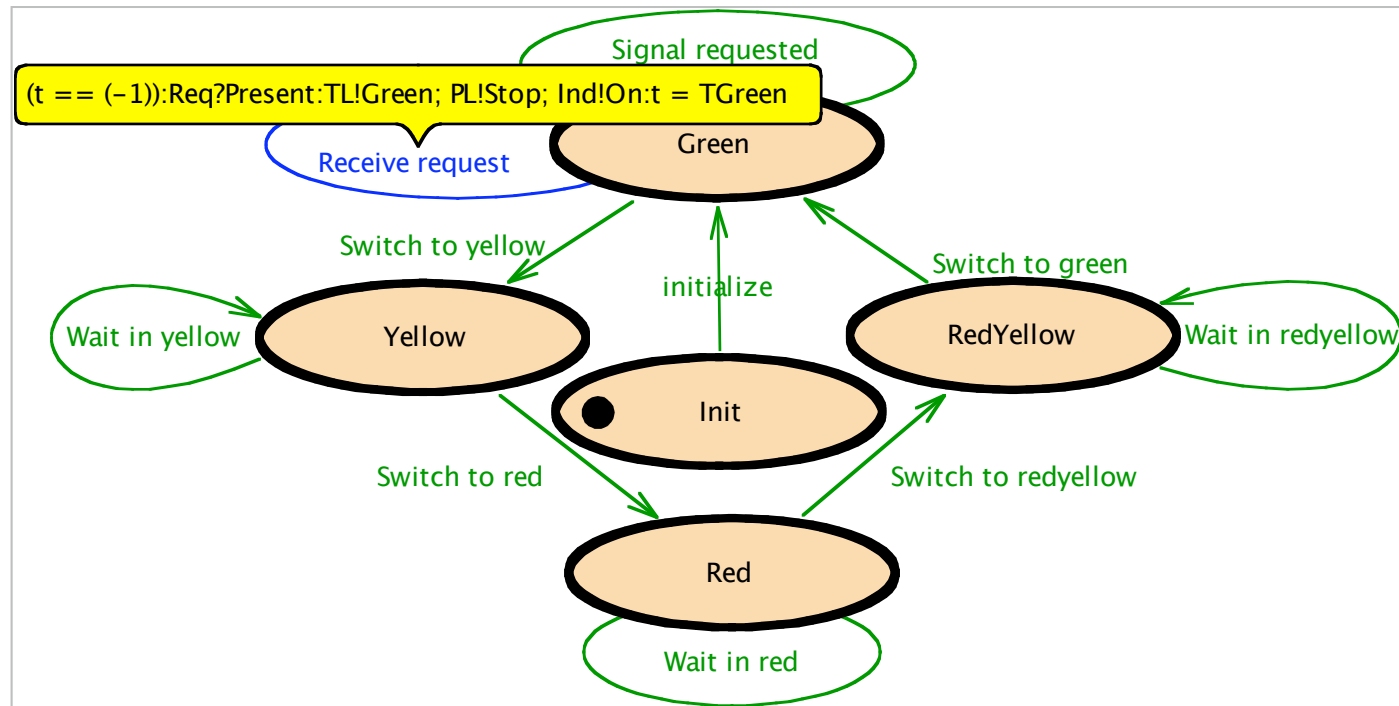
Concept: Asynchronous communication

Notations: Transition systems, tabular specifications, SDL

Model: Timed observation traces, untimed structured observation traces, untimed observation traces

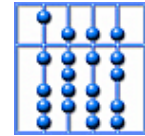


Notation: Transition System

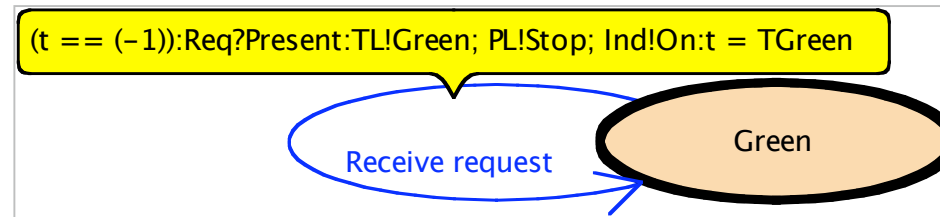


Transition: Describing atomic actions

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



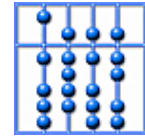
Concept: Execution Step



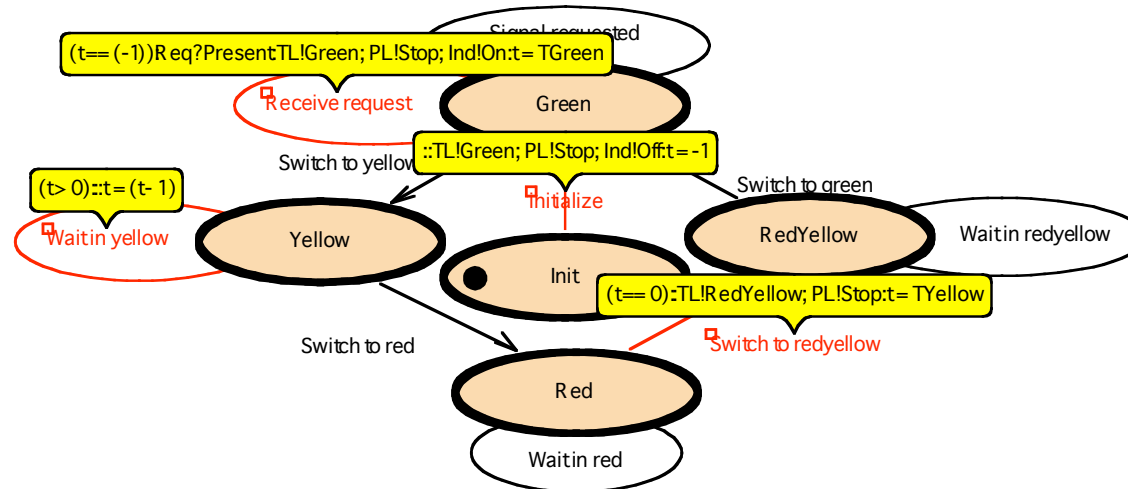
State	Green	Green	Green
t	-1	10	9
Req	Present	-	-
TL	-	Green	-
PL	-	Stop	-
Ind	-	On	-

Concept: Execution Step:

- Pair of observations about the system
- Variants: Immediate output/delayed output
 - Pre-Observation: Control State, Data State, Input Ports
 - Post-Observation: Output Ports, Data State, Control State

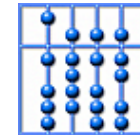


Example: Execution Trace of Transition System

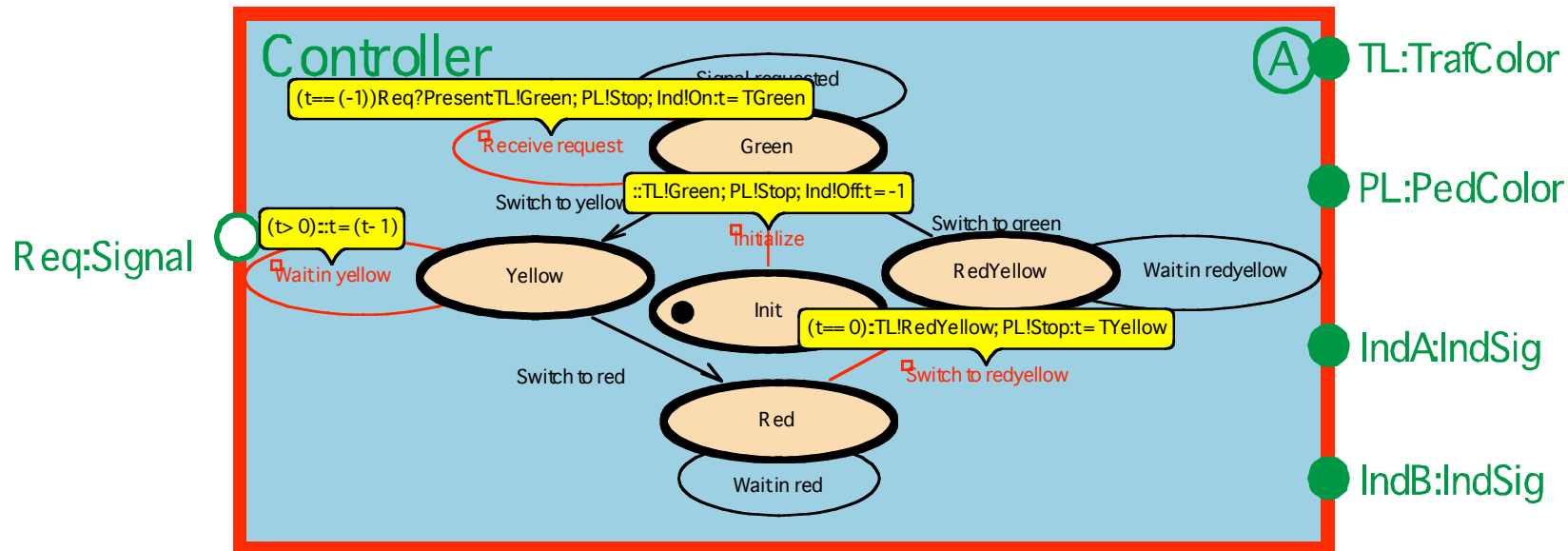


State	Init	Green	Green	...	Green	Yellow	...	Yellow	Red
t	-1	-1	10	...	0	10	...	0	20
Req	-	Present	-	-	-	-	-	-	-
TL	-	Green	Green	-	-	Yellow	-	-	Red
PL	-	Stop	Stop	-	-	-	-	-	Go
Ind	-	-	On	-	-	-	-	-	Off
	0	1	2	3					

Concept: Execution Trace = Sequence of consecutive execution steps

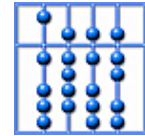


Example: Observation Trace of Transition System

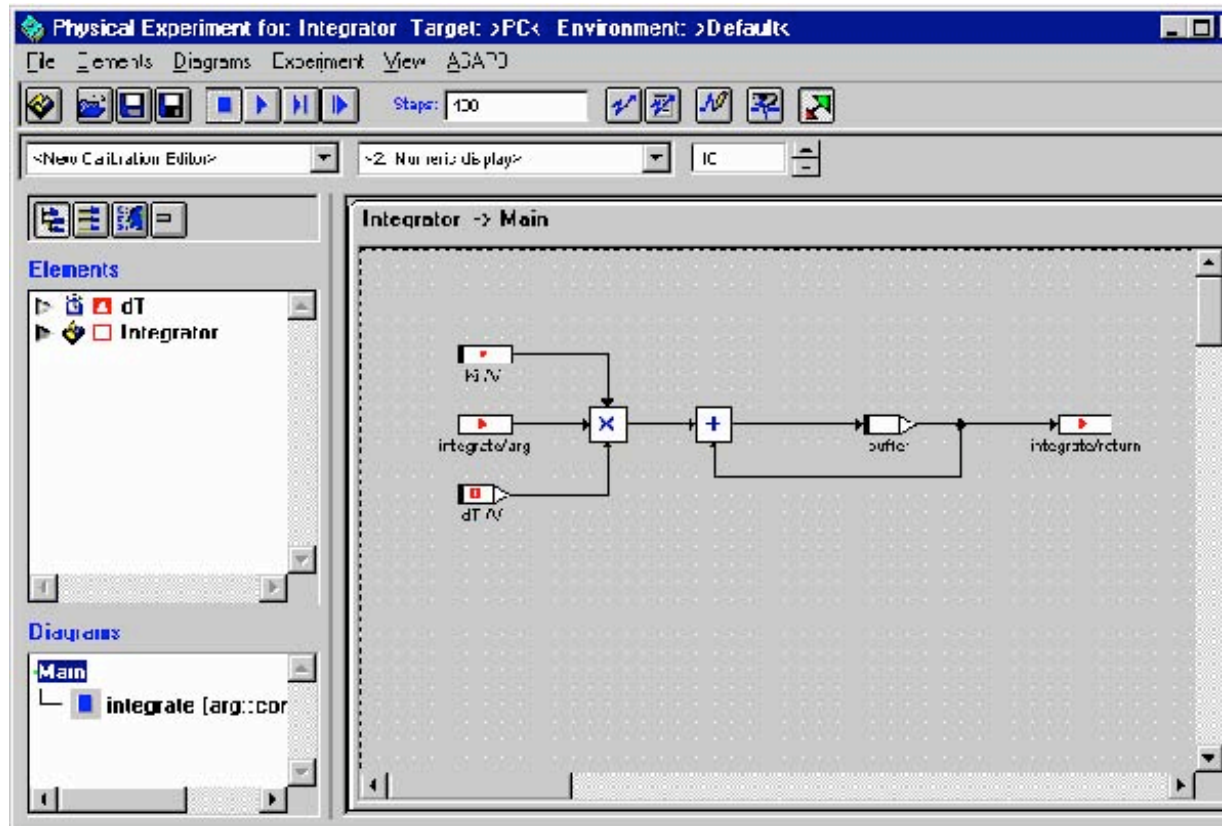


Req	-	Present	-	-	-	-	-	-	-
TL	-	Green	Green	-	-	Yellow	-	-	Red
PL	-	Stop	Stop	-	-	-	-	-	Go
Ind	-	-	On	-	-	-	-	-	Off
	0	1	2	3					

Concept: Observation Trace = Sequence of consecutive interface observations



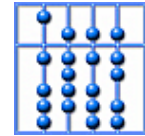
Notation: Data Flow Networks



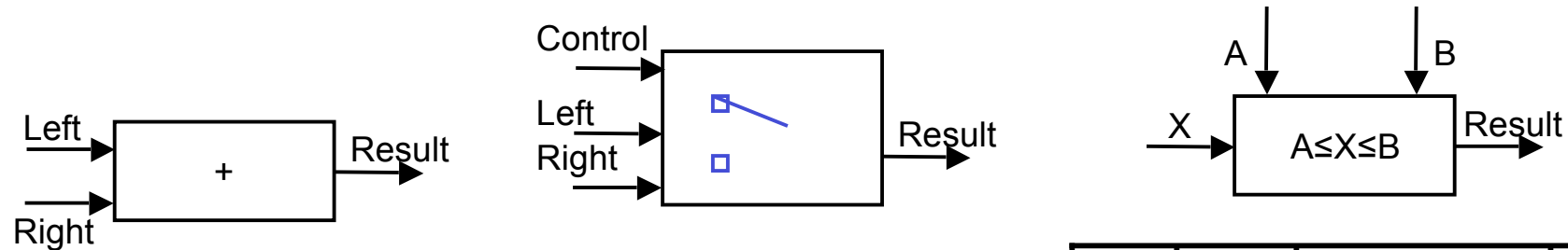
Example :
ASCET-SD

Data Flow Networks: Network of data flow nodes

- Nodes: Standard processing entities
- Communication: Directed, buffered channels



Notation: Table



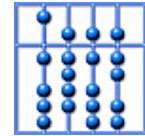
Left	Right	Result
x	y	x+y

Control	Left	Right	Result
true	x	-	x
false	-	y	y

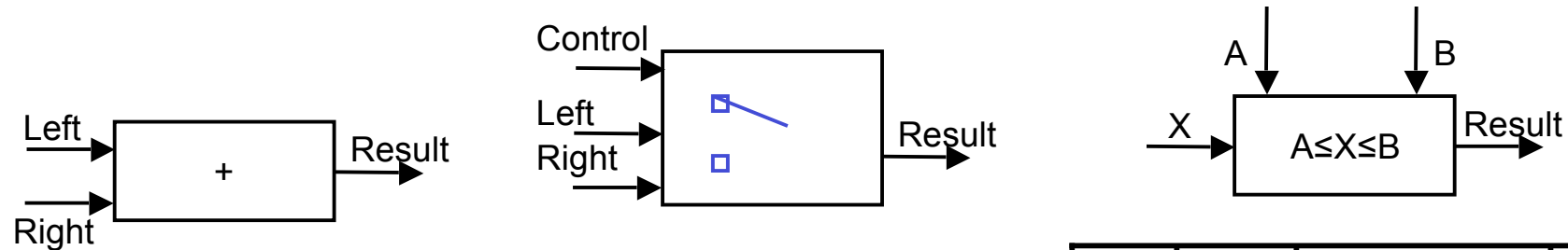
A	B	In	Result
min	max	$x < \min$	false
min	max	$\min \leq x \leq \max$	true
min	max	$\max < x$	false

Notation: Tabular description of data flow nodes

- Input: Combination of messages to be consumed
- Output: Combination of messages to be produced
- Execution scheme:
 - Consume inputs according to matching combination
 - Produce outputs according to corresponding combination



Formalization: Table



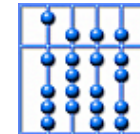
Left	Right	Result
x	y	x+y

Control	Left	Right	Result
true	x	-	x
false	-	y	y

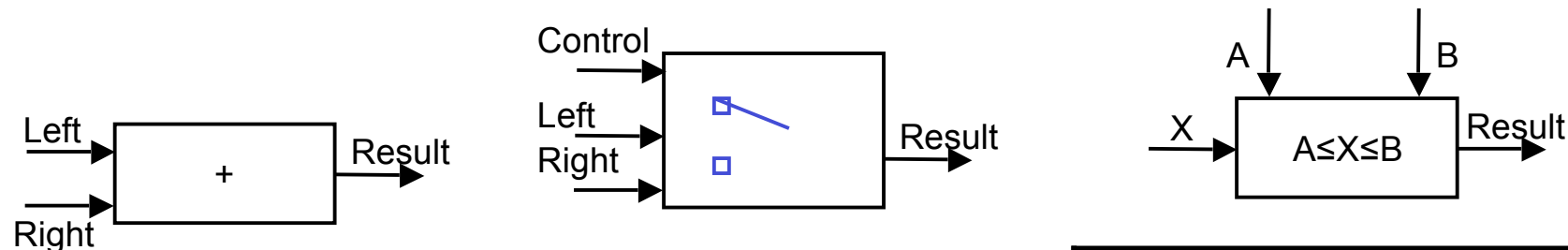
A	B	In	Result
min	max	$x < \min$	false
min	max	$\min \leq x \leq \max$	true
min	max	$\max < x$	false

Formalization: Untimed structured behaviors

- Model: Kahn function as behavioral set B of $(is_1, \dots, is_k, os_1, \dots, os_l)$
- Formalization: Table entry $(i_1, \dots, i_k, o_1, \dots, o_l)$
 - $(is_1, \dots, is_k, os_1, \dots, os_l) = (i_1 \cdot is'_1, \dots, i_k \cdot is'_k, o_1 \cdot os'_1, \dots, o_k \cdot os'_l)$
 - $(is_1, \dots, is_k, os_1, \dots, os_l) \in B$
 - Empty output for all missing table entries
- Example: Adder:
 - Missing entry: $(\langle \rangle, \langle \rangle, \langle \rangle)$,
 - Defined entries: $(1, 3, 4), (1 \cdot 2, 3 \cdot 4, 4 \cdot 6)$



Example: Untimed Structured Behavior



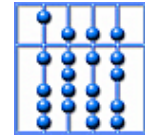
Left	Right	Result
x	y	x+y

Control	Left	Right	Result
true	x	-	x
false	-	y	y

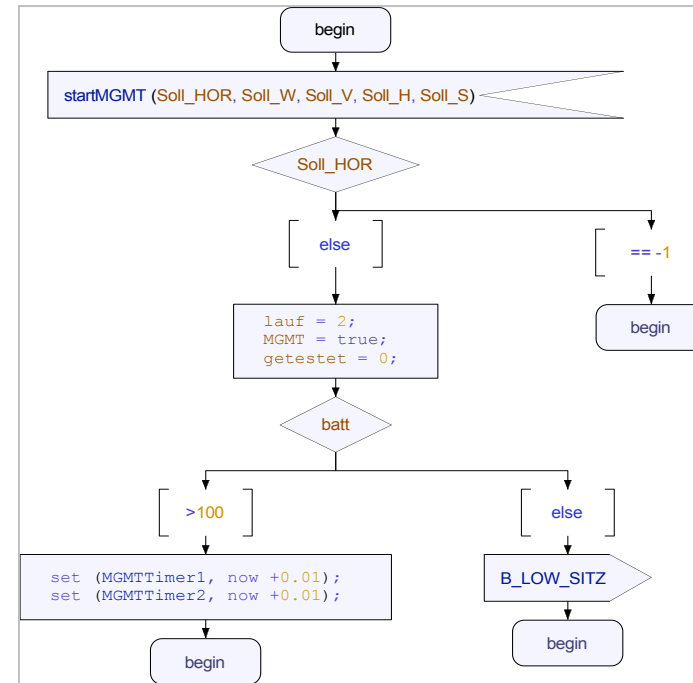
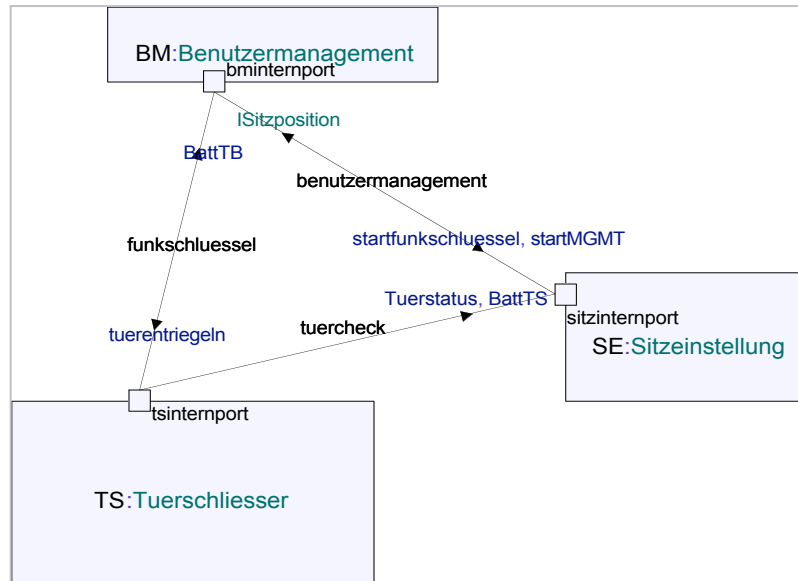
A	B	X	Result
min	max	$x < \min$	false
min	max	$\min \leq x \leq \max$	true
min	max	$\max < x$	false

Examples:

- Adder: (Left,Right,Result)
 ($\langle \rangle, \langle \rangle, \langle \rangle$), (1 • 2, $\langle \rangle, \langle \rangle$), (1 • 2, 3 • 4 • 5 • 6, 4 • 6)
- Multiplexer: (Control, Left,Right, Result)
 ($\langle \rangle, \langle \rangle, \langle \rangle, \langle \rangle$), (true • false • true, 1 • 2, 3 • 4 • 5 • 6, 1 • 3)
- Interval: (A,B,X,Result)
 ($\langle \rangle, \langle \rangle, \langle \rangle, \langle \rangle$), (-10 • -10, -15 • 7 • 12, $\langle \rangle, \langle \rangle$), (-10 • -10, -15 • 7 • 12, 10 • 10, false • true • false)

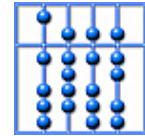


Notation: SDL



System: Network of communicating processes

- Processes: Extended state machines
- Communication: Directed, buffered channels



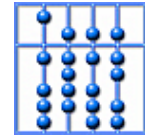
Notation: SDL

SDL: Specification and Description Language

- Application Focus: Telecommunication systems
- Standard: ITU, Part of UML 2.0 Standard
- Tool: Telelogic Tau

Language Basics:

- System: Networks of communication components (blocks)
- Communication: Directed, typed channels
- Messages: Typed, parametric signals
- Message exchange: Non-blocking send by buffered channels
- Components: Communicating state machines



Notation: SDL

Notational elements:

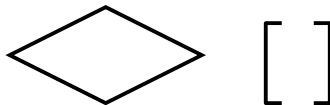
- Output: Sending a message



- Input: Receiving a message



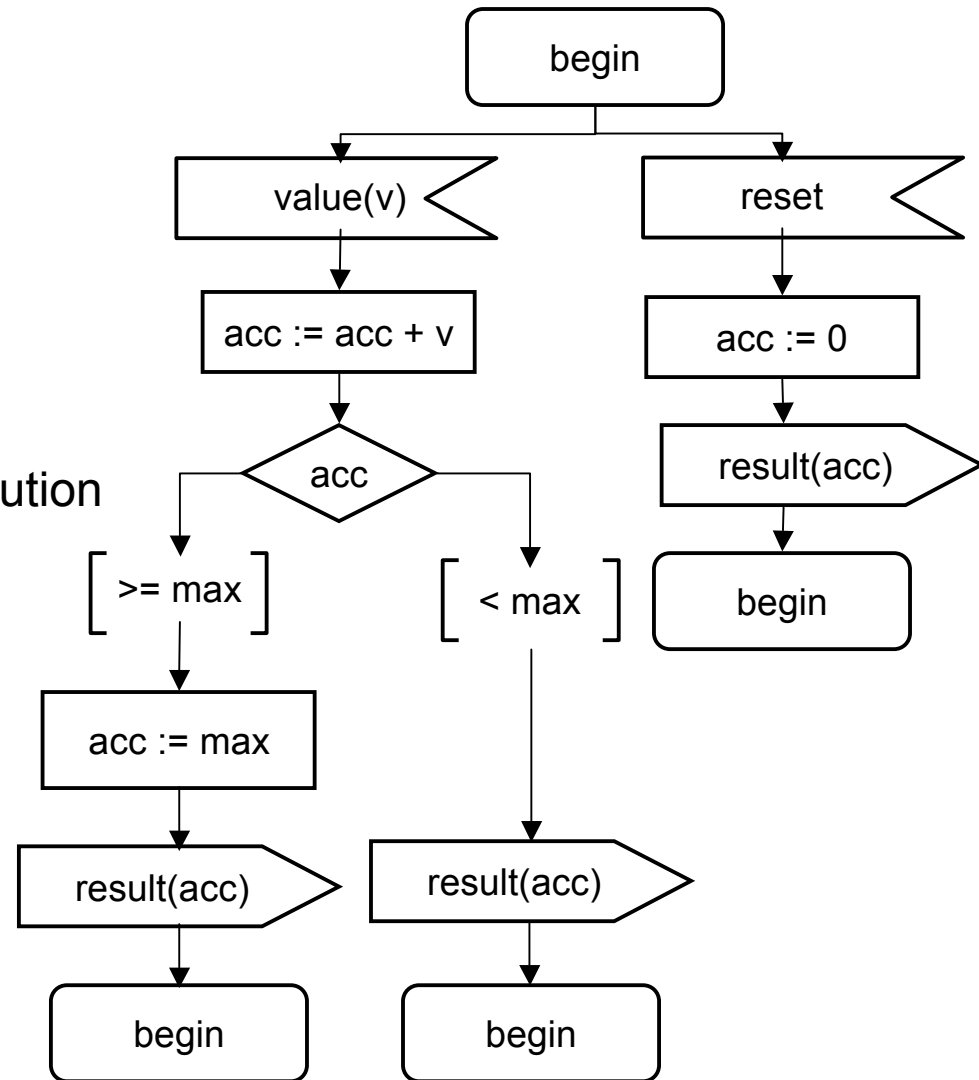
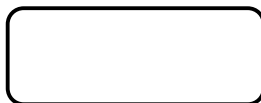
- Question/decision: Controlling execution

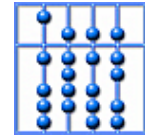


- Task: Defining assignments

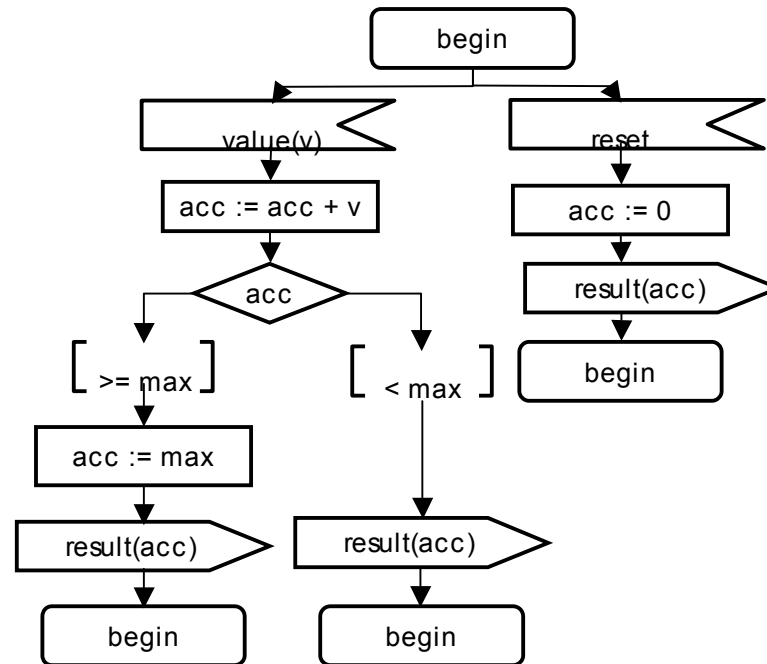


- States: Structuring control flow



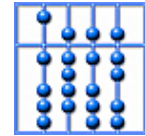


Model: Untimed Behavior



Formalization: Set of observation traces $t \in (\text{In} \cup \text{Out})^\omega$

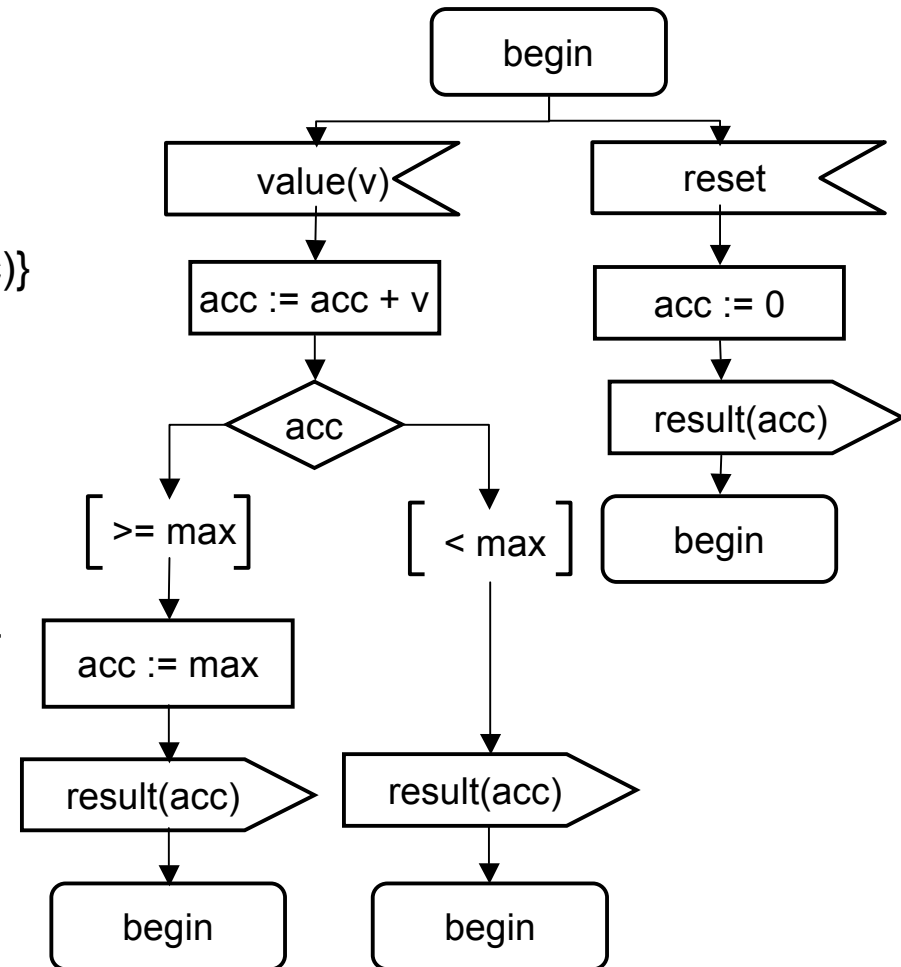
- Output: Sending a message M
 $t \in \text{In}^* \cdot M \cdot (\text{In} \cup \text{Out})^\omega$
- Input: Receiving messages M_1, \dots, M_k
 $t \in (\text{In} \setminus \{M_1, \dots, M_k\})^* \cdot \{M_1, \dots, M_k\} \cdot (\text{In} \cup \text{Out})^\omega$

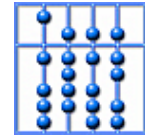


Example: Formalization

Formalization:

- Input: $In = \{ \text{value}(v) \mid v \in \text{Real} \} \cup \{ \text{reset} \} \cup \{ \text{stop} \}$
- Output: $Out = \{ \text{result}(v) \mid v \in \text{Real} \}$
- Behavior: Largest set B_1 with:
 - $B_1(\text{acc}) = \{ \langle \rangle \} \cup \{ \text{stop} \cdot t \mid t \in B_1(\text{acc}) \} \cup \{ \text{value}(v) \cdot t \mid t \in B_2(\text{acc}, v) \} \cup \{ \text{reset} \cdot t \mid t \in B_3(\text{acc}) \}$
 - $B_2(\text{acc}, v) = B_4(\text{acc} + v)$
 - $B_3(\text{acc}) = B_6(0)$
 - $B_4(\text{acc}) = \{ t \mid t \in B_5(\text{acc}) \wedge \text{acc} \geq \text{max} \} \cup \{ t \mid t \in B_6(\text{acc}) \wedge \text{acc} < \text{max} \}$
 - $B_5(\text{acc}) = B_6(\text{max})$
 - $B_6(\text{acc}) = \{ i \cdot \text{result}(\text{acc}) \cdot t \mid i \in In^* \wedge i \cdot t \in B_1(\text{acc}) \}$





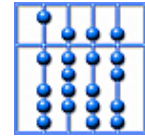
Example: Formalization

Initialized behavior: Largest set $B = B_1(0)$ with:

- $B_1(\text{acc}) = \{ \langle \rangle \} \cup \{ \text{stop} \cdot t \mid t \in B_1(\text{acc}) \}$
 $\cup \{ \text{value}(v) \cdot t \mid t \in B_2(\text{acc}, v) \} \cup \{ \text{reset} \cdot t \mid t \in B_3(\text{acc}) \}$
- $B_2(\text{acc}, v) = B_4(\text{acc} + v)$
- $B_3(\text{acc}) = B_6(0)$
- $B_4(\text{acc}) = \{ t \mid t \in B_5(\text{acc}) \wedge \text{acc} \geq \text{max} \} \cup \{ t \mid t \in B_6(\text{acc}) \wedge \text{acc} < \text{max} \}$
- $B_5(\text{acc}) = B_6(\text{max})$
- $B_6(\text{acc}) = \{ i \cdot \text{result}(\text{acc}) \cdot t \mid i \in \text{In}^* \wedge i \cdot t \in B_1(\text{acc}) \}$

Observations:

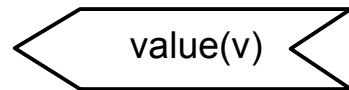
- $\langle \rangle$
- stop
- $\text{value}(1) \cdot \text{result}(1), \text{value}(2) \cdot \text{result}(2), \text{value}(3) \cdot \text{result}(3), \dots$
- $\text{reset} \cdot \text{result}(0)$
- $\text{stop} \cdot \text{stop}, \text{stop} \cdot \text{reset} \cdot \text{result}(0), \text{stop} \cdot \text{value}(1) \cdot \text{result}(1), \dots$
- $\text{value}(1) \cdot \text{result}(1) \cdot \text{stop}, \text{value}(1) \cdot \text{stop} \cdot \text{result}(1), \text{value}(1) \cdot \text{result}(1) \cdot \text{reset} \cdot \text{result}(0),$
 $\text{value}(1) \cdot \text{reset} \cdot \text{result}(1) \cdot \text{result}(0), \text{value}(1) \cdot \text{result}(1) \cdot \text{value}(2) \cdot \text{result}(2), \dots$
- $\text{reset} \cdot \text{result}(0) \cdot \text{stop}, \text{reset} \cdot \text{stop} \cdot \text{result}(0), \text{reset} \cdot \text{result}(0) \cdot \text{value}(1) \cdot \text{result}(1), \dots$
-



Notation: Extensions

Core model: unsuitable for reactive systems (e.g., switches)

- Order of messages: Merging of input messages
 - Problem: Implicit sequentialization
 - Extension: Save (re-insert in buffer)

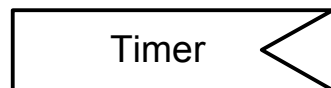


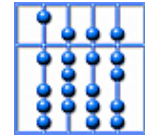
- Blocking input: Reaction on missing messages
 - Problem: Time-out actions
 - Extensions: Timers

- Setting timers

[set (Timer, now + 0.1)]

- Receiving timeouts





Questions

1. Describe an (angelic) merge component using SDL. What are its observation traces?
2. Is a similar component implementable using Kahn functions? Explain your answer.
3. What is the consequence of this observation for standard programming languages?
4. Define the observation traces for alarm clock process using timed observation traces. What are the equivalent untimed (structured) observation traces?
5. Do the structured observation traces of Question 5 describe a Kahn function?
6. Exercise: Implement sender and receiver of the alternating bit protocol using automata and SDL.