

Semantic Adaptation for Models of Computation

ACSD 2011 in Newcastle

Frédéric Boulanger

Cécile Hardebolle, Christophe Jacquet, Dominique Marcadet

Supélec E3S – Computer Science Department

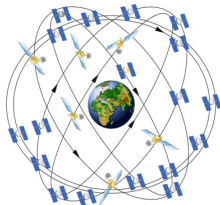
June 24 2011

- 1 Introduction
- 2 Languages and models of computation
- 3 Semantic adaptation
- 4 Conclusion

Context

Modeling and validation of embedded software systems

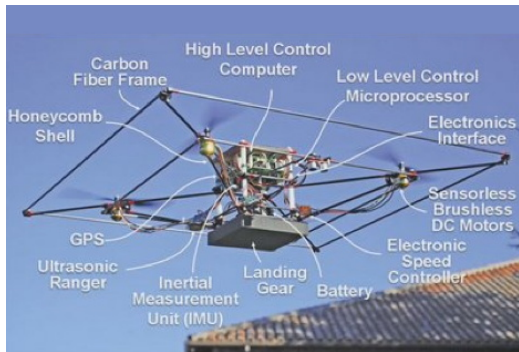
- Tight coupling with the environment
- Cost of errors



Heterogeneity

- Different specialties
- Different methods and different modeling tools

Starmac (Stanford/Berkeley)



Differential equations

Functions of a complex variable

Mechanics

Logics systems

Control science

Signal processing

Telecommunications

State machines

Aerodynamics

Energy

Discrete events

Sampled systems

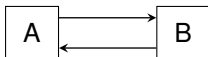
- 1 Introduction
- 2 Languages and models of computation
- 3 Semantic adaptation
- 4 Conclusion

Heterogeneous modeling

- Necessity to handle several modeling languages
- Principle : language \rightarrow model of computation (common syntax)

Model of computation

- Set of rules for composing the behavior of the components of a model.
Semantics of the structure of the model
- Algorithm for solving the rules \Rightarrow model of execution

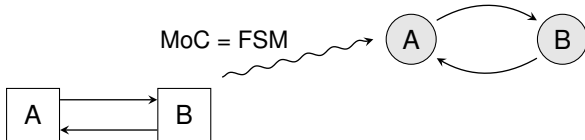


Heterogeneous modeling

- Necessity to handle several modeling languages
- Principle : language \rightarrow model of computation (common syntax)

Model of computation

- Set of rules for composing the behavior of the components of a model.
Semantics of the structure of the model
- Algorithm for solving the rules \Rightarrow model of execution

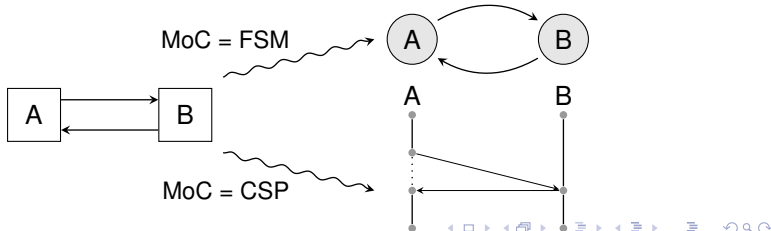


Heterogeneous modeling

- Necessity to handle several modeling languages
- Principle : language \rightarrow model of computation (common syntax)

Model of computation

- Set of rules for composing the behavior of the components of a model.
Semantics of the structure of the model
- Algorithm for solving the rules \Rightarrow model of execution



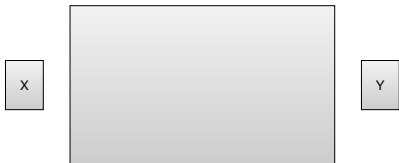
Modeling Framework

- Define models of computation (model semantics)
- Define semantic adaptation between heterogeneous models

Design principles

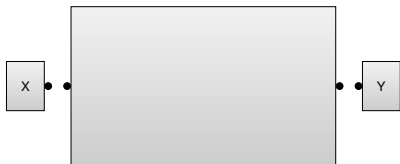
- Meta-model + generic execution engine:
 - schedule** choose the next component to be observed
 - update** update the interface of a component
 - propagate** propagate information toward other components
- Behavior of a model = fixed point of $propagate \circ update \circ schedule$
- Existence, unicity, reachability by iteration of the fixed point ?

Unit of behavior : **Block**, “black box” with an interface



Unit of behavior : **Block**, “black box” with an interface

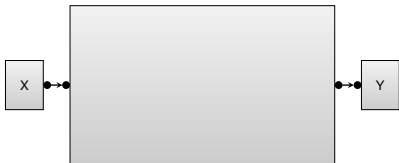
Unit of interface : **Pin**, for sending and getting information



Unit of behavior : **Block**, “black box” with an interface

Unit of interface : **Pin**, for sending and getting information

Structure : **Relations** between pins, semantics is defined by the MoC

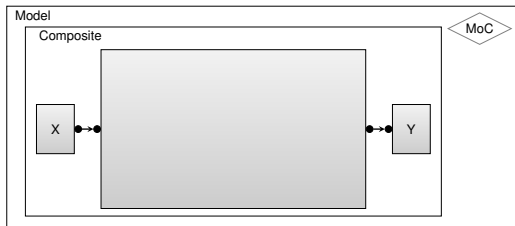


Unit of behavior : **Block**, “black box” with an interface

Unit of interface : **Pin**, for sending and getting information

Structure : **Relations** between pins, semantics is defined by the MoC

Model : **Model** = structure + MoC



Unit of behavior : **Block**, “black box” with an interface

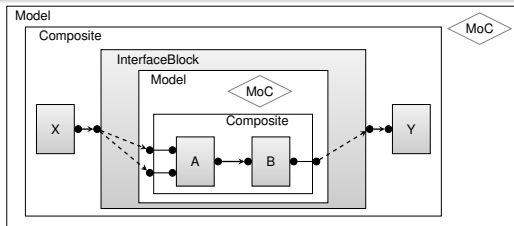
Unit of interface : **Pin**, for sending and getting information

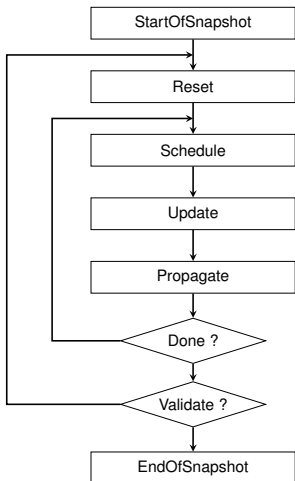
Structure : **Relations** between pins, semantics is defined by the MoC

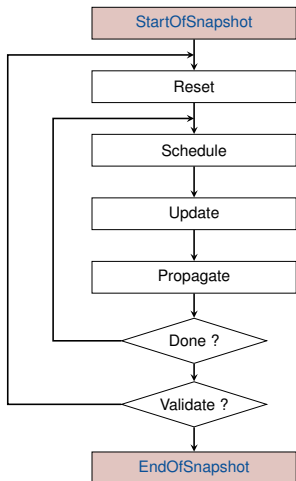
Model : **Model** = structure + MoC

Hierarchical heterogeneity

InterfaceBlock : behavior is described by a **Model**



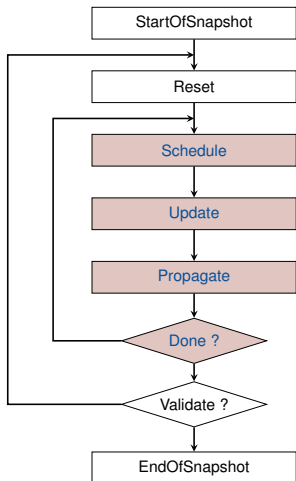




Behavior of a model = series of observations

Synchronous approach to the observation of models:

- no communication with the environment during the snapshot;
- no change in the internal state of the blocks.

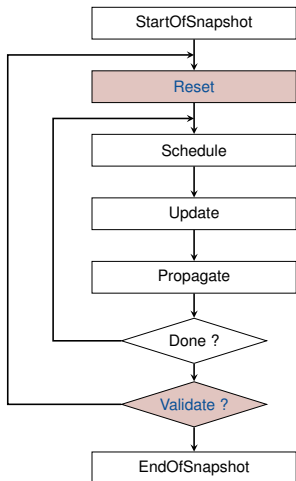


Computation of a fixed point by iteration:

- sequential observation of the blocks;
- update of their interface;
- propagation of the information according to the relations between pins.

Schedule, **Propagate** and **Done** are the three operations which define a MoC.

Update represents the observable behavior of the blocks.



It is possible to reject the fixed point which has been reached.

Search for another fixed point from different initial conditions.

- 1 Introduction
- 2 Languages and models of computation
- 3 Semantic adaptation**
- 4 Conclusion

Heterogeneous modeling \Rightarrow definition of different MoCs

Interaction between models that use **different** models of computation?

How to combine

- State machines
- Block diagrams
- Process networks
- Discrete systems
- Continuous systems

Heterogeneous modeling \Rightarrow definition of different MoCs

Interaction between models that use **different** models of computation?

How to combine

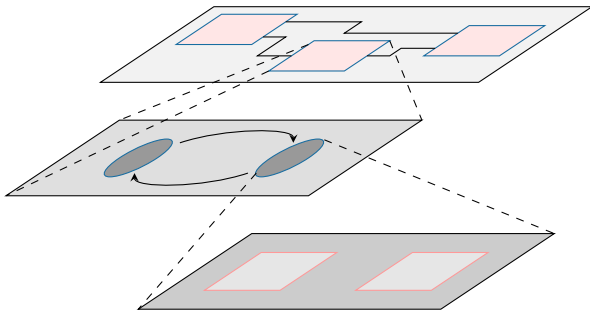
- State machines
- Block diagrams
- Process networks
- Discrete systems
- Continuous systems



How to tame the beast?

Hierarchical composition

- Approach used in Ptolemy and ModHel'X
- Each hierarchical level uses only one MoC
- MoCs are combined by pairs only



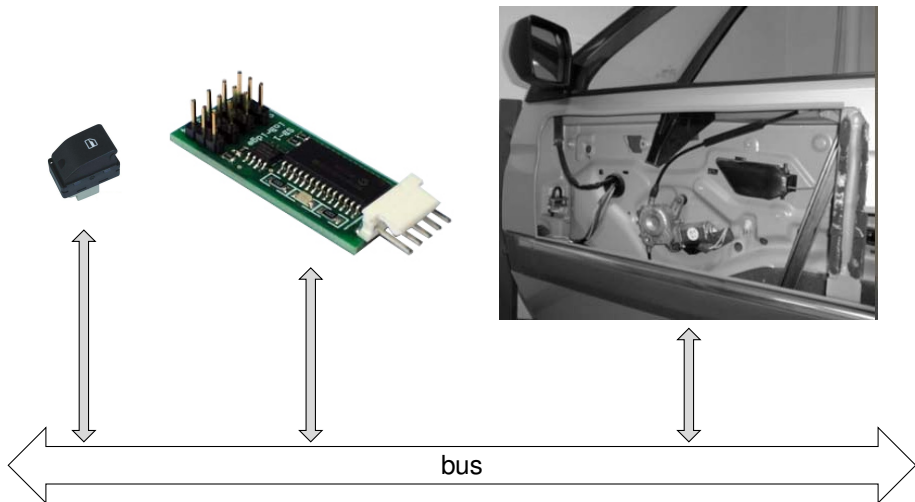
ModHel'X: semantic adaptation along three axes

- Adaptation of **data**:
different MoCs may use different kinds of data
- Adaptation of **time**:
different notions of time are used in different MoCs
- Adaptation of **control**:
instants at which a model should be observed depend on the MoC

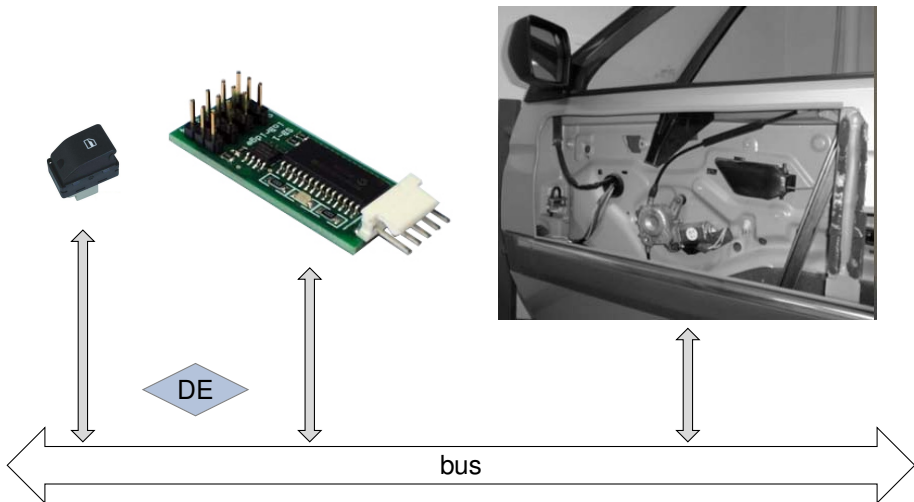
EDONA project

- Precise definition of MoCs from the automotive domain
- Design of configurable semantic adaptation patterns
- Example of a power window

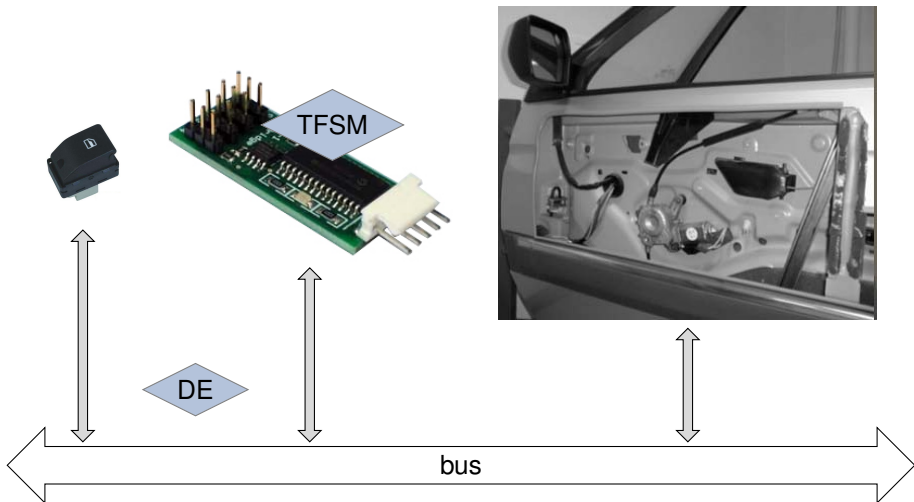
Example : power window



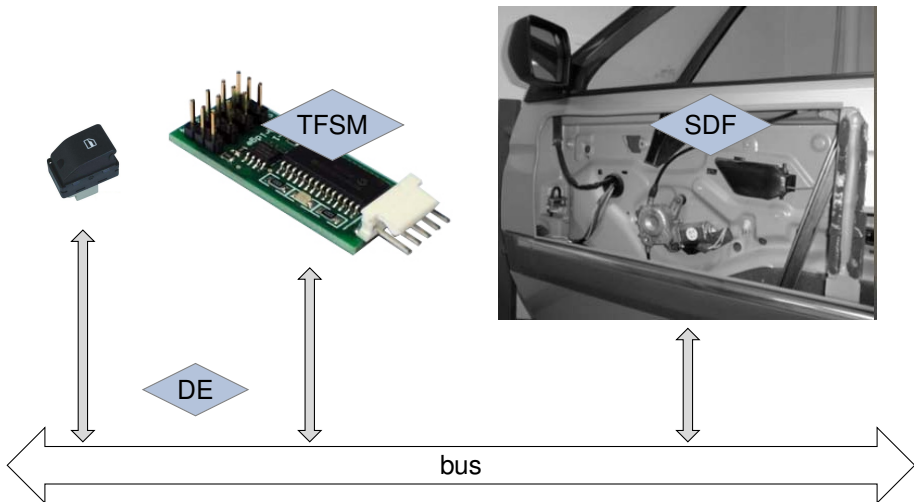
Example : power window



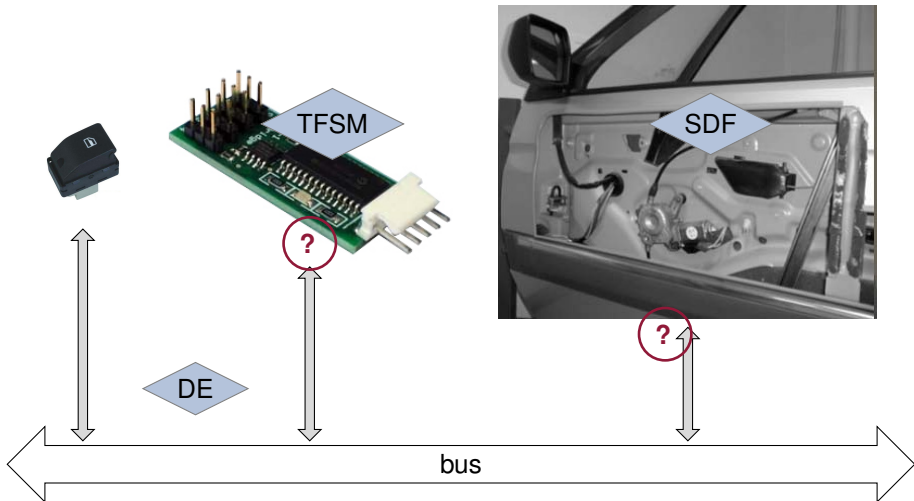
Example : power window



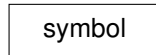
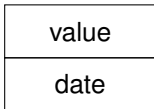
Example : power window



Example : power window



Adaptation of data between DE and TFSM

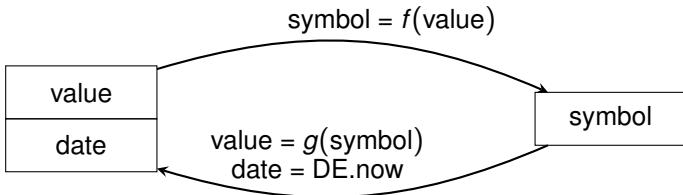


Adaptation of data between DE and TFSM

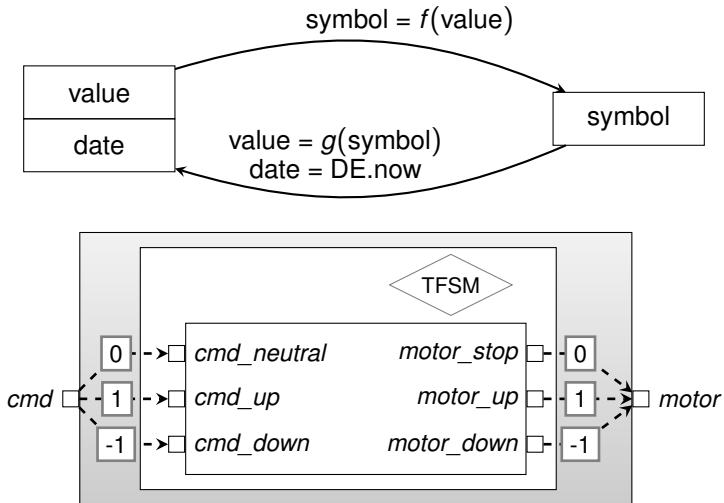
$$\text{symbol} = f(\text{value})$$



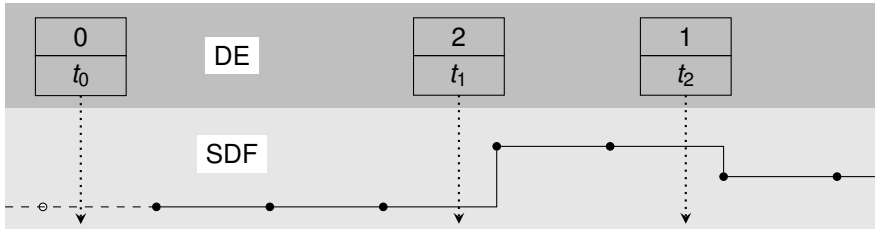
Adaptation of data between DE and TFSM



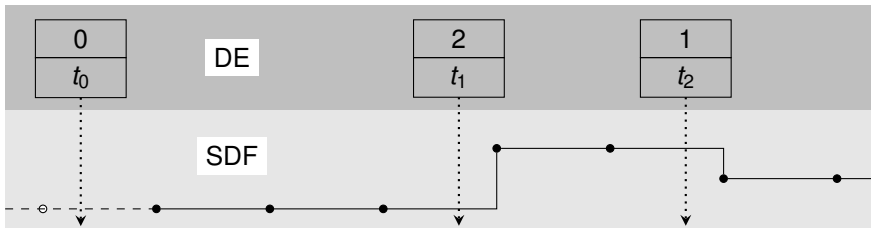
Adaptation of data between DE and TFSM



Adaptation of data between DE and SDF

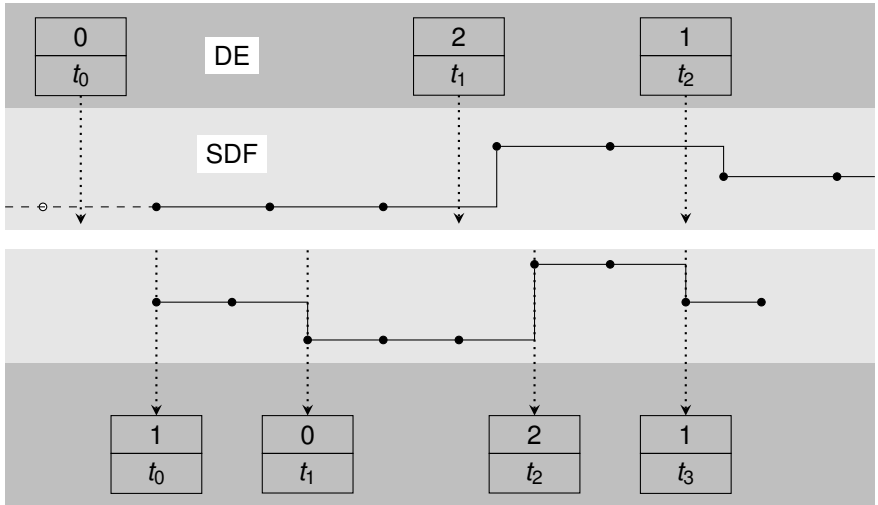


Adaptation of data between DE and SDF



- Relative position of events / samples \Rightarrow adaptation of time
- Presence of samples \Rightarrow adaptation of control

Adaptation of data between DE and SDF



Adaptation of time

Time in DE

- Time stamps in $\mathbb{R} \times \mathbb{N}$
- Synchronization, causality
- Controls when events are processed

Time in TFMSM

- Measure of the elapsed time in a state
- Time stamps have no special meaning

Adaptation DE – TFMSM

- Mapping from TFMSM durations to differences between DE time stamps
- Consequences on control

Adaptation of time

Time in DE

- Time stamps in $\mathbb{R} \times \mathbb{N}$
- Synchronization, causality
- Controls when events are processed


Time in SDF


- Series of samples, no real notion of date
- No notion of duration


Adaptation DE – SDF

- Give a DE time stamp to SDF data samples
- Sampling period, consequences on control

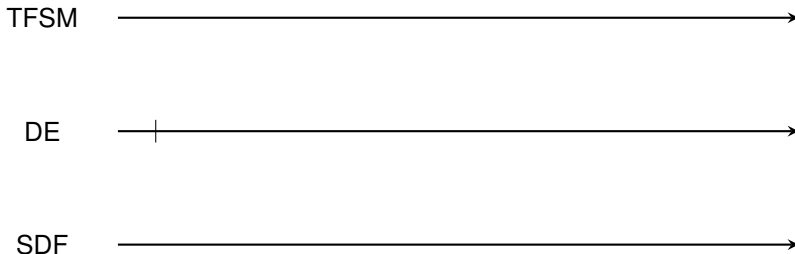
Adaptation of control

TFSM 

DE 

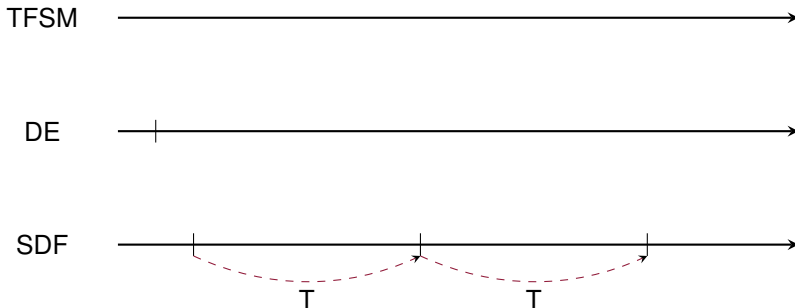
SDF 

Adaptation of control



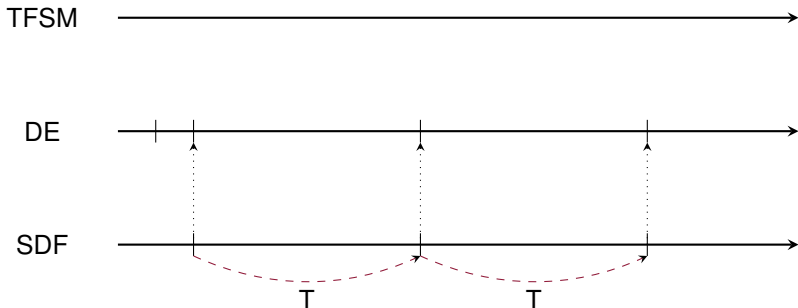
Event in DE, no input for the state machine

Adaptation of control



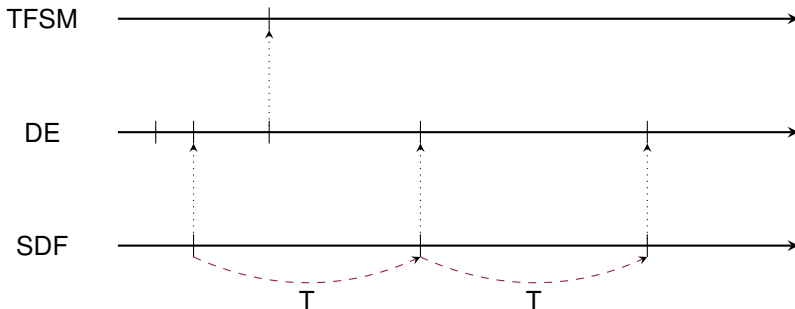
Periodic control for SDF

Adaptation of control



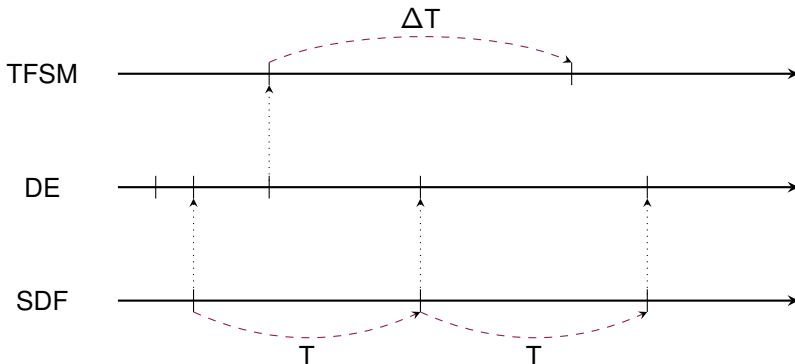
Periodic control for SDF \Rightarrow control in DE

Adaptation of control



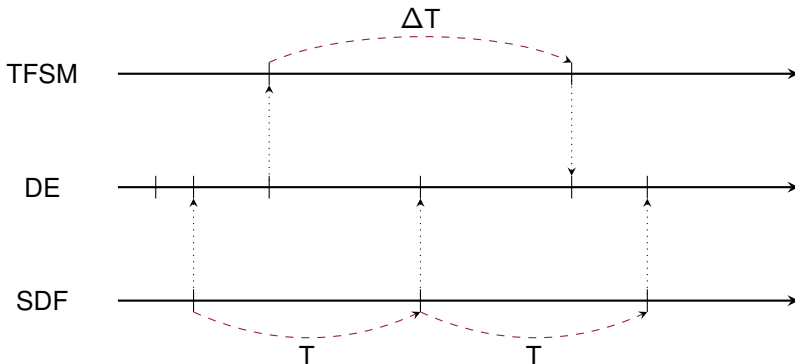
Event in DE, input for the state machine
 \Rightarrow control in TFSM

Adaptation of control



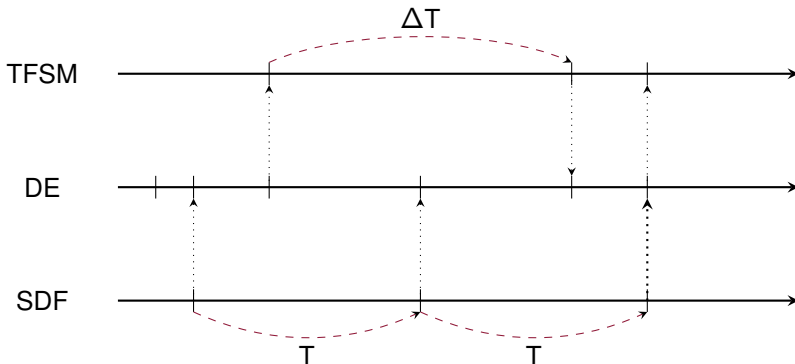
Firing of a timed transition in the state machine

Adaptation of control



Firing of a timed transition in the state machine
 \Rightarrow control in DE

Adaptation of control

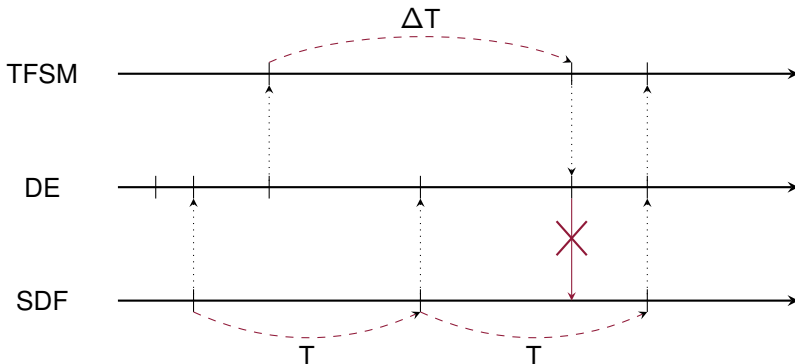


Periodic observation of SDF

⇒ control in DE + event for the state machine

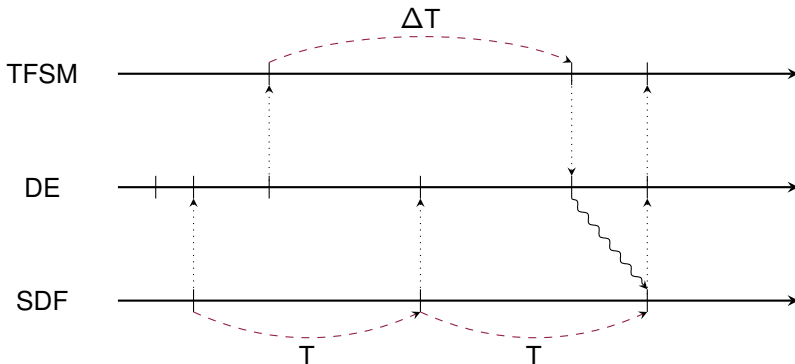
⇒ control in TFSM

Adaptation of control



Data never creates control for SDF

Adaptation of control



Data never creates control for SDF
 \Rightarrow data is memorized at the DE-SDF interface

Today

- Adaptation of data: configurable semantic adaptation patterns
- Adaptation of time: ad hoc handling in interface blocks
- Adaptation of control: constraints on the date of the next snapshot

The future

- Adaptation of data: special MoC?
- Adaptation of time and control: clock calculus
(Clock Constraint Specification Language)

- 1 Introduction
- 2 Languages and models of computation
- 3 Semantic adaptation
- 4 Conclusion**

Multi-paradigm modeling

- Accept heterogeneity
- Handle heterogeneity at the modeling level
- Many special purpose tools (Simulink, System C-AMS)

Our approach

- Unifying syntax for **heterogeneous models**
- Definition of **models of execution**
- Explicit description of the **semantic adaptation** between models

Key points

- Well defined semantics for models and combinations of models
- Joint use of already existing tools
- Modularity (black box approach)

Thanks for your attention

Questions?

▶ Thermostat

▶ Fuse

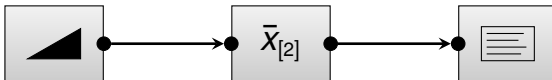
▶ Snapshot with non-strict blocks

▶ Super dense time

▶ Snapshot computation

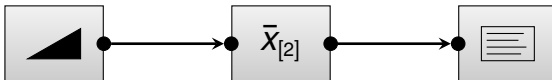
Synchronous data flow MoC

- Instantaneous propagation of data
- Consumption and production of a fixed number of data samples
- Snapshot = shortest production/consumption cycle



Synchronous data flow MoC

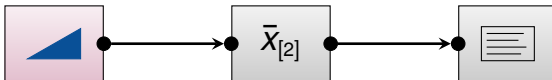
- Instantaneous propagation of data
- Consumption and production of a fixed number of data samples
- Snapshot = shortest production/consumption cycle



StartOfSnapshot

Synchronous data flow MoC

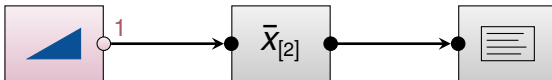
- Instantaneous propagation of data
- Consumption and production of a fixed number of data samples
- Snapshot = shortest production/consumption cycle



Schedule \rightarrow ramp

Synchronous data flow MoC

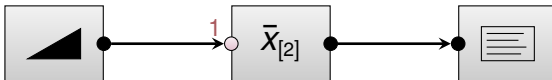
- Instantaneous propagation of data
- Consumption and production of a fixed number of data samples
- Snapshot = shortest production/consumption cycle



Update ramp

Synchronous data flow MoC

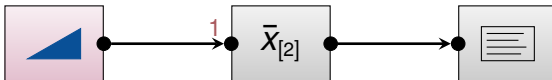
- Instantaneous propagation of data
- Consumption and production of a fixed number of data samples
- Snapshot = shortest production/consumption cycle



Propagate

Synchronous data flow MoC

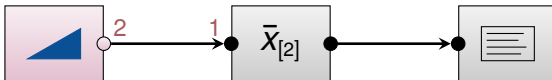
- Instantaneous propagation of data
- Consumption and production of a fixed number of data samples
- Snapshot = shortest production/consumption cycle



Schedule \rightarrow ramp

Synchronous data flow MoC

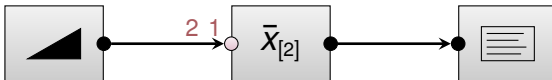
- Instantaneous propagation of data
- Consumption and production of a fixed number of data samples
- Snapshot = shortest production/consumption cycle



Update ramp

Synchronous data flow MoC

- Instantaneous propagation of data
- Consumption and production of a fixed number of data samples
- Snapshot = shortest production/consumption cycle



Propagate

Synchronous data flow MoC

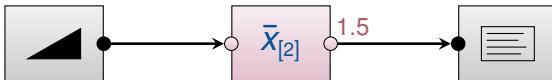
- Instantaneous propagation of data
- Consumption and production of a fixed number of data samples
- Snapshot = shortest production/consumption cycle



Schedule \rightarrow average

Synchronous data flow MoC

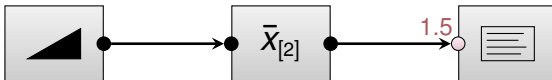
- Instantaneous propagation of data
- Consumption and production of a fixed number of data samples
- Snapshot = shortest production/consumption cycle



Update average

Synchronous data flow MoC

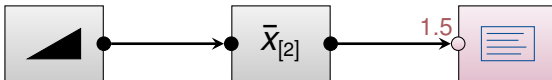
- Instantaneous propagation of data
- Consumption and production of a fixed number of data samples
- Snapshot = shortest production/consumption cycle



Propagate

Synchronous data flow MoC

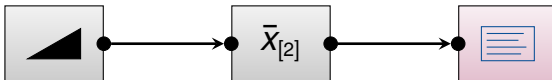
- Instantaneous propagation of data
- Consumption and production of a fixed number of data samples
- Snapshot = shortest production/consumption cycle



Schedule \rightarrow display

Synchronous data flow MoC

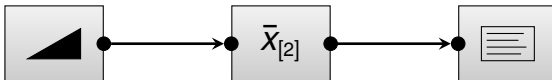
- Instantaneous propagation of data
- Consumption and production of a fixed number of data samples
- Snapshot = shortest production/consumption cycle



Update display

Synchronous data flow MoC

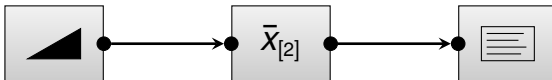
- Instantaneous propagation of data
- Consumption and production of a fixed number of data samples
- Snapshot = shortest production/consumption cycle



Done

Synchronous data flow MoC

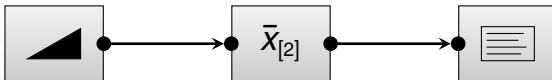
- Instantaneous propagation of data
- Consumption and production of a fixed number of data samples
- Snapshot = shortest production/consumption cycle



Validate

Synchronous data flow MoC

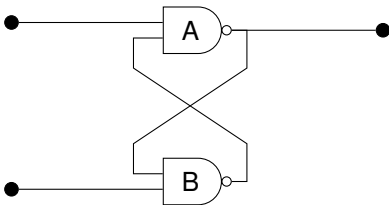
- Instantaneous propagation of data
- Consumption and production of a fixed number of data samples
- Snapshot = shortest production/consumption cycle



EndOfSnapshot

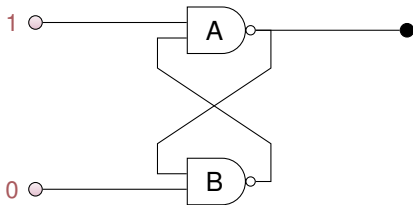
Synchronous reactive MoC

- Instantaneous propagation of data
- Non-strict blocks (may react to partial inputs)
- Snapshot = all signals are known



Synchronous reactive MoC

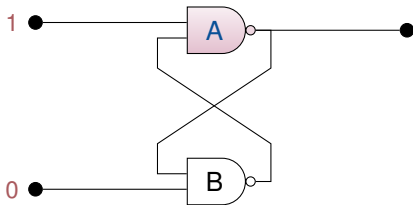
- Instantaneous propagation of data
- Non-strict blocks (may react to partial inputs)
- Snapshot = all signals are known



StartOfSnapshot

Synchronous reactive MoC

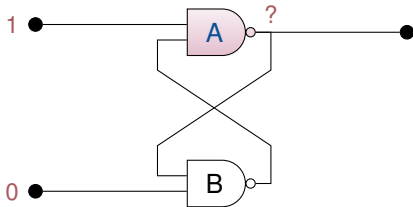
- Instantaneous propagation of data
- Non-strict blocks (may react to partial inputs)
- Snapshot = all signals are known



Schedule \rightarrow A

Synchronous reactive MoC

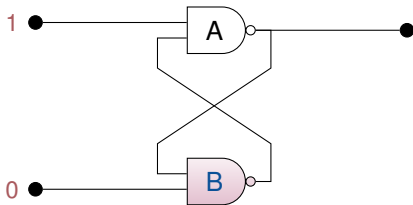
- Instantaneous propagation of data
- Non-strict blocks (may react to partial inputs)
- Snapshot = all signals are known



Update A

Synchronous reactive MoC

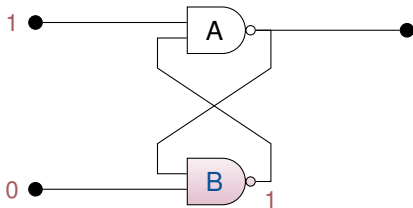
- Instantaneous propagation of data
- Non-strict blocks (may react to partial inputs)
- Snapshot = all signals are known



Schedule \rightarrow B

Synchronous reactive MoC

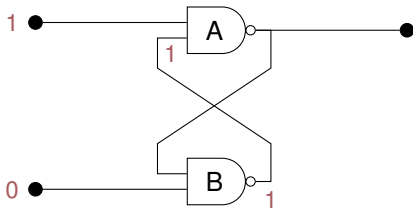
- Instantaneous propagation of data
- Non-strict blocks (may react to partial inputs)
- Snapshot = all signals are known



Update B

Synchronous reactive MoC

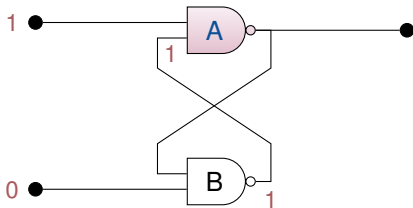
- Instantaneous propagation of data
- Non-strict blocks (may react to partial inputs)
- Snapshot = all signals are known



Propagate

Synchronous reactive MoC

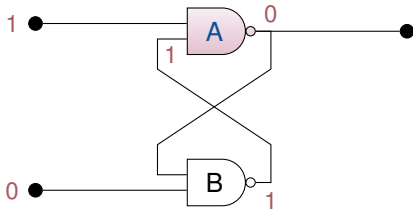
- Instantaneous propagation of data
- Non-strict blocks (may react to partial inputs)
- Snapshot = all signals are known



Schedule \rightarrow A

Synchronous reactive MoC

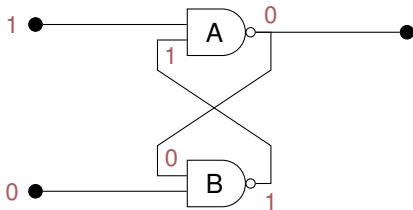
- Instantaneous propagation of data
- Non-strict blocks (may react to partial inputs)
- Snapshot = all signals are known



Update A

Synchronous reactive MoC

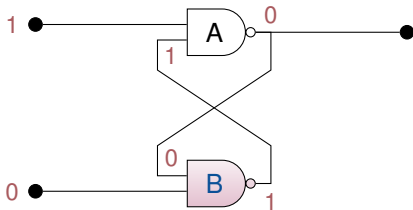
- Instantaneous propagation of data
- Non-strict blocks (may react to partial inputs)
- Snapshot = all signals are known



Propagate

Synchronous reactive MoC

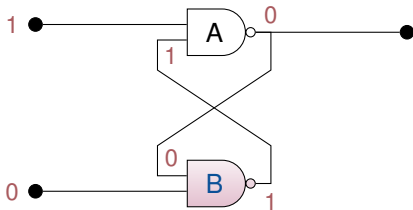
- Instantaneous propagation of data
- Non-strict blocks (may react to partial inputs)
- Snapshot = all signals are known



Schedule \rightarrow B

Synchronous reactive MoC

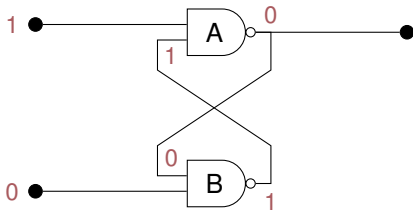
- Instantaneous propagation of data
- Non-strict blocks (may react to partial inputs)
- Snapshot = all signals are known



Update B

Synchronous reactive MoC

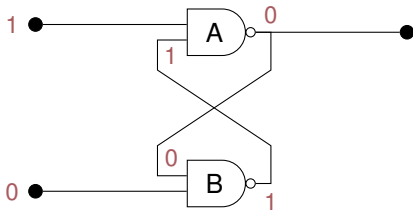
- Instantaneous propagation of data
- Non-strict blocks (may react to partial inputs)
- Snapshot = all signals are known



Done

Synchronous reactive MoC

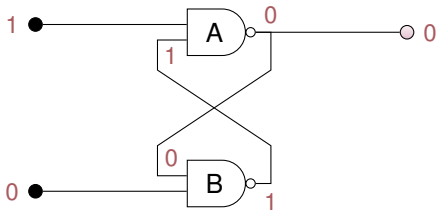
- Instantaneous propagation of data
- Non-strict blocks (may react to partial inputs)
- Snapshot = all signals are known



Validate

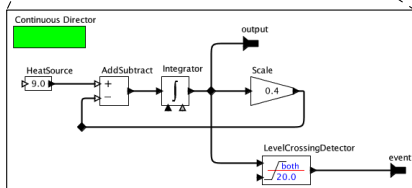
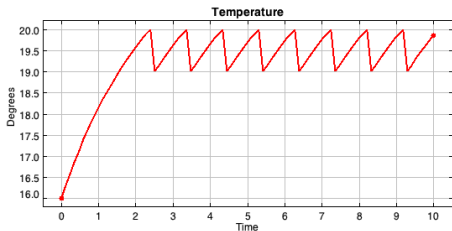
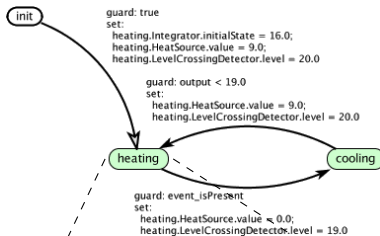
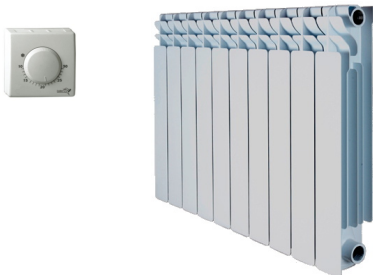
Synchronous reactive MoC

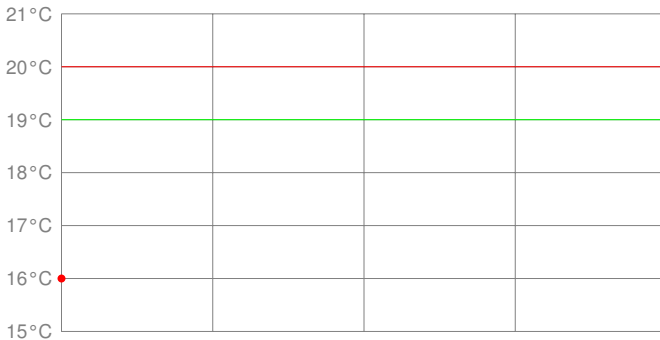
- Instantaneous propagation of data
- Non-strict blocks (may react to partial inputs)
- Snapshot = all signals are known



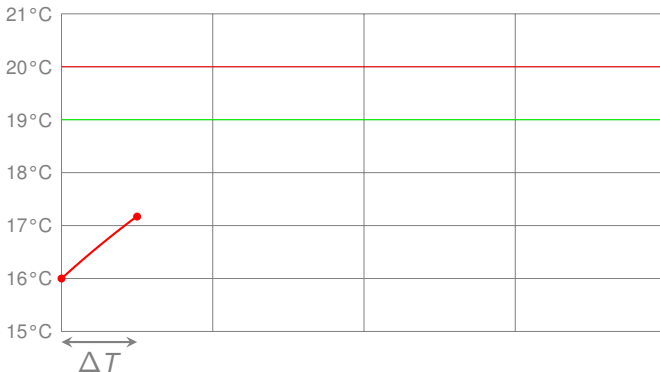
EndOfSnapshot

Example of non validation in CT





State = heating



State = heating

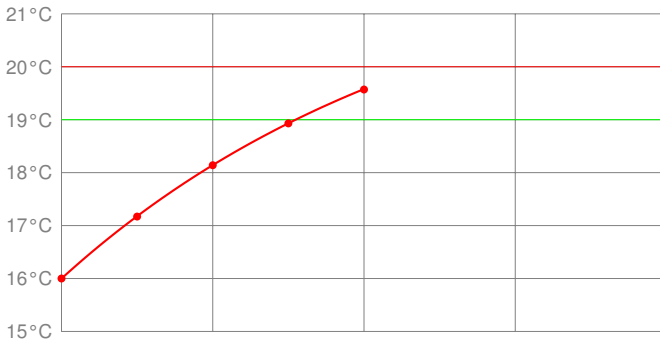
ΔT determined according to the precision on T



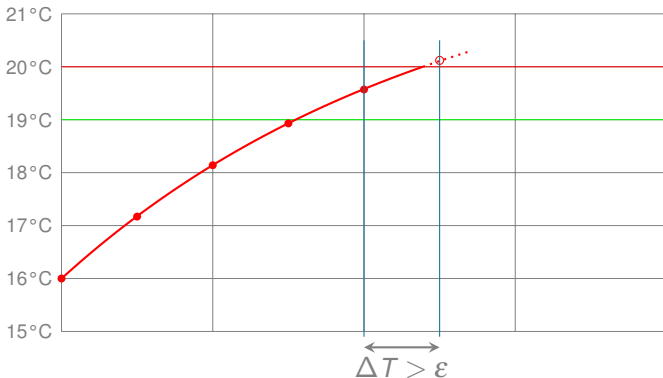
State = heating



State = heating

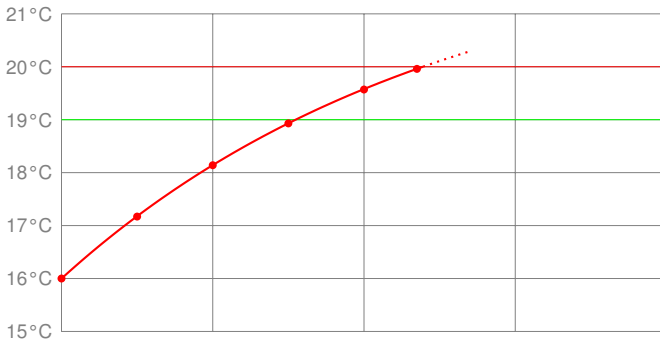


State = heating

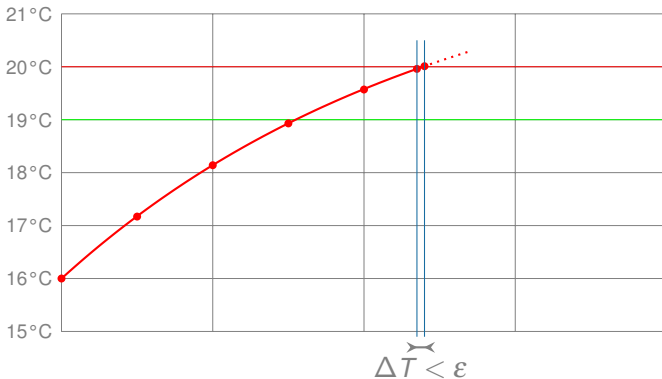


State = heating

The upper threshold is reached, but we don't know when.
The snapshot is not valid



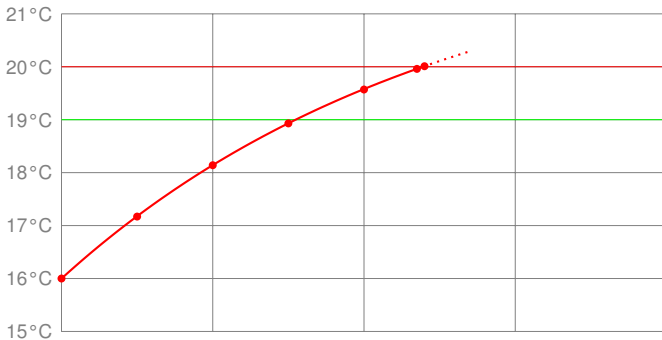
State = heating



State = heating

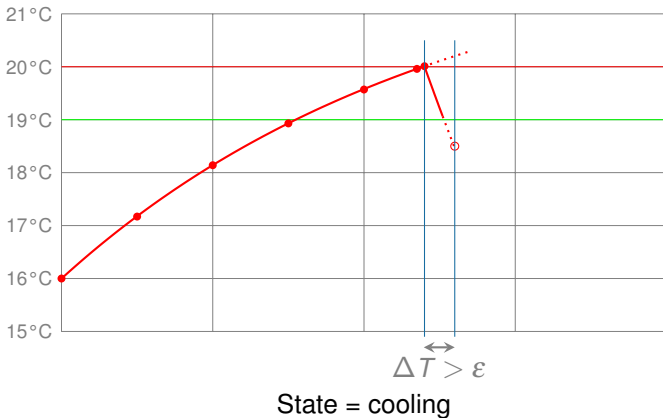
The upper threshold is reached, the data is precise enough.

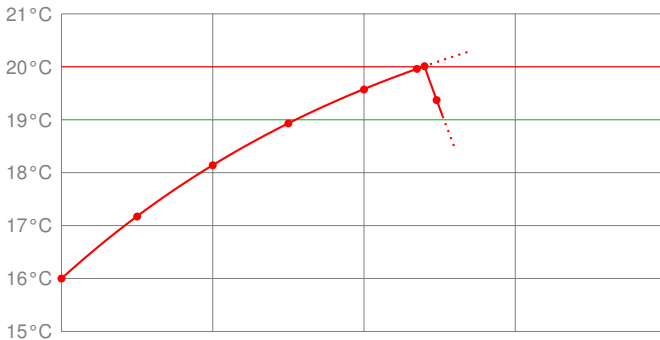
State changes



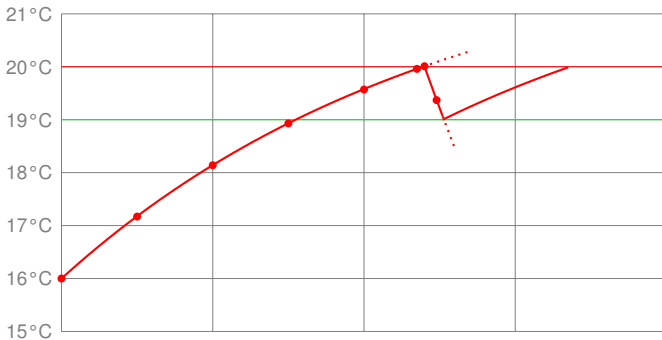
State = cooling

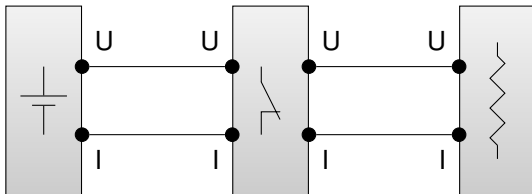
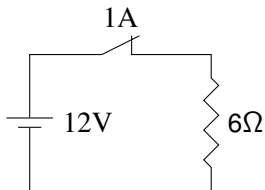
Example of non validation in CT

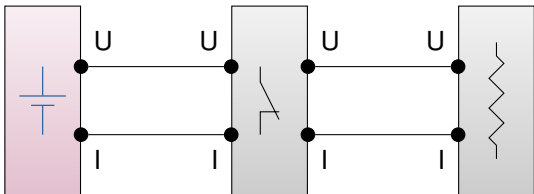
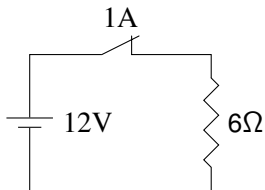




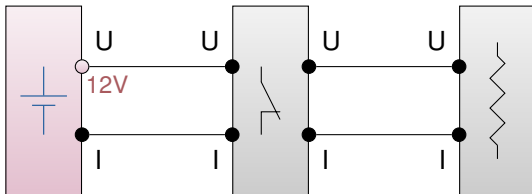
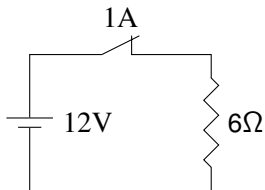
State = cooling



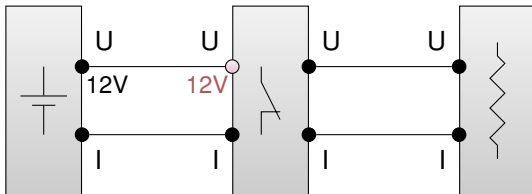
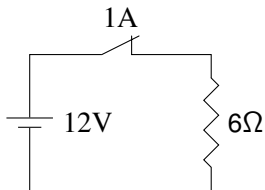




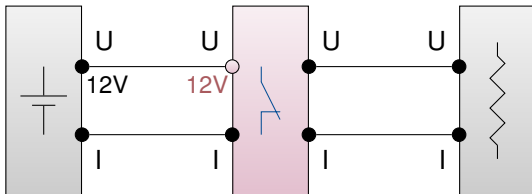
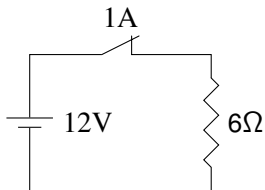
Schedule → battery



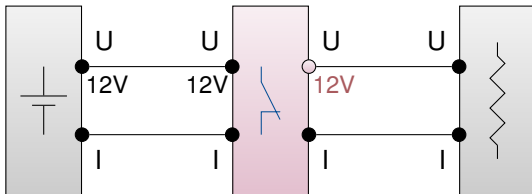
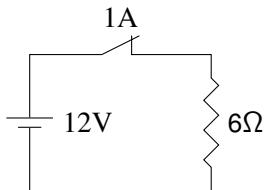
Update battery



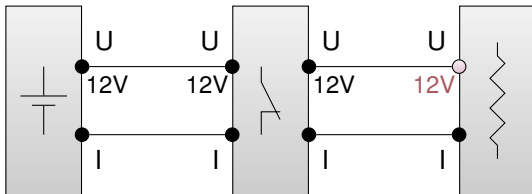
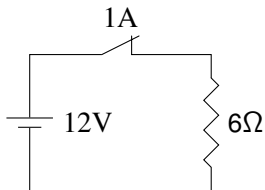
Propagate



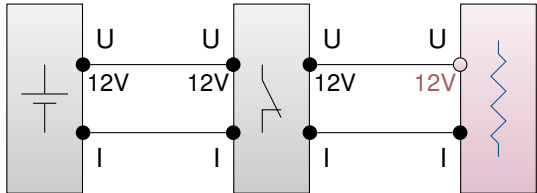
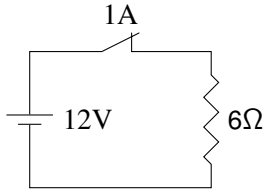
Schedule \rightarrow fuse



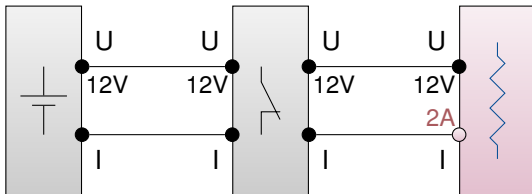
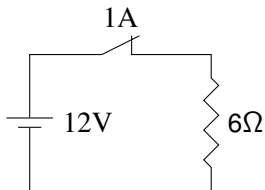
Update fuse



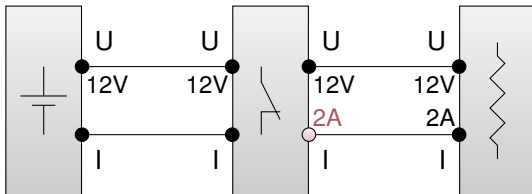
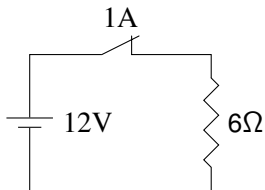
Propagate



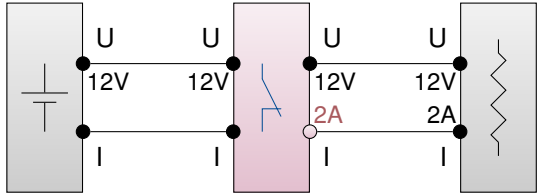
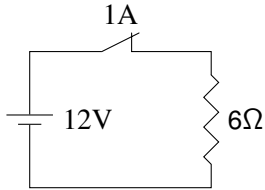
Schedule \rightarrow resistor



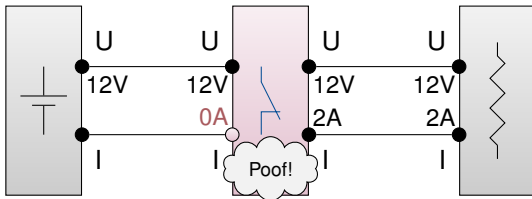
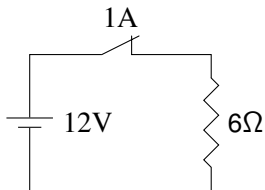
Update resistor



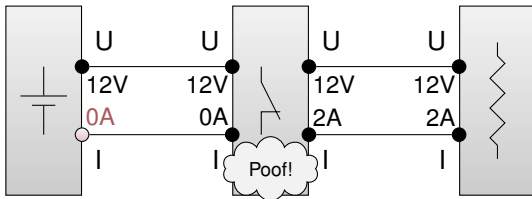
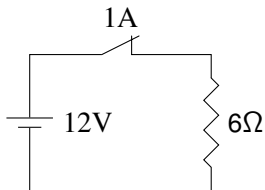
Propagate



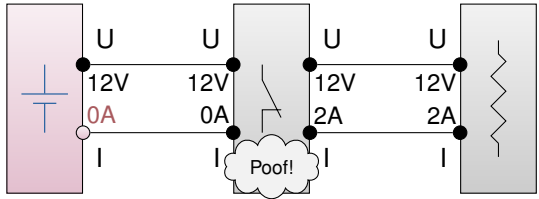
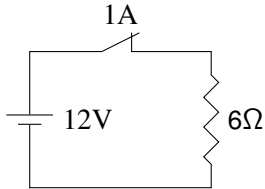
Schedule → fuse



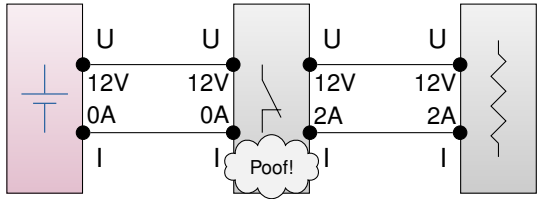
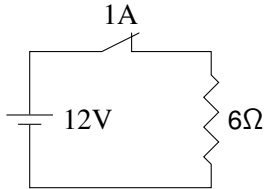
Update fuse



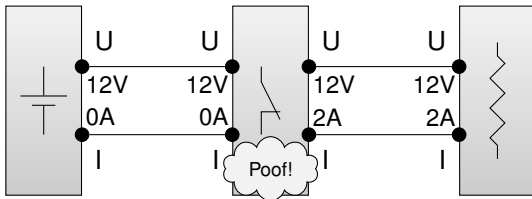
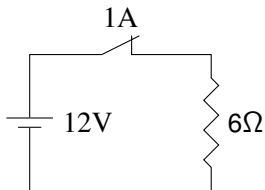
Propagate



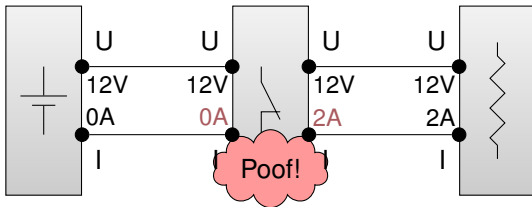
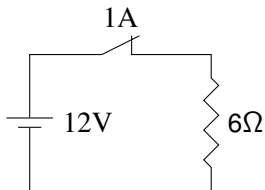
Schedule → battery



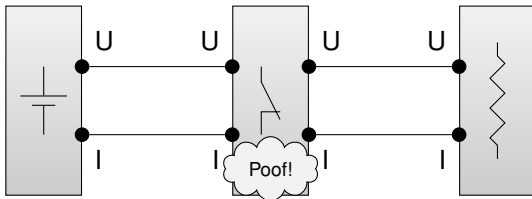
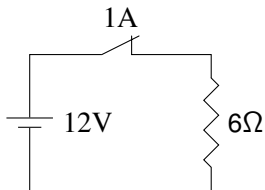
Update battery



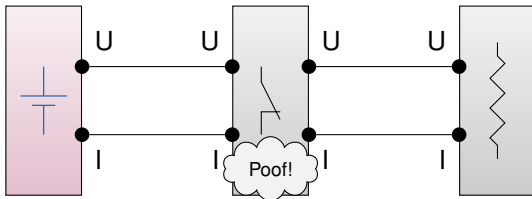
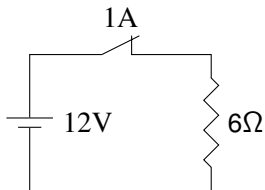
Done



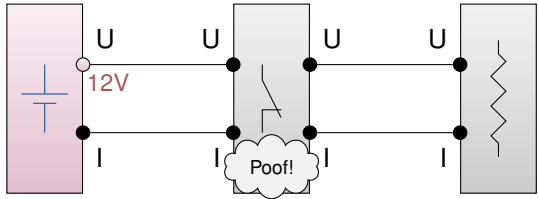
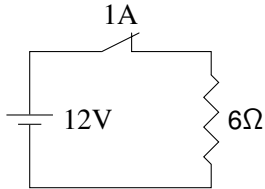
Validate



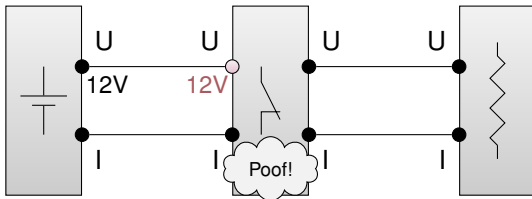
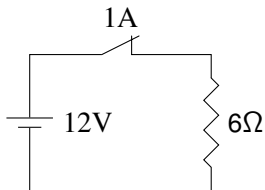
Reset



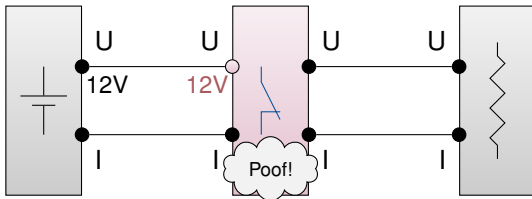
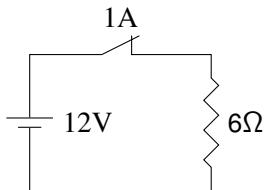
Schedule \rightarrow battery



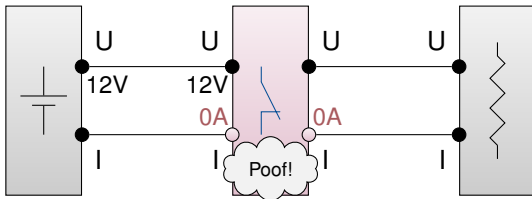
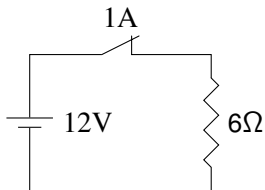
Update battery



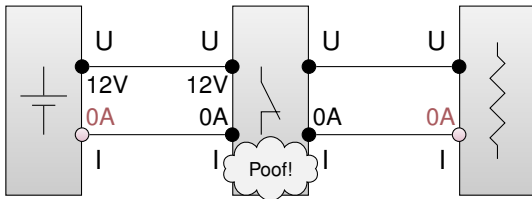
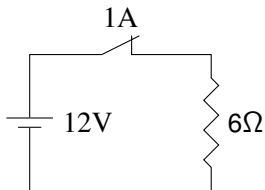
Propagate



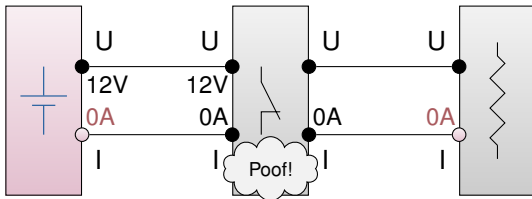
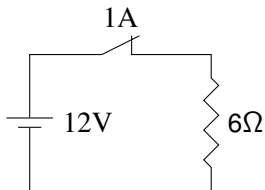
Schedule → fuse



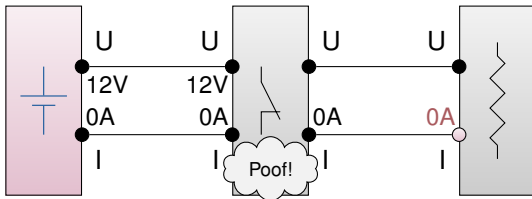
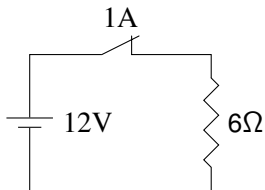
Update fuse



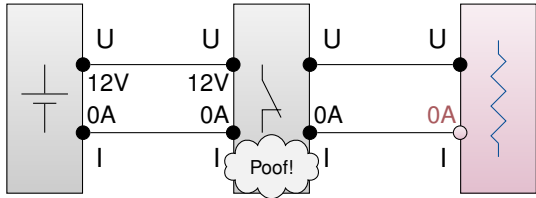
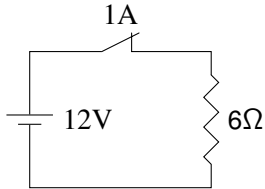
Propagate



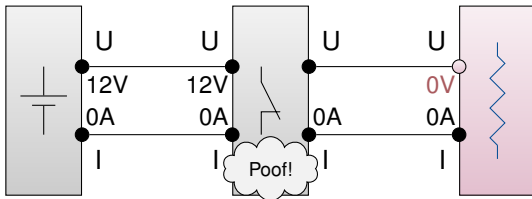
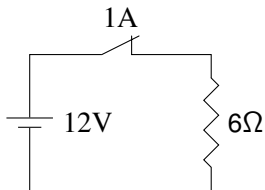
Schedule → battery



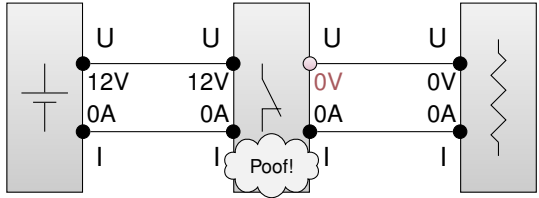
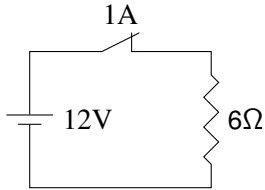
Update battery



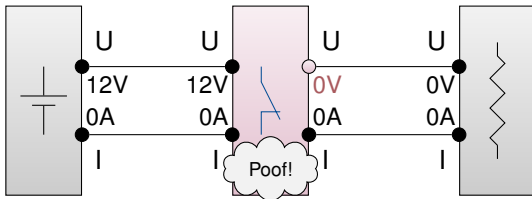
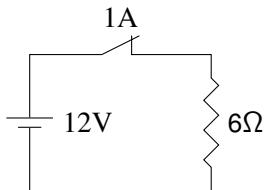
Schedule → resistor



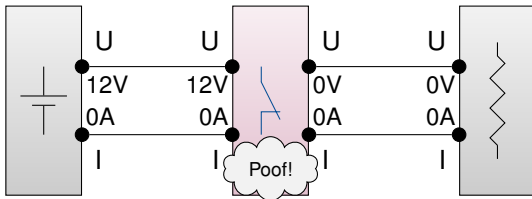
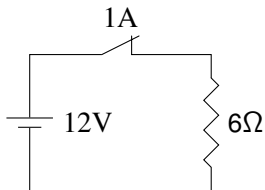
Update resistor



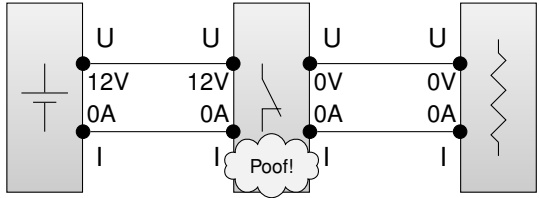
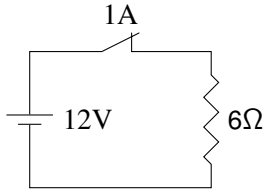
Propagate



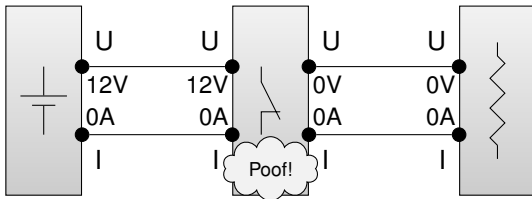
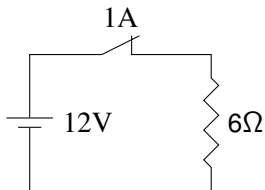
Schedule → fuse



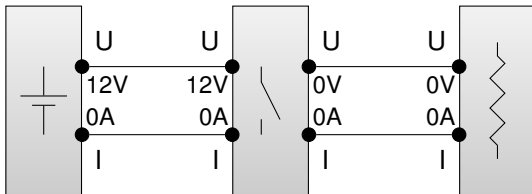
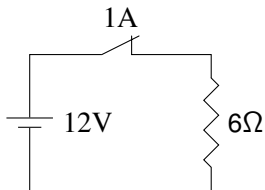
Update fuse



Done



Validate



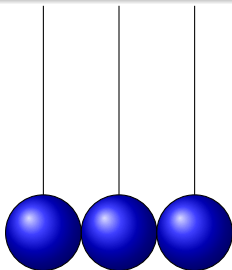
EndOfSnapshot



Goal

Model causality between simultaneous events

Sequencing of instantaneous actions (\approx synchronous microsteps)

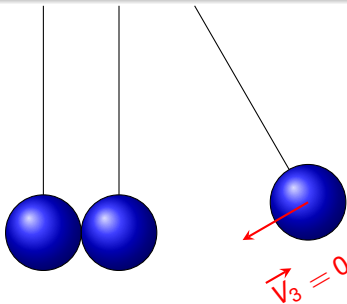


$$t \in \mathbb{R} \times \mathbb{N}$$

Goal

Model causality between simultaneous events

Sequencing of instantaneous actions (\approx synchronous microsteps)

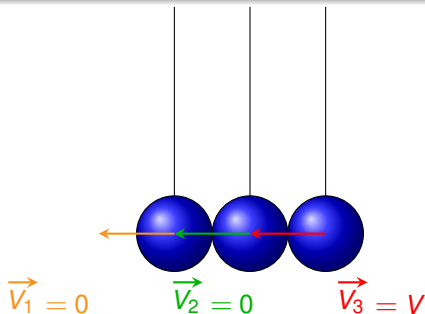


$$t = \langle t_0, 0 \rangle$$

Goal

Model causality between simultaneous events

Sequencing of instantaneous actions (\approx synchronous microsteps)

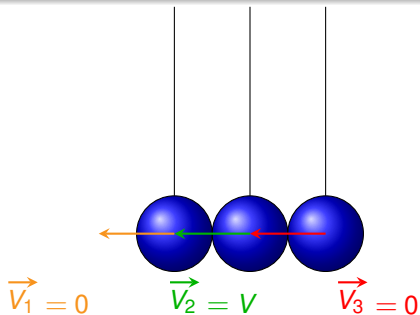


$$t = \langle t_1, 0 \rangle$$

Goal

Model causality between simultaneous events

Sequencing of instantaneous actions (\approx synchronous microsteps)

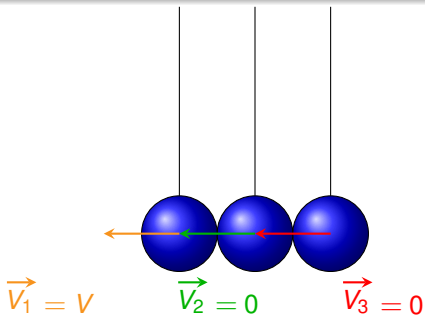


$$t = \langle t_1, 1 \rangle$$

Goal

Model causality between simultaneous events

Sequencing of instantaneous actions (\approx synchronous microsteps)

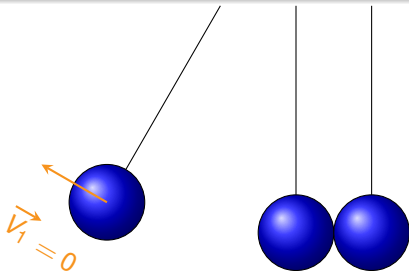


$$t = \langle t_1, 2 \rangle$$

Goal

Model causality between simultaneous events

Sequencing of instantaneous actions (\approx synchronous microsteps)



$$t = \langle t_2, 0 \rangle$$