The Tagged Events Specification Language

Reconciling Heterogeneous Execution Traces

Frédéric Boulanger

Christophe Jacquet Cécile Hardebolle Iuliana Prodan Hai Nguyen Van

CentraleSupélec – LMF

About me

Frédéric Boulanger frederic.boulanger@centralesupelec.fr

Professor at CentraleSupélec since 1994

Researcher at Laboratoire Méthodes Formelles (LMF), created in 2021

Former head of the Department of Computer Science

In charge of the Software Sciences 3rd year concentration:

- 20 to 30 students each year, majoring in computer science
- focus on theoretical fundations, languages, semantics, proofs
- modeling, specification, verification, MDE

Output to all sectors: counselling, finance, research, development, R&D

Agenda

1. Context: execution of heterogeneous models
2. TESL
3. Solving TESL specifications
4. Running simulations
5. Semantic Framework for Timed Coordination Languages
6. Conclusion

Context: Heterogeneous Models

Context: Heterogeneous Models

Software

Sensors

Networks

Control

Electronics

Aerodynamics

Actuators

Mechanics

Context: Heterogeneous Models

Environment

Software

Sensors

Networks

Control

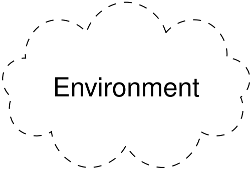
Electronics

Aerodynamics

Actuators

Mechanics

Context: Heterogeneous Models



$$\frac{dS_T}{dt} = r_T - S_T\left(\frac{\beta_{I_B S_T} I_B + \beta_{I_V S_T} I_V}{N_B + N_V}\right) - d_M S_T$$

$$\frac{dI_T}{dt} = S_T\left(\frac{\beta_{I_B S_T} I_B + \beta_{I_V S_T} I_V}{N_B + N_V}\right) - d_I I_T$$

$$\frac{dS_V}{dt} = r_V - S_V\left(\frac{\beta_{I_T S_V} I_T + \beta_{I_B S_V} I_B + \beta_{I_V S_V} I_V}{N_T + N_V + N_B}\right) - d_M S_V$$

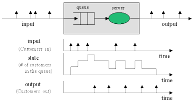
$$\frac{dI_V}{dt} = S_V\left(\frac{\beta_{I_T S_V} I_T + \beta_{I_B S_V} I_B + \beta_{I_V S_V} I_V}{N_T + N_V + N_B}\right) - d_I I_V$$

$$\frac{dS_B}{dt} = r_B - S_B\left(\frac{\beta_{I_T S_B} I_T + \beta_{I_V S_B} I_V}{N_T + N_V}\right) - d_M S_B$$

$$\frac{dI_B}{dt} = S_B\left(\frac{\beta_{I_T S_B} I_T + \beta_{I_V S_B} I_V}{N_T + N_V}\right) - d_I I_B,$$

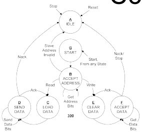
Software

Sensors



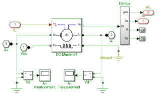
Networks

Control



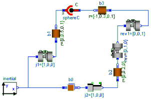
Electronics

Aerodynamics



Actuators

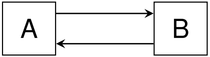
Mechanics



Execution of Heterogeneous Models

Execution of homogeneous parts

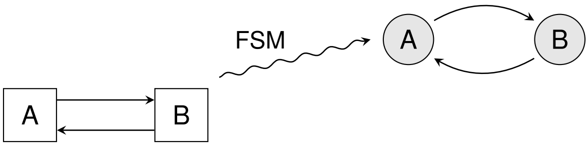
- Interpret the structure according to a paradigm
- Each paradigm brings notions of:
  - Data (events, samples, symbols, functions of continuous time)
  - Time (logical, chronometric, with or without durations)
  - Control (triggering of behaviors, availability of data, concurrency)



Execution of Heterogeneous Models

Execution of homogeneous parts

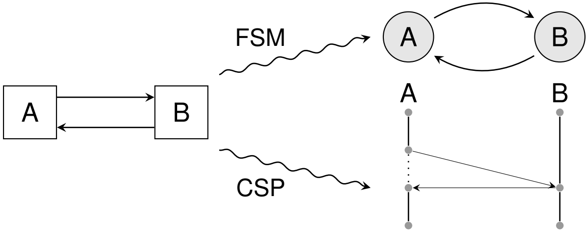
- Interpret the structure according to a paradigm
- Each paradigm brings notions of:
  - Data (events, samples, symbols, functions of continuous time)
  - Time (logical, chronometric, with or without durations)
  - Control (triggering of behaviors, availability of data, concurrency)



Execution of Heterogeneous Models

Execution of homogeneous parts

- Interpret the structure according to a paradigm
- Each paradigm brings notions of:
  - Data (events, samples, symbols, functions of continuous time)
  - Time (logical, chronometric, with or without durations)
  - Control (triggering of behaviors, availability of data, concurrency)



Execution of Heterogeneous Models

Execution of homogeneous parts

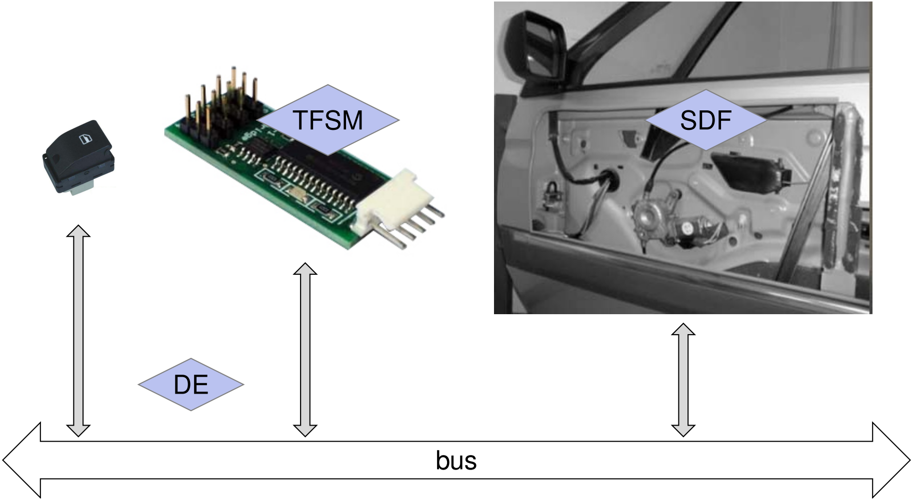
- Interpret the structure according to a paradigm
- Each paradigm brings notions of:
  - Data (events, samples, symbols, functions of continuous time)
  - Time (logical, chronometric, with or without durations)
  - Control (triggering of behaviors, availability of data, concurrency)

Reconciliation of heterogeneous execution traces

- Transform data at the boundaries
- Synchronize different time scales
- Compute control

Example: power window

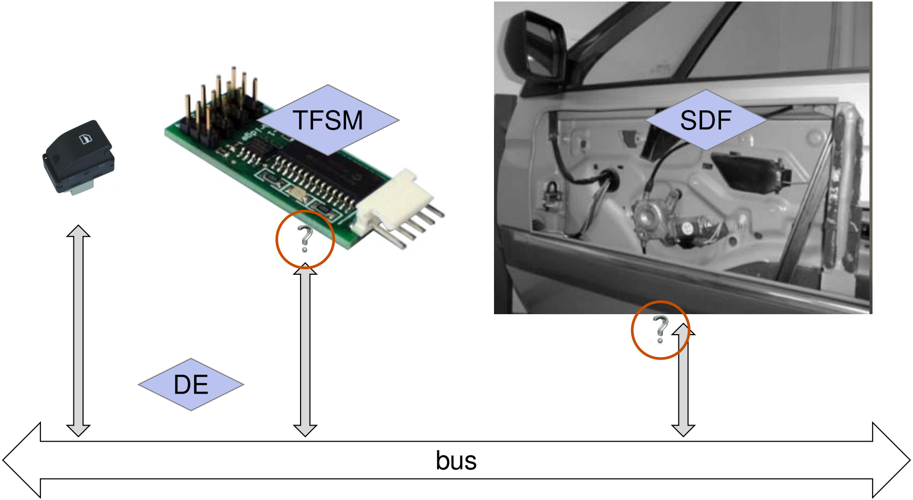
demo



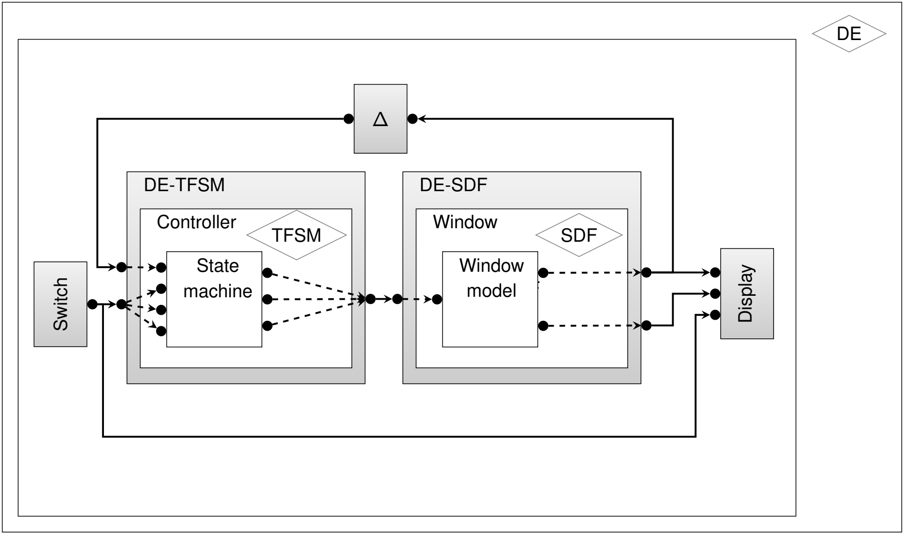
Example

Example: power window

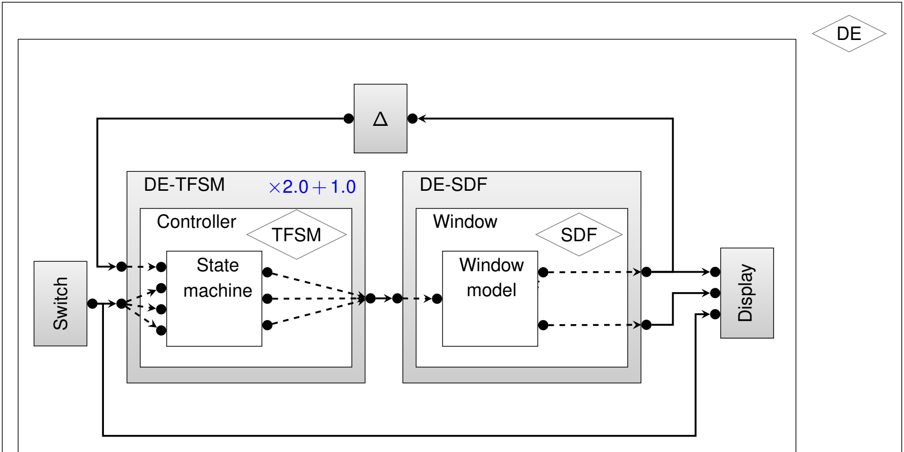
demo



Time and Control in the Power Window

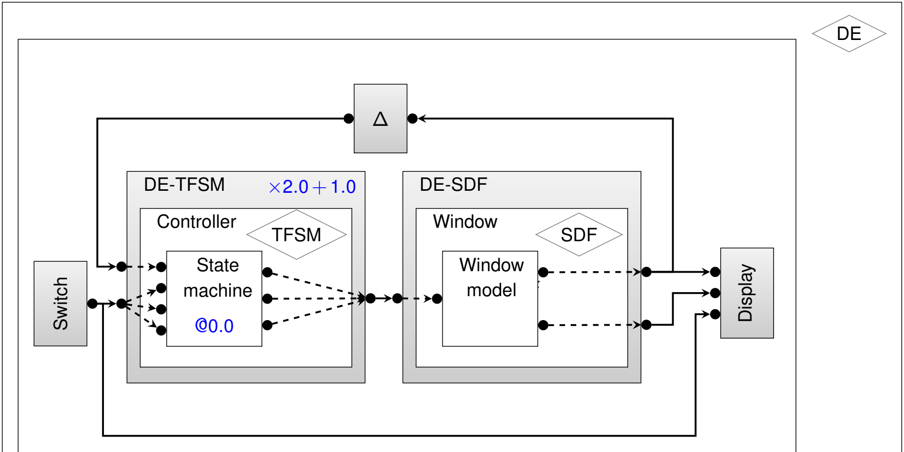


Time and Control in the Power Window



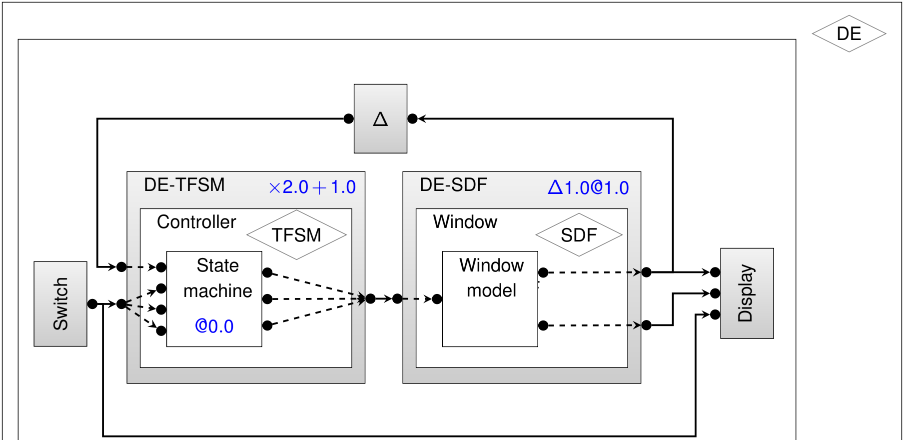
Time runs twice as fast in state machine than in DE and is offset by 1

Time and Control in the Power Window



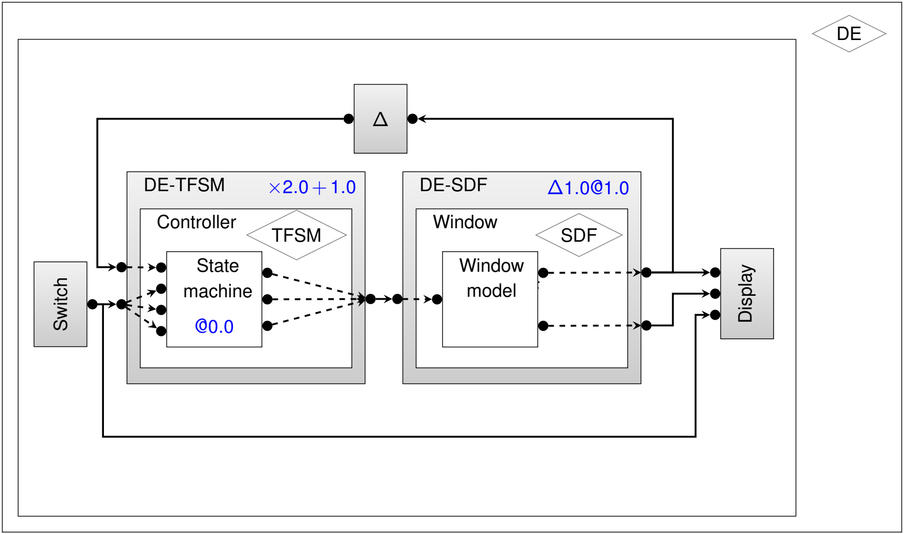
The state machine must receive control at time 0 on the TFSM time scale

Time and Control in the Power Window

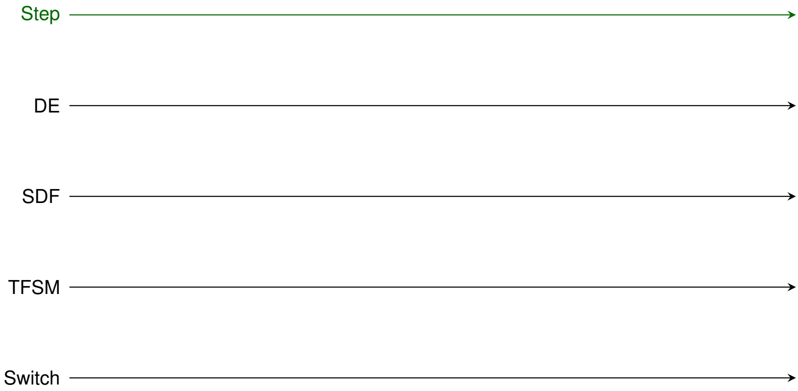


The window model must receive control with period 1.0
starting at 1.0 on the DE time scale

Time and Control in the Power Window

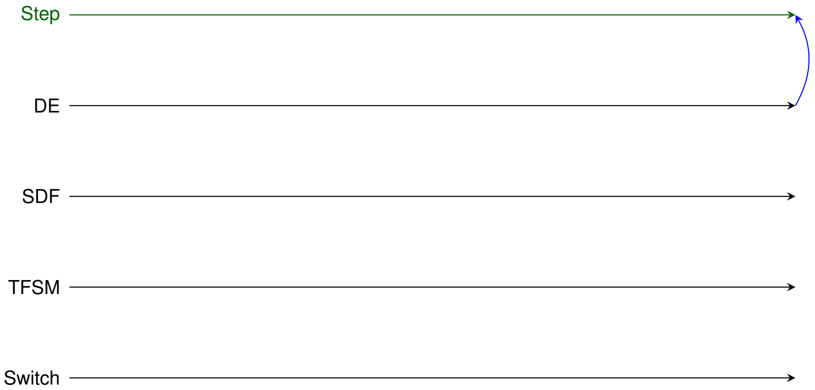


Modeling Time and Control in the Power Window



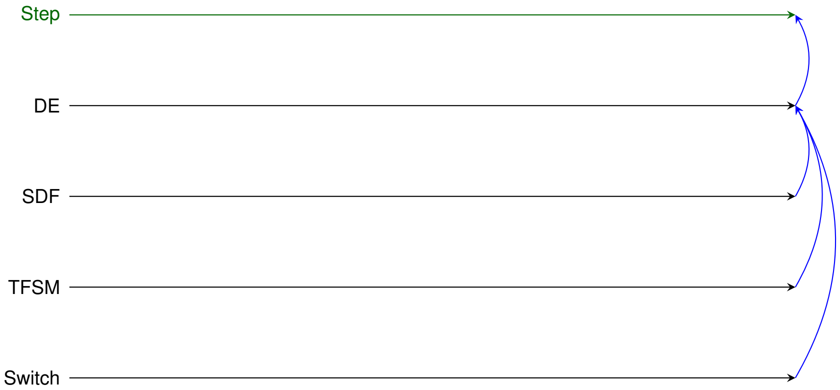
Clocks are used for modeling the control of different parts of the model

Modeling Time and Control in the Power Window



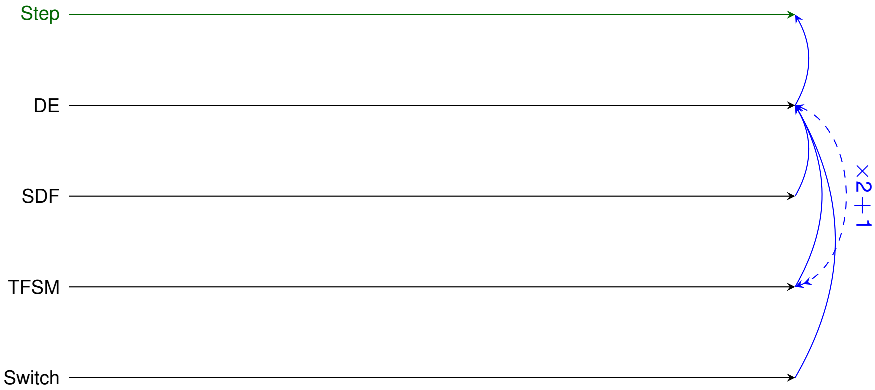
Control for the top level model implies control in the simulation

Modeling Time and Control in the Power Window



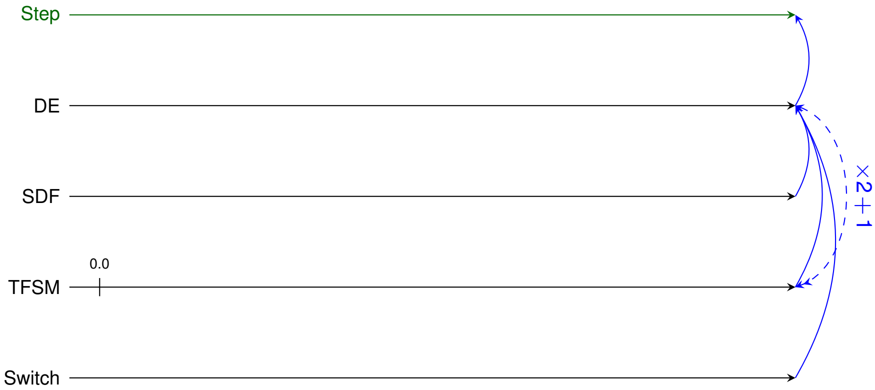
Control for embedded models imply control for the embedding model

Modeling Time and Control in the Power Window



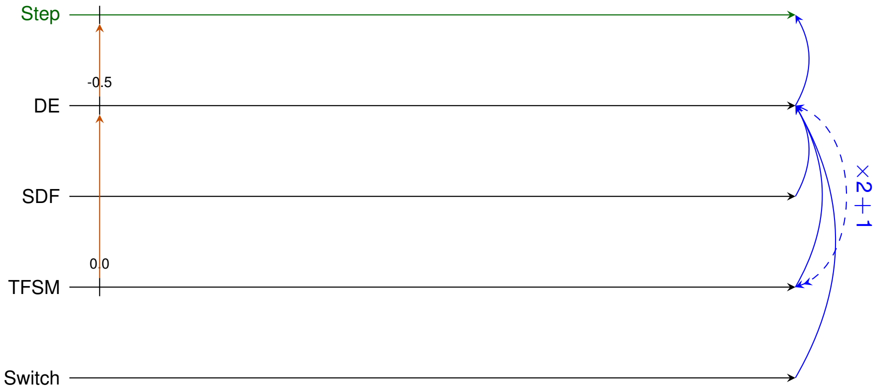
Time in TFSM runs twice as fast as in DE and is offset by one

Modeling Time and Control in the Power Window



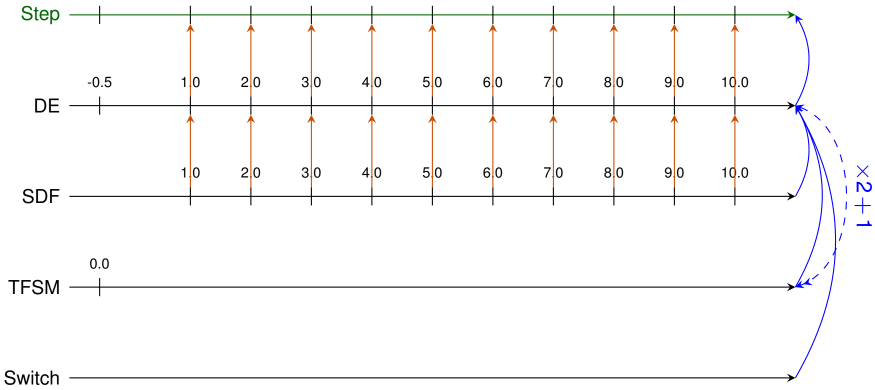
TFSM must receive control at 0.0

Modeling Time and Control in the Power Window



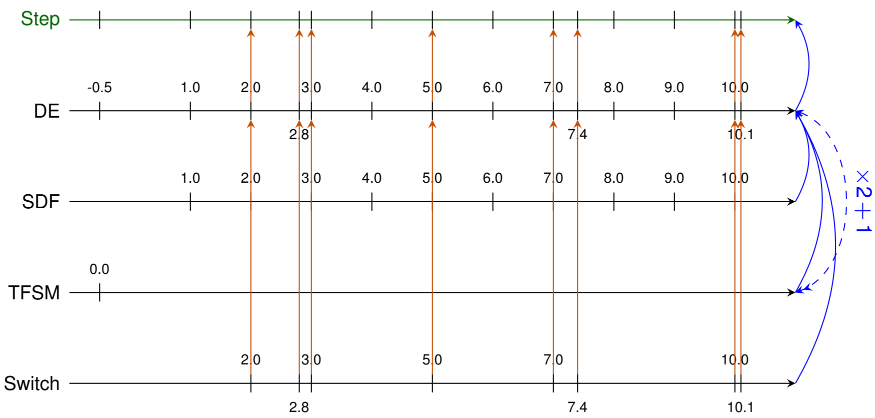
So there must be control in DE at -0.5

Modeling Time and Control in the Power Window



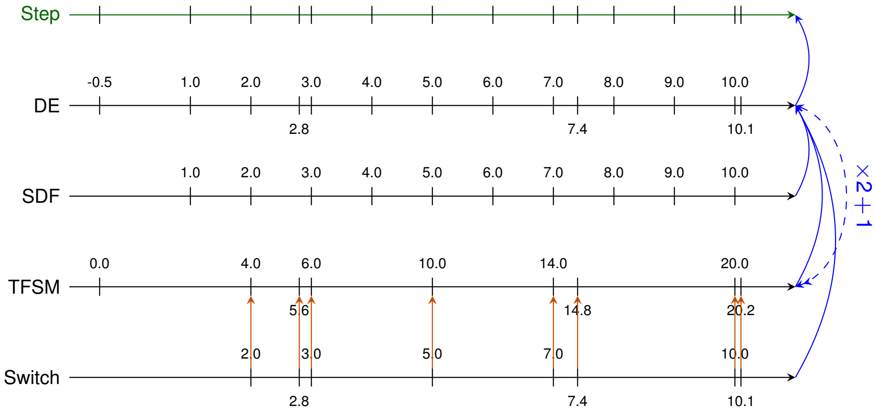
SDF must receive control with period 1.0 on DE time starting at 1.0

Modeling Time and Control in the Power Window



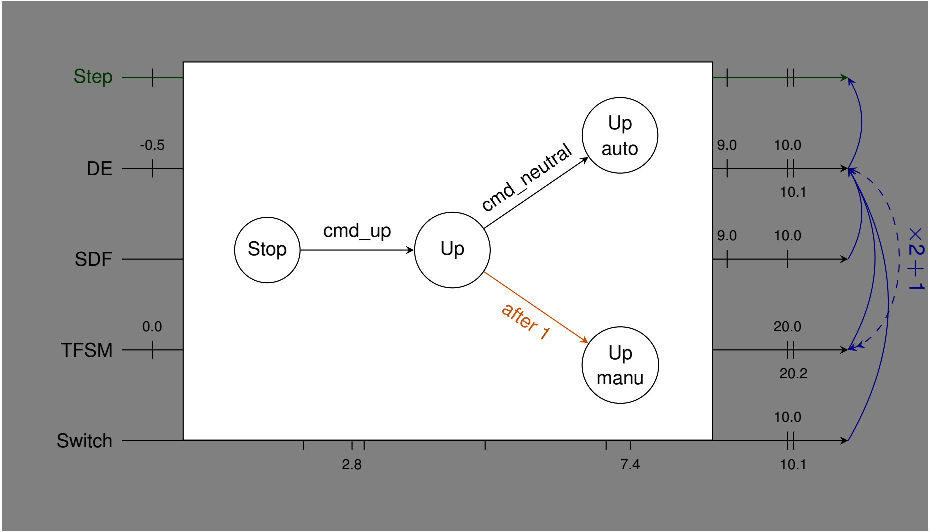
The switch block must receive control to produce data

Modeling Time and Control in the Power Window



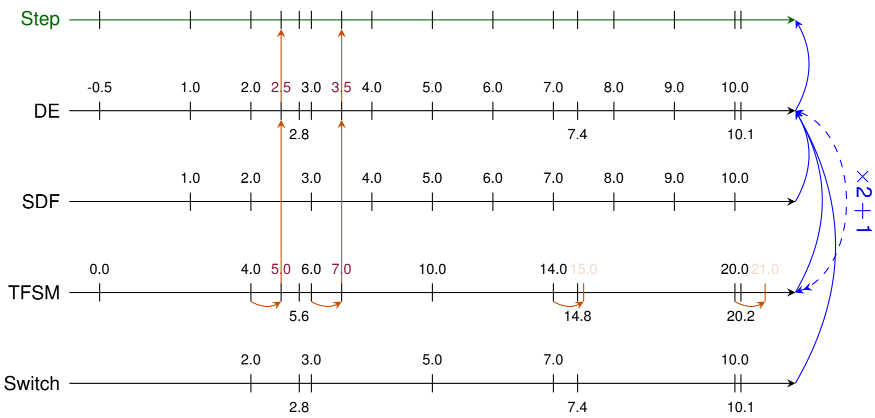
DE semantics creates control for the TFSM when it receives inputs

Modeling Time and Control in the Power Window



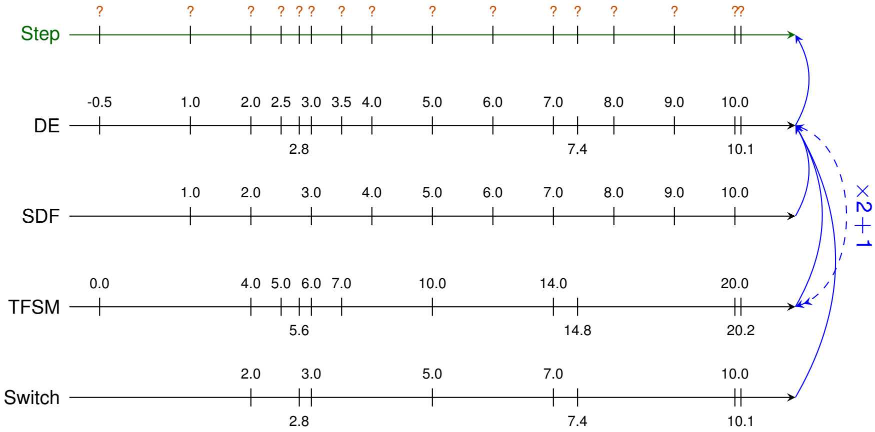
The TFSM controller model has timed transitions

Modeling Time and Control in the Power Window



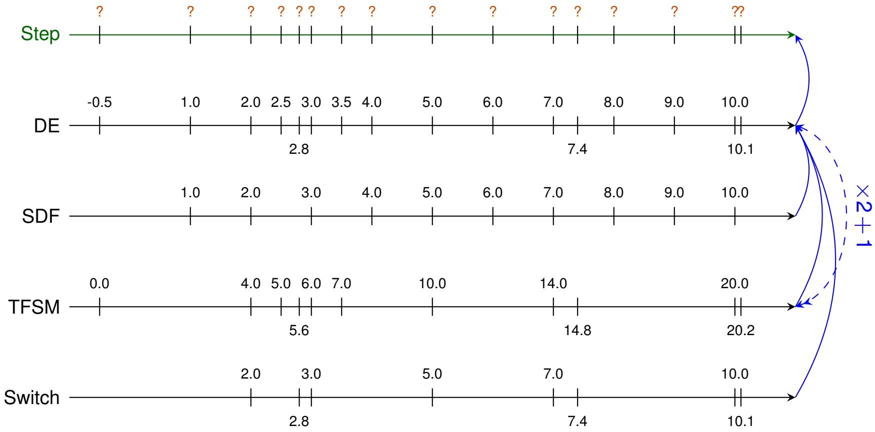
TFSM semantics creates control in TFSM for timed transitions

Modeling Time and Control in the Power Window



When do these events occur?

Modeling Time and Control in the Power Window



When do these events occur? Driving clocks drive the simulation.

Agenda

TESL

Goal

- Model relations between control and time in heterogeneous models
- Allow for deterministic simulations
- Allow the synchronization of the simulation with the environment

Sources

- Synchronous languages (Esterel) for causality and preemption
- Tagged Signal Model and MARTE UML profile for the notions of time
- CCSL for the declarative syntax

Key ideas

- Events are modeled by clocks
- Event occurrences are modeled by ticks
- A tick has a tag which belongs to the domain of its clock
- A behavior is a series of instants which contain simultaneous ticks

TESL static elements

Clocks

- T-clock: clock with time domain T
- Time domain T = ordered set with $+$, $-$, $\times$, $/$ operations, 0 and 1
- Examples of time domains: $\{\star\}, \mathbb{Z}, \mathbb{D}, \mathbb{Q}, \mathbb{R}, \mathbb{R} \times \mathbb{N}$

Implications

- `a implies b` each instant with a tick on *a* also has a tick on *b*
- Conditional implication: implication guarded by a Mealy machine

Time delays

- `a time delayed by d on m implies b` $d > 0$

Tag relations

- $\mathrm{dom}(a) \underset{r}{\overset{d}{\rightleftarrows}} \mathrm{dom}(b)$ link the time scales of *a* and *b*

TESL dynamic elements

Creation of ticks

- Sporadic clock: a clock "preloaded" with ticks at given tags
- Periodic clock: a clock with an initial tick and a time delay on itself

Building behaviors

At any instant I_i

- $$\frac{t \in a, t \in I_i, a \text{ implies } b}{\exists t' \in b, t' \in I_i} \quad \text{(causality)}$$
- $$\frac{t \in a, t \in I_i, t' \in b, t \equiv t'}{t' \in I_i} \quad \text{(synchronization)}$$

With $t \equiv t' \Leftrightarrow \begin{cases} & \hat{d}(t) = t' \\ \text{or} & \\ & t = \hat{r}(t') \end{cases}$

Example: implication

Specification

```
Z-clock a periodic 1
Z-clock s sporadic 2, 6
Z-clock e sporadic 4
unit-clock b

a sustained from s to e implies b
@tagref a  @maxstep 10  @output tikz standalone
```

Example: implication

Specification

```
Z-clock a periodic 1
Z-clock s sporadic 2, 6
Z-clock e sporadic 4
unit-clock b

a sustained from s to e implies b
@tagref a  @maxstep 10  @output tikz standalone
```

Result

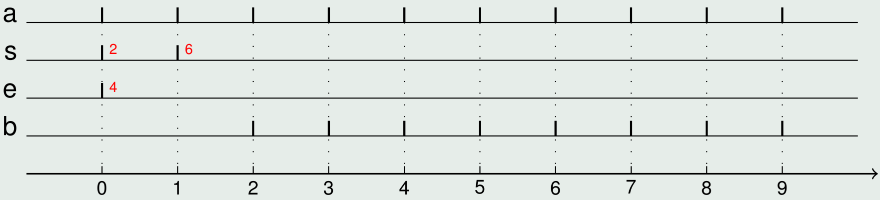
Example: implication

Specification

```
Z-clock a periodic 1
Z-clock s sporadic 2, 6
Z-clock e sporadic 4
unit-clock b

a sustained from s to e implies b
@tagref a  @maxstep 10  @output tikz standalone
```

Result



Example: implication

Specification

```
Z-clock a periodic 1
Z-clock s sporadic 2, 6       tag relation s = a
Z-clock e sporadic 4          tag relation e = a
unit-clock b

a sustained from s to e implies b
@tagref a  @maxstep 10  @output tikz standalone
```

Example: implication

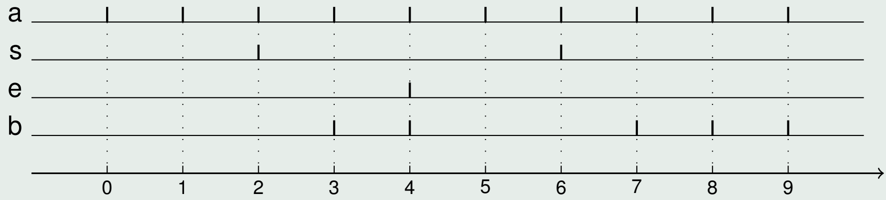
Specification

```
Z-clock a periodic 1
Z-clock s sporadic 2, 6
Z-clock e sporadic 4
unit-clock b

a sustained from s to e implies b
@tagref a  @maxstep 10  @output tikz standalone
```

```
tag relation s = a
tag relation e = a
```

Result



Example: Time delayed implication

Specification

```
Q-clock a sporadic 2, 4
Q-clock m    tag relation m = a
unit-clock b

a time delayed by 2.5 on m implies b
@tagref a  @output tikz standalone
```

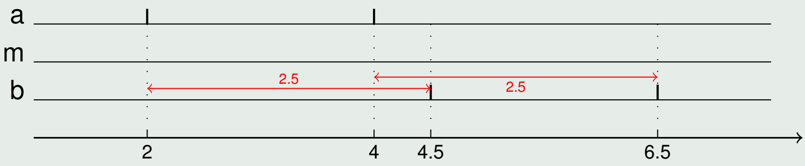
Example: Time delayed implication

Specification

```
Q-clock a sporadic 2, 4
Q-clock m    tag relation m = a
unit-clock b

a time delayed by 2.5 on m implies b
@tagref a  @output tikz standalone
```

Result



Example: Time delayed implication

Specification

```
Q-clock a sporadic 2, 4
Q-clock m    tag relation m = 2*a + 1
unit-clock b

a time delayed by 2.5 on m implies b
@tagref a  @output tikz standalone
```

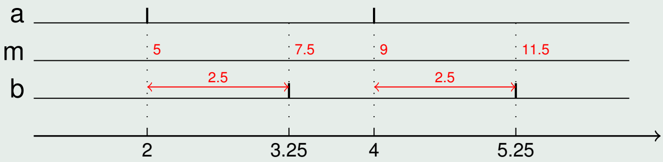
Example: Time delayed implication

Specification

```
Q-clock a sporadic 2, 4
Q-clock m   tag relation m = 2*a + 1
unit-clock b

a time delayed by 2.5 on m implies b
@tagref a  @output tikz standalone
```

Result

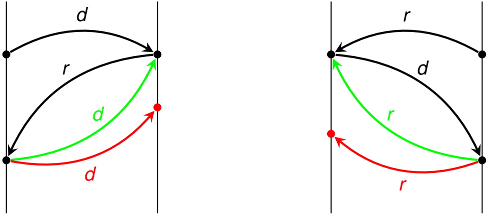


More about tag relations

Tag relations are pairs of non-decreasing functions (d, r) with:

- $d \circ r \circ d = d$
- $r \circ d \circ r = r$

But they are not necessarily bijections nor the reverse of each other.



This condition allows a clock to be “finer” than another clock without forcing different tags to be simultaneous on a clock.

More about tag relations

Specification

```
Z-clock a sporadic 2, 4, 5
Z-clock b     tag relation a = 2*b + 0
a implies b

@tagref b  @output tikz standalone
```

$$\texttt{a = 2*b + 0} \Rightarrow \begin{cases} d : t \mapsto 2t \\ r : t \mapsto t \div 2 \end{cases}$$

More about tag relations

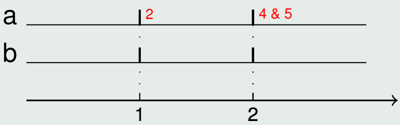
Specification

```
Z-clock a sporadic 2, 4, 5
Z-clock b     tag relation a = 2*b + 0
a implies b

@tagref b  @output tikz standalone
```

$$\texttt{a = 2*b + 0} \Rightarrow \begin{cases} d : t \mapsto 2t \\ r : t \mapsto t \div 2 \end{cases}$$

Result



More about tag relations

Specification

```
Z-clock a sporadic 2, 4, 5
Z-clock b    tag relation a = 2*b + 0
a implies b

@tagref a  @output tikz standalone
```

a = 2*b + 0 $\Rightarrow \begin{cases} d : t \mapsto 2t \\ r : t \mapsto t \div 2 \end{cases}$

More about tag relations

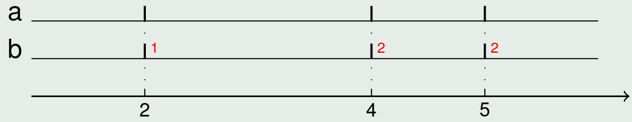
Specification

```
Z-clock a sporadic 2, 4, 5
Z-clock b    tag relation a = 2*b + 0
a implies b

@tagref a  @output tikz standalone
```

`a = 2*b + 0` $\Rightarrow \begin{cases} d : t \mapsto 2t \\ r : t \mapsto t \div 2 \end{cases}$

Result



Agenda

Solving TESL specifications

Goal

- Build a series of instants (I_i)
- Each instant contains simultaneous ticks
  according to the causality and synchronization rules
- All ticks must be assigned to an instant
- For any clock c, $t \in c, t \in I_i, t' \in c, t' \in I_j, j > i \Rightarrow t' \geq t$

Solving TESL specifications

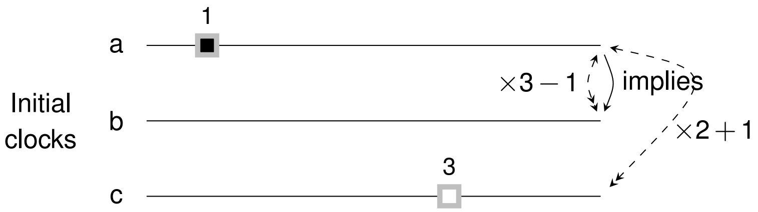
Goal

- Build a series of instants (I_i)
- Each instant contains simultaneous ticks
  according to the causality and synchronization rules
- All ticks must be assigned to an instant
- For any clock c, $t \in c, t \in I_i, t' \in c, t' \in I_j, j > i \Rightarrow t' \geq t$

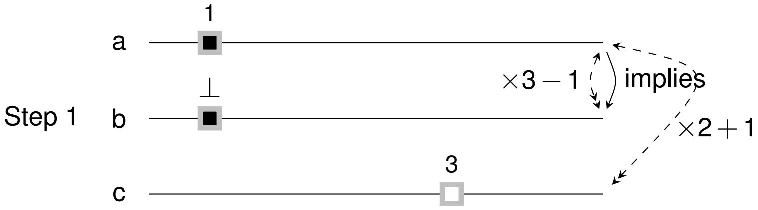
Building an instant

- A tick can be forced into an instant (input)
- A tick on a *greedy* clock is put into an instant as soon as possible
- A tick on a non-greedy clock is put into an instant only when needed
- The causality and synchronization rules are applied
  until a fixed-point is reached (it contains at most one tick per clock)

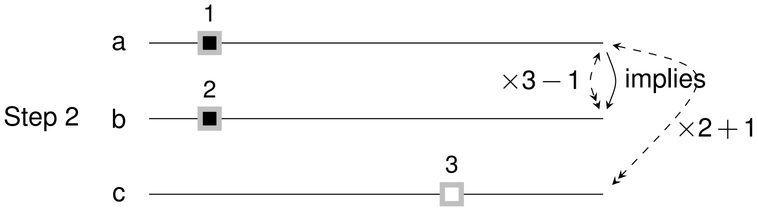
Solving TESL specifications



Solving TESL specifications

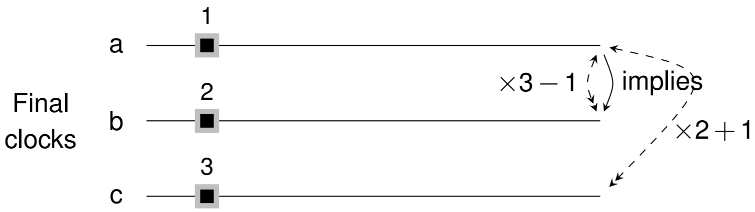


Applying the implication relation



Using the tag relation between a and b

Solving TESL specifications



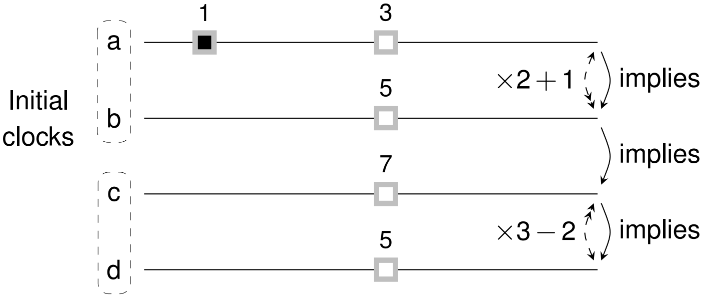
Using the tag relation between b and c

Dealing with time islands

A time island is a connected subgraph of the tag relation graph.

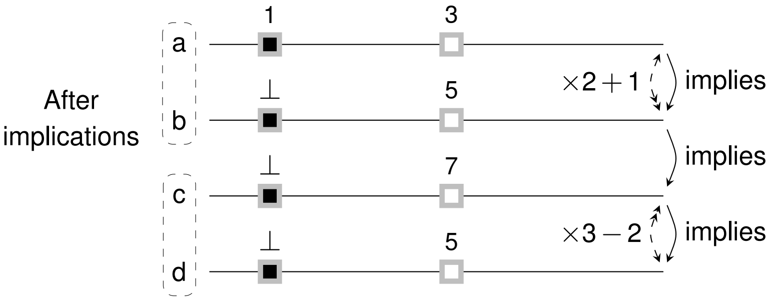
Merging a floating tick with a tagged tick

can be done independently in each time island.



Dealing with time islands

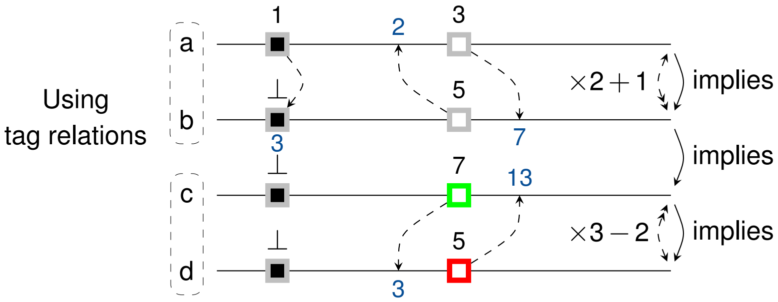
A time island is a connected subgraph of the tag relation graph.
Merging a floating tick with a tagged tick
can be done independently in each time island.



Dealing with time islands

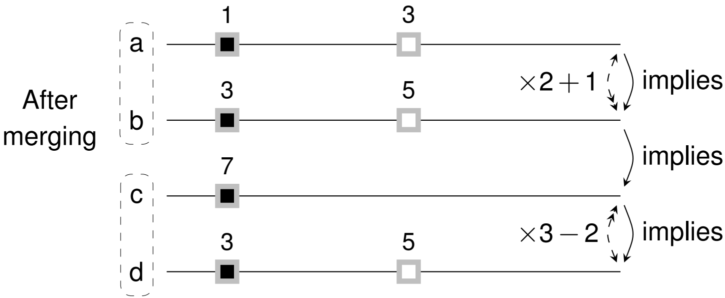
A time island is a connected subgraph of the tag relation graph.

Merging a floating tick with a tagged tick
can be done independently in each time island.



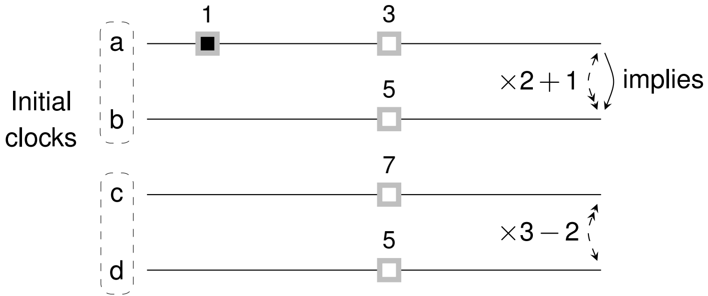
Dealing with time islands

A time island is a connected subgraph of the tag relation graph.
Merging a floating tick with a tagged tick
can be done independently in each time island.



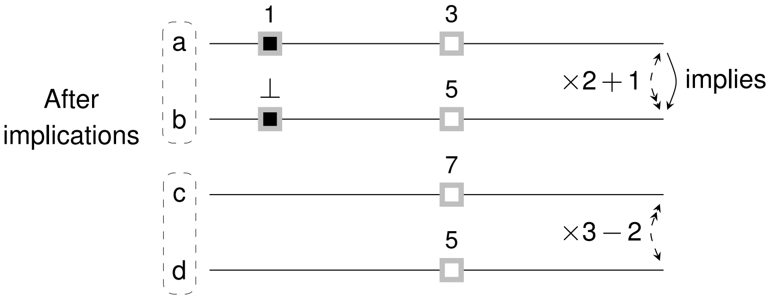
Greedy clocks

Assuming clocks *c* and *d* are greedy,
we should put ticks 7 and 5 in an instant as soon as possible



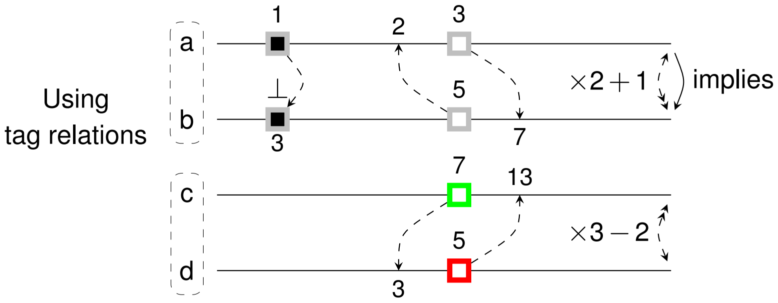
Greedy clocks

Assuming clocks *c* and *d* are greedy,
we should put ticks 7 and 5 in an instant as soon as possible



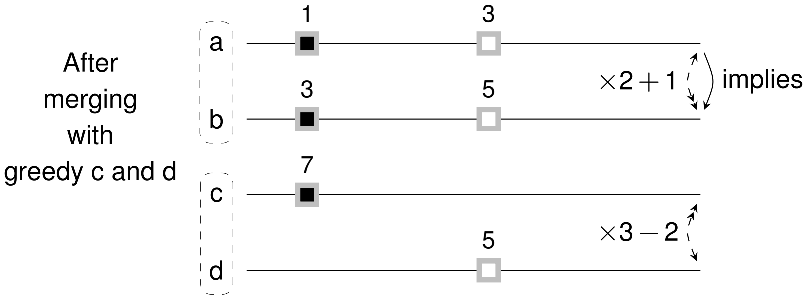
Greedy clocks

Assuming clocks *c* and *d* are greedy,
we should put ticks 7 and 5 in an instant as soon as possible



Greedy clocks

Assuming clocks *c* and *d* are greedy,
we should put ticks 7 and 5 in an instant as soon as possible



Agenda

TESL for running simulations

Model time

- Modeling causality between events in a model
- Modeling time delays in a model
- Modeling relations between time scales in a model

TESL for running simulations

Model time

- Modeling causality between events in a model
- Modeling time delays in a model
- Modeling relations between time scales in a model

“Real” time

- Mapping external events to model events
- Mapping durations from external time to model time

TESL for running simulations

Model time

- Modeling causality between events in a model
- Modeling time delays in a model
- Modeling relations between time scales in a model

“Real” time

- Mapping external events to model events
- Mapping durations from external time to model time

Examples of *driving clocks*

- Event feeder clock (ticks each time an external event occurs)
- Real-time periodic clock (ticks periodically on the system clock)
- Time synchronizing clock (synchronizes its ticks with system time)
- AFAP clock (ticks as fast as possible)

TESL for running simulations

For each simulation step

1. Solve the specification with greedy driving clocks
2. Wait for any of the driving clock with a tick in the resulting instant
3. Solve the specification with non-greedy driving clocks
4. Compute the simulation step using the resulting instant
5. Compute the next specification

TESL for running simulations

For each simulation step

1. Solve the specification with greedy driving clocks
2. Wait for any of the driving clock with a tick in the resulting instant
3. Solve the specification with non-greedy driving clocks
4. Compute the simulation step using the resulting instant
5. Compute the next specification

Advantages

- The simulation engine is free from platform specific code
- The same formalism is used for both model time and real time
- The mapping between model time and real time is explicit
- Model-specific synchronization can be added as new driving clocks

Runs

TESL specifications describe *runs*

A run is a sequence of observations of an enumerable set of clocks $\mathbb{K}$

Each observation is an *instant* at which a clock:

- may tick or not
- has a time stamp

$$\rho : \mathbb{N} \to \mathbb{K} \to (\mathbb{B} \times \mathbb{T})$$

Runs

TESL specifications describe *runs*

A run is a sequence of observations of an enumerable set of clocks $\mathbb{K}$

Each observation is an *instant* at which a clock:

- may tick or not
- has a time stamp

$$\rho : \mathbb{N} \to \mathbb{K} \to (\mathbb{B} \times \mathbb{T})$$

Time cannot flow backwards:

$$i > j \Longrightarrow \mathrm{time}(\rho\ i\ c) \geq \mathrm{time}(\rho\ j\ c)$$

Denotational Semantics

$$[\![C_1 \ \texttt{sporadic}\ \tau\ \texttt{on}\ C_2]\!]_{\text{TESL}}$$

$$\stackrel{\text{def}}{=} \{\rho \mid \exists n \in \mathbb{N}.\ \text{ticks}(\rho\ n\ C_1) \wedge \text{time}(\rho\ n\ C_2) = \tau\}$$

$$[\![C_1 \ \texttt{implies}\ C_2]\!]_{\text{TESL}}$$

$$\stackrel{\text{def}}{=} \{\rho \mid \forall n \in \mathbb{N}.\ \text{ticks}(\rho\ n\ C_1) \Longrightarrow \text{ticks}(\rho\ n\ C_2)\}$$

$$[\![\texttt{time relation}\ (C_1, C_2) \in R]\!]_{\text{TESL}}$$

$$\stackrel{\text{def}}{=} \left\{\rho \;\middle|\; \forall n \in \mathbb{N}.\ \big(\text{time}(\rho\ n\ C_1), \text{time}(\rho\ n\ C_2)\big) \in R\right\}$$

$$[\![C_{\text{master}}\ \texttt{time delayed by}\ \delta\tau\ \texttt{on}\ C_{\text{meas}}\ \texttt{implies}\ C_{\text{slave}}]\!]_{\text{TESL}}$$

$$\stackrel{\text{def}}{=} \{\rho \mid \forall n \in \mathbb{N}.\ \text{ticks}(\rho\ n\ C_{\text{master}}) \Longrightarrow \forall m \geq n.\ \text{elapsed}(\rho, C_{\text{meas}}, n, \delta\tau, m) \Longrightarrow \text{ticks}(\rho\ m\ C_{\text{slave}})\}$$

Denotational Semantics

$$\left[\!\left[C_1 \ \textbf{sporadic}\ \tau \ \textbf{on}\ C_2 \right]\!\right]_{\text{TESL}}$$

$$\stackrel{\text{def}}{=} \{\rho \mid \exists n \in \mathbb{N}.\ \text{ticks}(\rho\ n\ C_1) \wedge \text{time}(\rho\ n\ C_2) = \tau\}$$

$$\left[\!\left[C_1 \ \textbf{implies}\ C_2 \right]\!\right]_{\text{TESL}}$$

$$\stackrel{\text{def}}{=} \{\rho \mid \forall n \in \mathbb{N}.\ \text{ticks}(\rho\ n\ C_1) \Longrightarrow \text{ticks}(\rho\ n\ C_2)\}$$

$$\left[\!\left[\textbf{time relation}\ (C_1, C_2) \in R \right]\!\right]_{\text{TESL}}$$

$$\stackrel{\text{def}}{=} \left\{\rho \;\middle|\; \forall n \in \mathbb{N}.\ \big(\text{time}(\rho\ n\ C_1), \text{time}(\rho\ n\ C_2)\big) \in R\right\}$$

$$\left[\!\left[C_{\text{master}} \ \textbf{time delayed by}\ \delta\tau \ \textbf{on}\ C_{\text{meas}} \ \textbf{implies}\ C_{\text{slave}} \right]\!\right]_{\text{TESL}}$$

$$\stackrel{\text{def}}{=} \{\rho \mid \forall n \in \mathbb{N}.\ \text{ticks}(\rho\ n\ C_{\text{master}}) \Longrightarrow$$

$$\forall m \geq n.\ \text{elapsed}(\rho, C_{\text{meas}}, n, \delta\tau, m) \Longrightarrow \text{ticks}(\rho\ m\ C_{\text{slave}})\}$$

Reasonning on runs, proof of invariance by stuttering.

Operational Semantics

Specification = potential future

Action = make a decision in the present

State = decisions that have already been made

Operational Semantics

Specification = potential future

Action = make a decision in the present

State = decisions that have already been made

Context: $\Gamma \models_n \Psi \triangleright \Phi$

Γ is the past up to instant n

Ψ is the remaining constraint on instant n

Φ is the constraint on the future

Operational Semantics

Specification = potential future

Action = make a decision in the present

State = decisions that have already been made

Context: $\Gamma \models_n \Psi \triangleright \Phi$

Γ is the past up to instant n

Ψ is the remaining constraint on instant n

Φ is the constraint on the future

Primitive Constraints in Γ

$$[\![C \Uparrow_n]\!]_{\text{prim}} \stackrel{\text{def}}{=} \{\rho \mid \text{ticks}(\rho\ n\ C)\}$$

$$[\![C \not\Uparrow_n]\!]_{\text{prim}} \stackrel{\text{def}}{=} \{\rho \mid \neg\text{ticks}(\rho\ n\ C)\}$$

$$[\![C \Downarrow_n x]\!]_{\text{prim}} \stackrel{\text{def}}{=} \{\rho \mid \text{time}(\rho\ n\ C) = x\}$$

$$[\![(\text{tval}_{n_1}^{C_1}, \text{tval}_{n_2}^{C_2}) \in R]\!]_{\text{prim}} \stackrel{\text{def}}{=} \{\rho \mid (\text{time}(\rho\ n_1\ C_1), \text{time}(\rho\ n_2\ C_2)) \in R\}$$

Operational Rules

Introduction rule

$$\Gamma \models_n \varnothing \triangleright \Phi \quad \rightarrow_i \quad \Gamma \models_{n+1} \Phi \triangleright \varnothing$$

Operational Rules

Introduction rule

$$\Gamma \models_n \varnothing \triangleright \Phi \quad \rightarrow_i \quad \Gamma \models_{n+1} \Phi \triangleright \varnothing$$

Elimination rules

$$\Gamma \models_n \Psi \wedge (C_1 \ \mathbf{sporadic}\ \tau \ \mathbf{on}\ C_2) \triangleright \Phi \qquad (\text{sporadic} - \text{on}_{e1})$$
$$\rightarrow_e \Gamma \models_n \Psi \triangleright \Phi \wedge (C_1 \ \mathbf{sporadic}\ \tau \ \mathbf{on}\ C_2)$$

$$\Gamma \models_n \Psi \wedge (C_1 \ \mathbf{sporadic}\ \tau \ \mathbf{on}\ C_2) \triangleright \Phi \qquad (\text{sporadic} - \text{on}_{e2})$$
$$\rightarrow_e \Gamma \cup \left\{ C_1 \Uparrow_n, C_2 \Downarrow_n \tau \right\} \models_n \Psi \triangleright \Phi$$

$$\Gamma \models_n \Psi \wedge (C_1 \ \mathbf{implies}\ C_2) \triangleright \Phi \qquad (\text{implies}_{e1})$$
$$\rightarrow_e \Gamma \cup \left\{ C_1 \not\Uparrow_n \right\} \models_n \Psi \triangleright \Phi \wedge (C_1 \ \mathbf{implies}\ C_2)$$

$$\Gamma \models_n \Psi \wedge (C_1 \ \mathbf{implies}\ C_2) \triangleright \Phi \qquad (\text{implies}_{e2})$$
$$\rightarrow_e \Gamma \cup \left\{ C_1 \Uparrow_n, C_2 \Uparrow_n \right\} \models_n \Psi \triangleright \Phi \wedge (C_1 \ \mathbf{implies}\ C_2)$$

Operational Semantics

Properties

- Local termination (Ψ eventually becomes empty)
- Progress (We can reach any instant n in a run)
- Soundness and completeness with respect to the denotational semantics

$$\llbracket \varnothing \models_0 \Psi \triangleright \varnothing \rrbracket_{\text{config}} = \llbracket \Psi \rrbracket_{\text{TESL}}$$

Operational Semantics

Properties

- Local termination (Ψ eventually becomes empty)
- Progress (We can reach any instant *n* in a run)
- Soundness and completeness with respect to the denotational semantics

$$\llbracket \varnothing \models_0 \Psi \triangleright \varnothing \rrbracket_{\text{config}} = \llbracket \Psi \rrbracket_{\text{TESL}}$$

Uses

- Monitoring of heterogeneous systems
- Online testing

More efficient implementation in SML: Heron

Agenda

Conclusion

TESL

- Synchronous language with tags and durations
- Deterministic, with constructive semantics

Conclusion

TESL

- Synchronous language with tags and durations
- Deterministic, with constructive semantics

A Timed Coordination Language

- Usable for simulation (Eclipse plug-in + ModHel'X)
- With a well defined semantics (Isabelle/HOL theory)
- Usable for monitoring and testing (Heron implementation)

Conclusion

TESL

- Synchronous language with tags and durations
- Deterministic, with constructive semantics

A Timed Coordination Language

- Usable for simulation (Eclipse plug-in + ModHel'X)
- With a well defined semantics (Isabelle/HOL theory)
- Usable for monitoring and testing (Heron implementation)

Execution of Heterogeneous Models

- Modeling causality for data and control, modeling time delays
- Modeling synchronization between heterogeneous time scales

Conclusion

TESL

- Synchronous language with tags and durations
- Deterministic, with constructive semantics

A Timed Coordination Language

- Usable for simulation (Eclipse plug-in + ModHel'X)
- With a well defined semantics (Isabelle/HOL theory)
- Usable for monitoring and testing (Heron implementation)

Execution of Heterogeneous Models

- Modeling causality for data and control, modeling time delays
- Modeling synchronization between heterogeneous time scales

A Framework for Coordination Languages

- Past/Present/Future pattern
- Infrastructure for proving soundness and completeness

Thank You