Hybrid System Development in Event-B

Zheng Cheng Dominique Méry
LORIA & Université de Lorraine

Meeting IFIP WG 1.3 at Lipari
September 5-September 9

General Summary

Current Summary

What Are Hybrid Systems

General Ideas

- Examples: bouncing ball, thermostat, inverted pendulum ...
- A hybrid system is a dynamical system that exhibits both continuous and discrete dynamic behavior
- Hybrid system = continuous dynamics + discrete jump
- Model-based hybrid system design

What Are Hybrid Systems

General Ideas

- Examples: bouncing ball, thermostat, inverted pendulum ...
- A hybrid system is a dynamical system that exhibits both continuous and discrete dynamic behavior
- Hybrid system = continuous dynamics + discrete jump
- Model-based hybrid system design

Hybrid Modeling

- discrete variables ($x \in \mathbb{Z}$)
- continuous variables ($y \in \mathbb{R}^+ \to D$)
- Hybrid Modelling = continuous events + discrete events

Current Summary

Short Summary on Event-B

- Context: static properties of Event-B models
  - Sets: user-defined types
  - Constants: static object in development
  - Axioms: presumed properties about sets and constants
  - Theorems: derived properties about sets and constants

SETS
A
CONSTANTS
B, C, f
AXIOMS
$ax1 : B <: A$
$ax2 : C <: A$
$ax3 : g \in B \pfun C$
$\ldots$

Short Summary on Event-B

- Machine: behavioral properties of Event-B models
  - Variables: states
  - Invariants: properties of variables that always need to hold
  - Theorems: derived properties about variables
  - Events: possible state changes

EVENT e
ANY
  p
WHER
CONSTANTS
  B, C, f
AXIOMSS
  $ax1 : B <: A$
  $ax2 : C <: A$
  $ax3 : g \in B \nrightarrow C$
. . .

General form of an event

```
EVENT e
  ANY  t
  WHERE
    G(c, s, t, x)
  THEN
    x :|(P(c, s, t, x, x'))
  END
```

- c et s are constantes and visible sets by e
- x is a state variable or a list of variabless
- $G(c, s, t, x)$ is the condition for observing e.
- $P(c, s, t, x, x')$ is the assertion for the relation over x and x'.
- $BA(e)(c, s, x, x')$ is the *before-after* relationship for e and is defined by $\exists t.G(c, s, t, x) \ \wedge \ P(c, s, t, x, x')$.

Short Summary on Event-B

- Proof obligations: must be proved to show that Event-B models fulfill their specified properties.
  - INV: invariant preservation
  - FIS: action feasibility
  - ...

General form of proof obligations for an event e

Proofs obligations are simplified when they are generated by the module called POG and goals in sequents as $\Gamma \vdash G$:

1. $\Gamma \vdash G_1 \wedge G_2$ is decomposed into the two sequents $\begin{array}{l} (1)\Gamma \vdash G_1 \\ (2)\Gamma \vdash G_2 \end{array}$
2. $\Gamma \vdash G_1 \Rightarrow G_2$ is transformed into the sequent $\Gamma, G_1 \vdash G_2$

Proof obligations in Rodin

- *INIT/I/INV*: $C(s,c), INIT(c,s,x) \vdash I(c,s,x)$
- *e/I/INV*: $C(s,c), I(c,s,x), G(c,s,t,x), P(c,s,t,x,x') \vdash I(c,s,x')$
- *e/act/FIS*: $C(s,c), I(c,s,x), G(c,s,t,x) \vdash \exists x'.P(c,s,t,x,x')$

Short Summary on Event-B

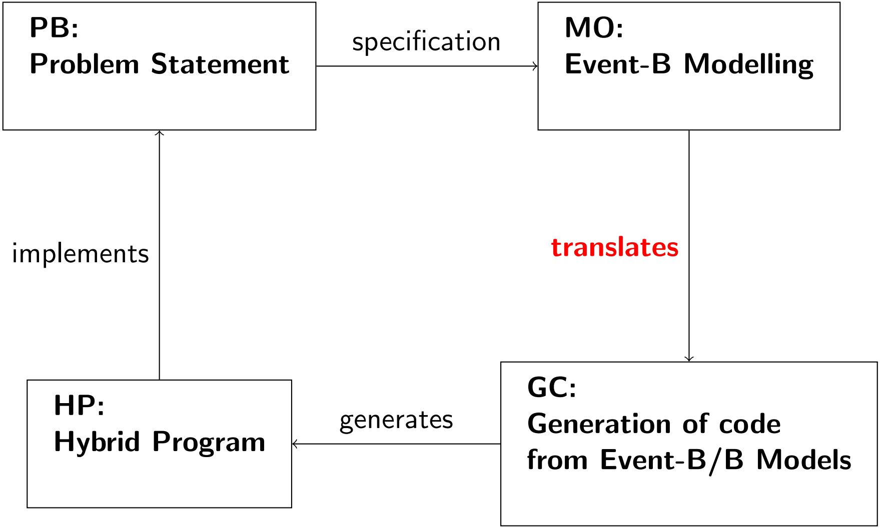
- Theory plugin: more modularize and reusable polymorphic "Context"
- Developed at University of Southampton
- Installation: `http://rodin-b-sharp.sourceforge.net/updates`
  - Modelling Extensions → Theory Feature

Extension of theories

The Event-B modelling language can be extended for handling entities as *differential equations*, *continuity*, ...

Current Summary

General Description of the Methodology



Problem Statement

- formalize the system to-be-developed using the **continuous action system**.
- precisely express the problem context.

Event-B Modelling

- model the formalized problem context in **Event-B**.
- use refinement methodology to design correct hybrid system by construction.
- focus on high-level system modeling.

Generation of code

- develop implementations in **Atelier-B**.
- certified translation from Event-B to Atelier-B.
- focus on low-level software development.

Hybrid Program

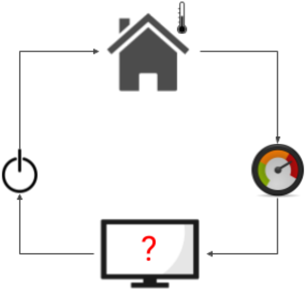
- validates the implementation against certain industry code standards by cross-validation.

Points in this talk

- Focus on Event-B modeling in this talk
- Illustrate our modeling on a smart heating system example
- Illustrate modularization of software-based component (Embedding Event-B Events as B Operation)

Current Summary

Smart Heating System

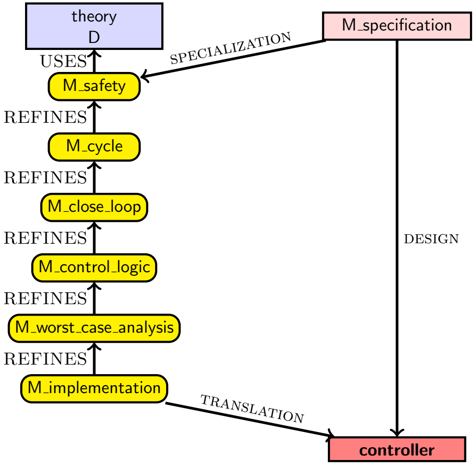


- 2 modes: ON/OFF
- Simple dynamics: $\dot{T}$=1/-1
- Sample at δ s
- Switch mode costs t_{act} s ($t_{act} < \delta$)
- Safety: $T_{min} \leq$ T $\leq T_{max}$

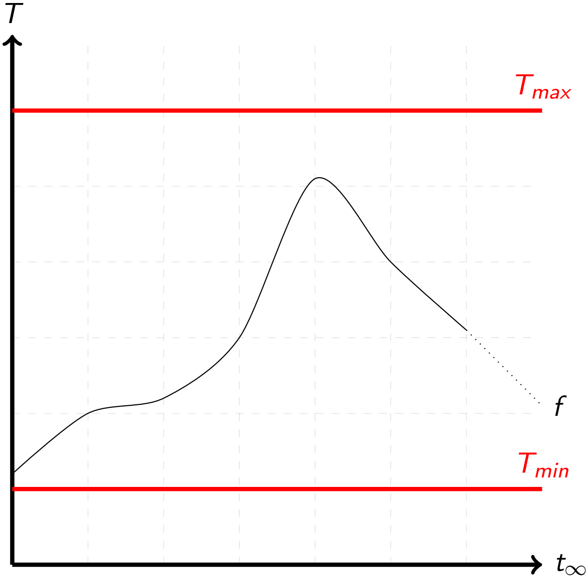
Our Goals

- Design systems in a logical framework, and reason their safety in a machine-checkable way.
- Taking implementation constraints into problem abstraction to reduce the implementation efforts.

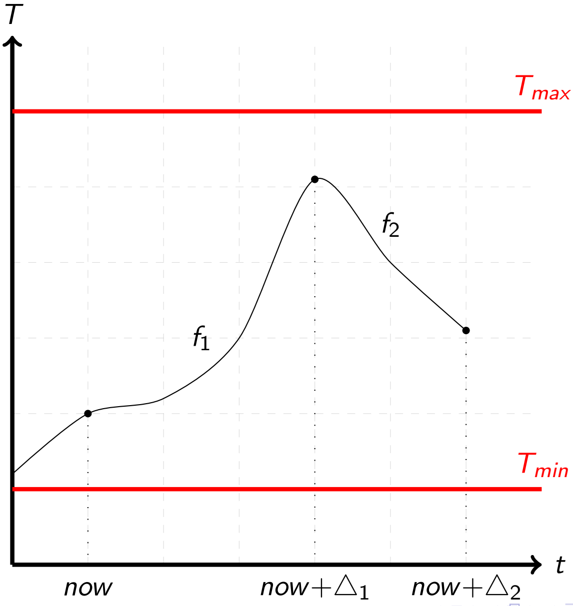
Refinement Strategy for Hybrid System Design



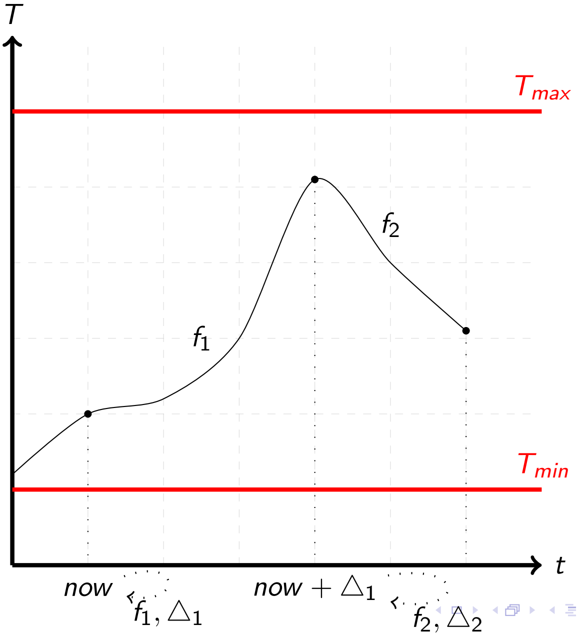
Smart Heating System (M_safety)



Smart Heating System (M_cycle)

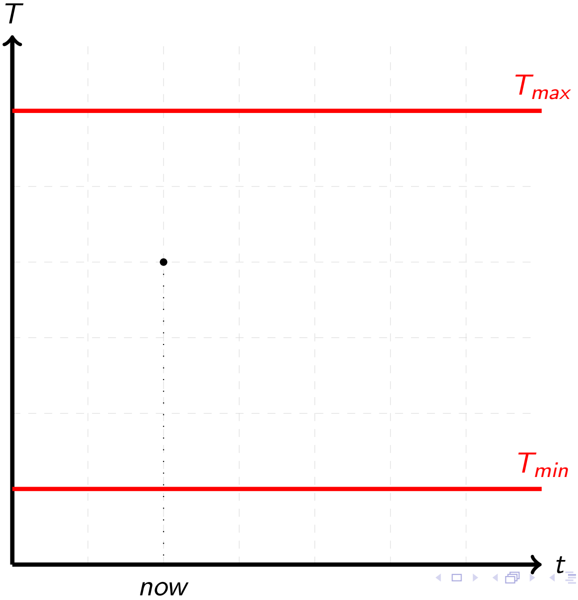


Smart Heating System (M_close_loop)



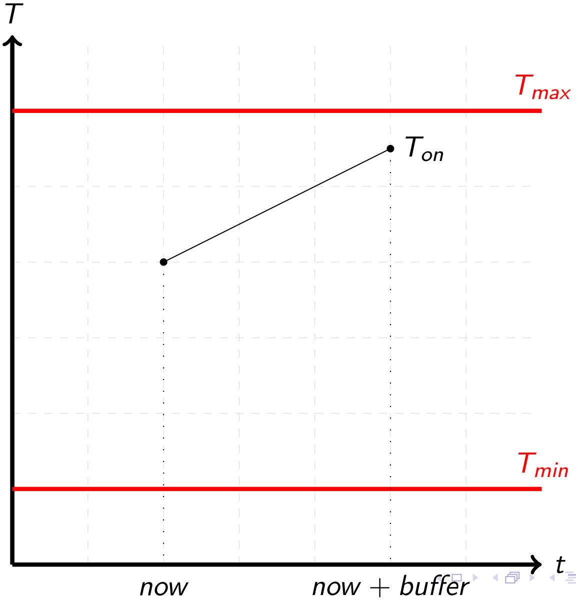
Smart Heating System (M_control_logic)

Case 1 (Bad): ON mode, $T(now) \leq T_{max}$, Stay ON

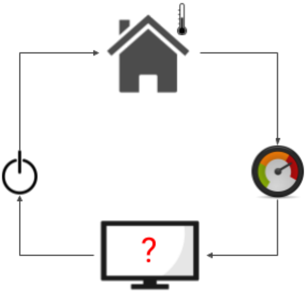


Smart Heating System (M_control_logic)

Case 1 (Good): ON mode, $T(now + buffer) \leq T_{max}$, Stay ON



Smart Heating System (Revisit)



- 2 modes: ON/OFF

$\rightarrow$ the only actuation we can do

- Simple dynamics: $\dot{T}$=1/-1

$\rightarrow$ monotonicity

- Sample at δ s

$\rightarrow$ Decision at sampling time

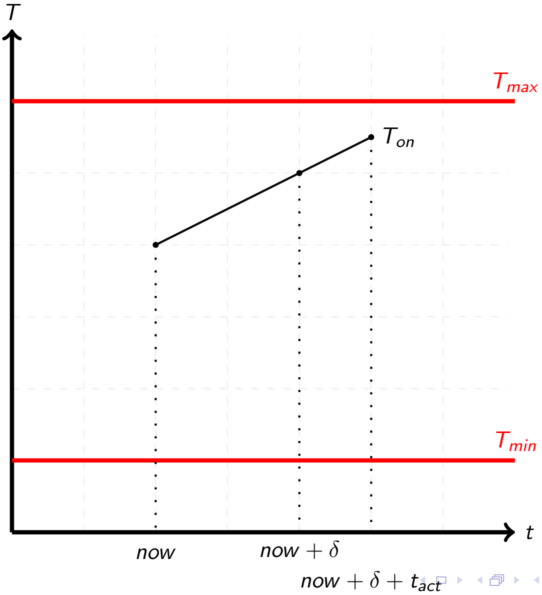
- Switch mode costs t_{act} s ($t_{act} < \delta$)

$\rightarrow$ Cost of switch mode

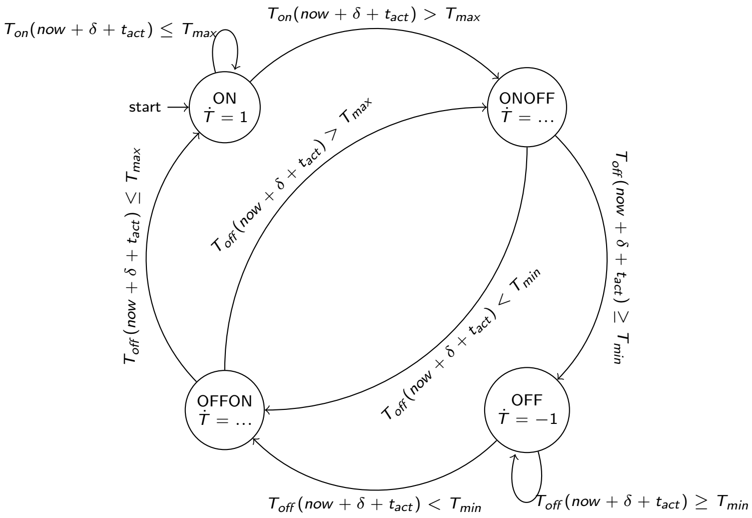
- Safety: $T_{min} \leq$ T $\leq T_{max}$

Smart Heating System (M_worst_case_analysis)

Case 1: ON mode, $T(now + \delta + t_{act}) \leq T_{max}$, Stay ON



Smart Heating System (M_implementation)



Smart Heating System (M_implementation)

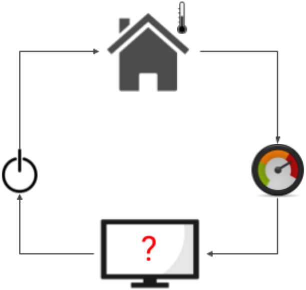
```
if q = ON ∨ q = OFFON then
    if T_on(now + δ + t_act) ≤ T_max then
        q ← ON
    else
        q ← ONOFF
    end if
else if q = OFF ∨ q = ONOFF then
    if T_off(now + δ + t_act) ≥ T_min then
        m ← OFF
    else
        m ← OFFON
    end if
end if
```

Overview of proof efforts

	Total	Auto.	Man.
M_specification	8	7	1
M_safety	14	11	3
M_cycle	16	9	7
M_close_loop	23	18	5
M_control_logic	42	27	15
M_worst_case_analysis	231	149	82
M_implementation	134	99	35
Total	468	320 (68%)	148 (32%)

Current Summary

Smart Heating System



- 2 modes: ON/OFF
- Dynamics to be developed at low-level modeling: $\dot{T}$
- Sample at δ seconds
- Safety: $T_{min} \leq \mathrm{T} \leq T_{max}$

. . . refining a little bit mpre!

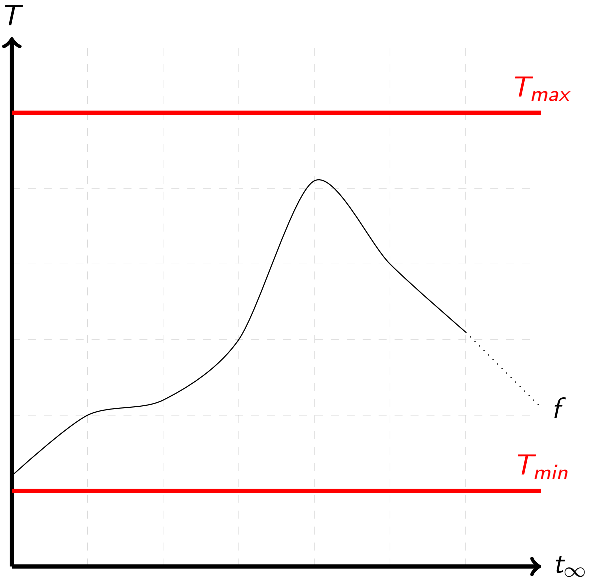
Adding Sensing and Actuating events.

Smart Heating System (Specification M0)

Checklist:

- Generic hybrid system state trajectory
- Generic safety property
- Big-step semantics

Smart Heating System (Safety M1)

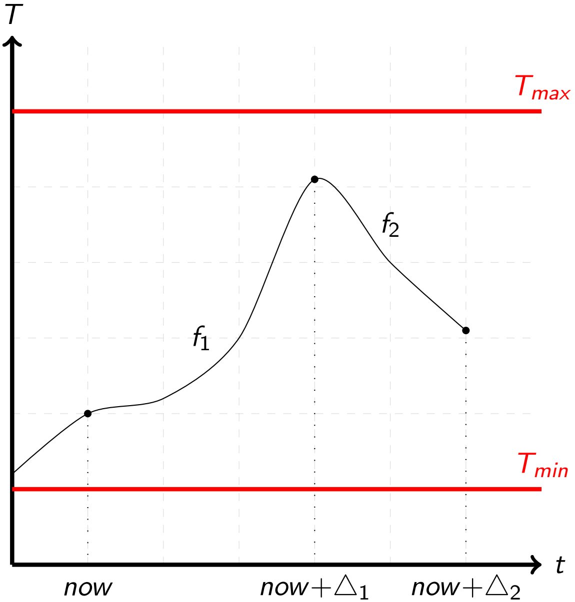


Smart Heating System (Safety M1)

Checklist:

- Concrete system state trajectory
- Concrete safety property
- Big-step semantics refined

Smart Heating System (Cycle M2)

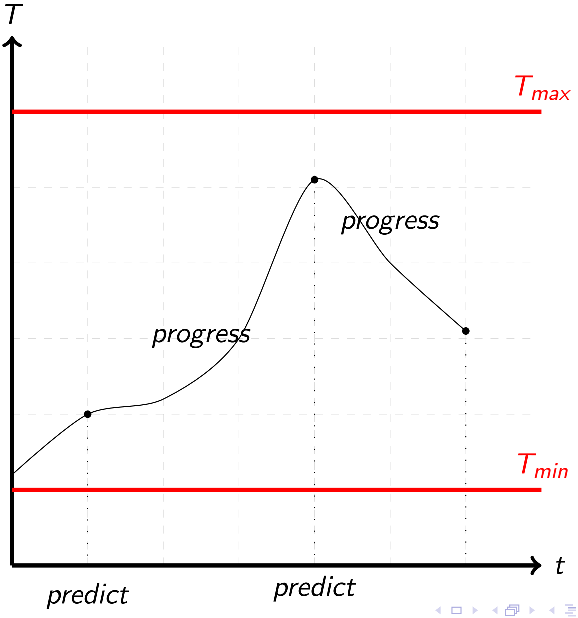


Smart Heating System (Cycle M2)

Checklist:

- Time pointer
- Refined system state trajectory
- Refined safety property
- Small-step semantics

Smart Heating System (Close-loop M3)



Smart Heating System (Close-loop M3)

Checklist:

- Variable for close-loop mode control
- Prediction (Controller)
- Progression (Plant)

Smart Heating System (Control Logic M4)

Time-triggered

Goal:

- Assuming the controller takes place in a safe system state
- Assuming exists a specification of system dynamics
- Planning for a trajectory that is safe before the controller takes place next time

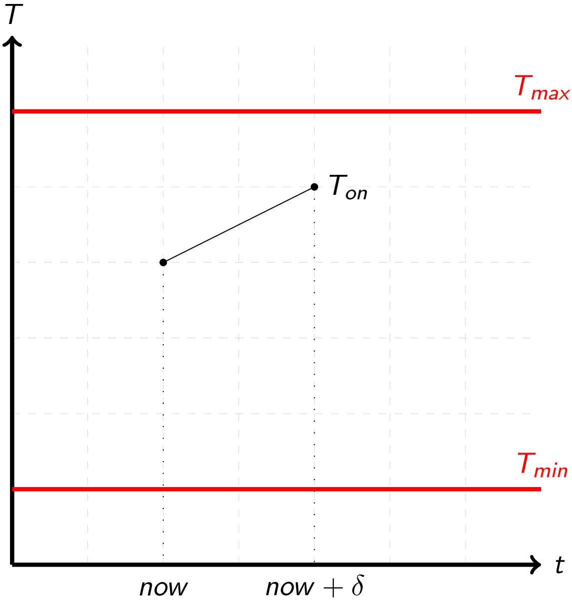
Smart Heating System

Sub-system Specification

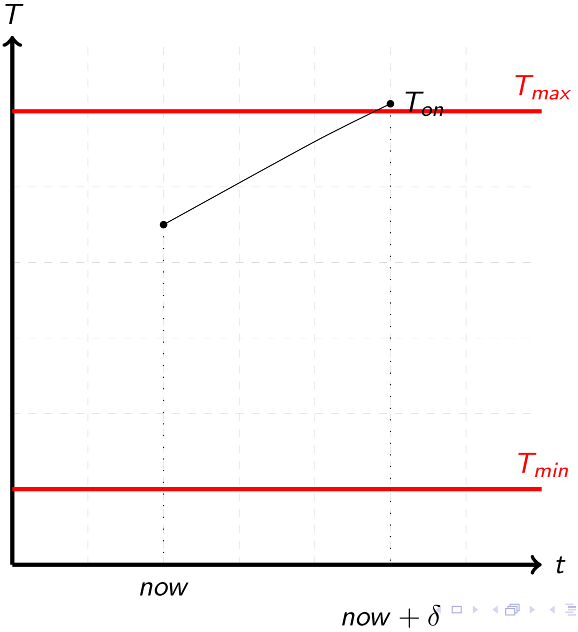
A specification for the dynamics of heating system:

- mode ON: monotonically increasing
  $(\forall t1, t2 \cdot t1 \geq t2 \rightarrow T(t1) \geq T(t2))$
- mode OFF: monotonically decreasing
  $(\forall t1, t2 \cdot t1 \geq t2 \rightarrow T(t1) \leq T(t2))$

Case 1: ON mode, $T(now + \delta) \leq T_{max}$, Stay ON



Case 2: ON mode, $T(now + \delta) > T_{max}$, TO OFF



Current Summary

Time-triggered Design in Event-B

Event $Prediction_i \;\widehat{=}$
  Refines $Prediction_i$
  Where ...
    $grd_1: \quad C_i(x)$
    $grd_2: \quad s = DECISION$
  Theorem
    $thm_1: \quad \forall t \cdot t \in (now, now + \delta] \Rightarrow Safe(x_{u_i}(t))$
  Then ...
    $act_1: \quad u, t_u := u_i, \delta$
    $act_2: \quad s := RUN$
End

Decomposing Time-triggered Design

Sensing: modeling sensor imperfections

Machine M_IMPL
Refines M_TIME_TRIGGERED
Variables x x_s ...
Invariants
  $inv_{xs}: \quad R_s(x_s, x)$
Events
  Event $Sense \ \widehat{=}$
  Where
    $grd_2: \quad s = SENSE$
  Then
    $act_1: \quad x_s :| \ R_s(x_s', x)$
    $act_2: \quad s := DECISION$
  End
  ...
End

Decomposing Time-triggered Design

Actuate: modeling actuator configurations

Machine M_IMPL
Refines M_TIME_TRIGGERED
Invariants
inv_{ud} $R_a(u_d, u)$
Events
Event $Actuate$ $\widehat{=}$
Where
$grd_1: \quad s = ACTUATE$
Then
$act_1: \quad u :\mid R_a(u_d, u')$
$act_2: \quad s := RUN$
End
...
End

Decomposing Time-triggered Design

Control

Event $Control_i \;\widehat{=}$
Refines $Prediction_i$
Where
$grd_1: \;\; CC_i(x_s)$
$grd_2: \;\; s = DECISION$
Then
$act_1: \;\; u_d \;,\;\; t_u \;:=\; u_{d_i} \;,\;\; \delta$
$act_2: \;\; s := ACTUATE$
End

Modularize Time-triggered Design

First step: extracting predicates from associated events

Event $Control_i \ \widehat{=}$
Refines $Control_i$
Where
$grd_1: \ CC_i(x_s)$
$grd_2: \ s = DECISION$
Then
$act_1: \ u_d, t_u :| \ CC_i(x_s) \Rightarrow u'_d = u_{d_i} \wedge t'_u = \delta$
$act_2: \ s := ACTUATE$
End

Modularize Time-triggered Design

Second step: based on the extraction, generating operations to-be-implemented

Operation $f_c \;\widehat{=}$
Parameters x_s
Returns u_d, t_u
Axioms

$$\wedge_i \left(CC_i(x_s) \Rightarrow u_d = u_{d_i} \wedge t_u = \delta \right)$$

End

Event $Control_i \;\widehat{=}$
Refines $Control_i$
Where
$grd_1 : \; CC_i(x_s)$
$grd_2 : \; s = DECISION$
Then
$act_1 : \; u_d, t_u :| \; f_c(x_s) = (u'_d, t'_u)$
$act_2 : \; s := ACTUATE$
End

Modularize Time-triggered Design

Third step: merge events

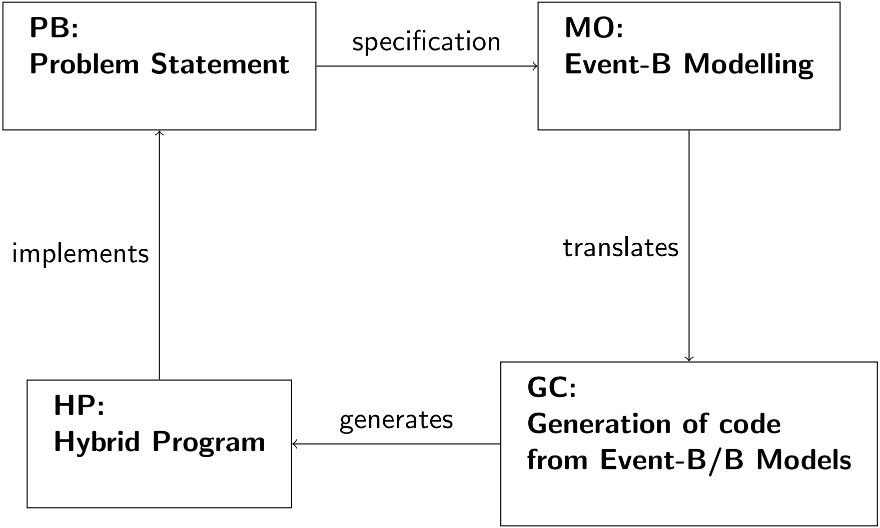
Event *Control* $\widehat{=}$
Refines $Control_1$, . . . , $Control_i$
Where
grd_1 : $CC_1(x_s) \lor \ldots \lor CC_i(x_s)$
grd_2 : $s = DECISION$
Then
act_1 : $u_d, t_u := f_c(x_s)$
act_2 : $s := ACTUATE$
End

Methodological Issues

- *Sense* and *Actuate* events use two predicates $R_s(x_s, x)$ and $R_a(u_d, u)$.
- Making *explicit* information on the sensing preciseness or on the actuating effectiveness is related to the domain analysis and formalization.
- The domain experts may provide a list of properties or assumptions on those predicates (frameworks as ontologies or knowledge domains).
- The *Control* event emphasizes on concrete control logic development based on the digitized data (i.e. the observed state x_s, and the discrete actuation command u_d).

Current Summary

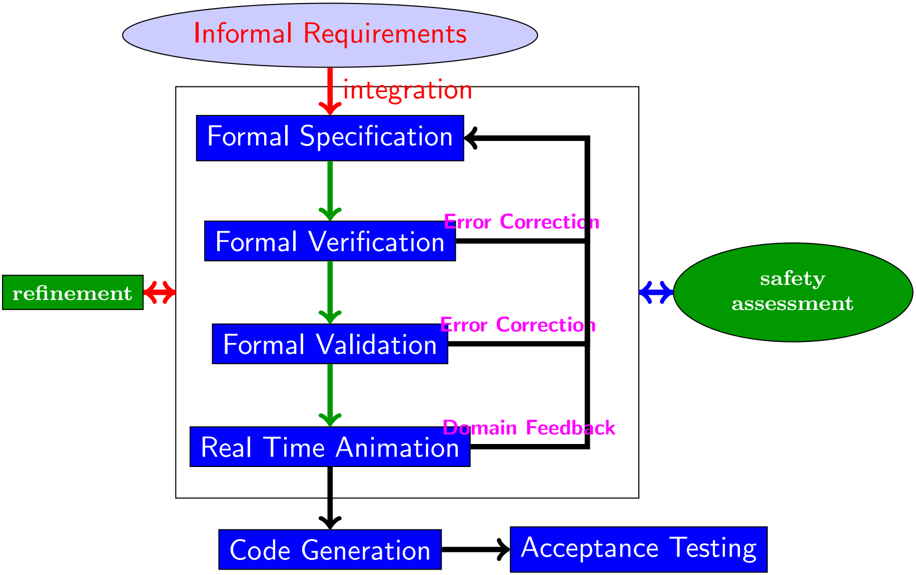
The four boxes diagram



Discussion

- **Problem Statement** - hybrid action systems (or hybrid automata or mathematics) with mathematical theories (reals, continuity, differentiability, ODEs, . . .)
- **Event-B Modelling**:
  - Dupont's approach by using and instantiating patterns and theories for event-triggered models with a link to Simulink code: CBAP predicate for continuous events and proofs are fully automated.
  - Mammar's approach by using and instantiating patterns and theories for event-triggered and time-triggered events models using dRL rules: one event models the time progress and some proofs are not automated.
  - Our approacs based on the same assumptions than Dupont's (no event for time progression); we use an notion of refinement close to the differential dynamic logic and and proofs are fully automated.
  - specification - the process is mainly directed by the informations available in the problem statement and the box incrementally *feeds* the refinement steps.

Refinement-based Specification



Discussion

- **Generation of Code** - The general process is based on the refinement of B machines into B0 implementation and is correct by experience.
- **Hybrid Program** - The activity aims to generate artifacts to validate the implementation generated from GC, such as:
  - code for Frama-C , and Polyspace to check against certain industry code standards (e.g. reachability, absence of non-determinism, absence of runtime error).
  - simulation models for Simulink and Stateflow to give a holistic view of the developed hybrid system.

Conclusion and Perspectives

- Uniform framework for designing a rich time-triggered Event-B model using the Rodin platform.
- Sound transformation of the control part into an operation of B.
- Enriching the picture at the different box.
- Develop Case Studies.
- Certification of each box MO et GC.

Bibliography

- Zheng Cheng, Dominique Méry: A Refinement Strategy for Hybrid System Design with Safety Constraints. MEDI 2021: 3-17
- A Refinement Strategy for Hybrid System Design with Safety Constraints Zheng Cheng, Dominique Méry [Research Report] Université de Lorraine; INRIA; CNRS. 2020 https://hal.inria.fr/hal-02895528/file/merymain.pdf
- Yamine Aït Ameur, Dominique Méry: Making explicit domain knowledge in formal system development. Sci. Comput. Program. 121: 100-127 (2016)
- Dines Bjørner: Domain Science and Engineering - A Foundation for Software Development. Monographs in Theoretical Computer Science. An EATCS Series, Springer 2021, ISBN 978-3-030-73483-1, pp. 3-319
- Dines Bjørner: Domain Analysis and Description Principles, Techniques, and Modelling Languages. ACM Trans. Softw. Eng. Methodol. 28(2): 8:1-8:67 (2019)