UNIVERSITY OF Southampton

Real-time deadlines and delays in Event-B

Michael Butler, Chenyang Zhu, Corina Cirstea

IFIP WG 2.3 Meeting, Mooloolaba,
July 17-21 2017

Deadlines and delays

- Example deadlines:
  - Within 300ms of the brake pedal being pressed, the vehicle brakes shall be activated
  - Within 500ms of the train entering route R, the signal for R shall turn red
- Example delays:
  - The sender shall wait at least 100ms for a response before retrying the send
  - It takes at least 10 seconds for the aircraft landing gear to be extended
- Deadline: upper bound on duration
- Delay: lower bound on duration

Clarification

- Deadline/delays are typically based on on assumptions about dynamics
  - E.g., up to 300ms will not significantly increase the actual braking distance beyond the driver's perceived braking distance
  - We are not trying to analyse the justification for the deadlines/delays based on the dynamics

- Instead we focus on
  - Capturing timing requirements precisely as part of Event-B model
  - Verifying consistency between end-to-end timing requirements and timing requirements on individual components or phases (decomposition)
  - Verifying consistency between safety requirements and timing requirements
  - … using discrete time

Deadline patterns

- Trigger-Response pair
  - Trigger event: driver presses brake pedal
  - Response event: vehicle brakes activated

- Alternative responses
  - Trigger: pilot commands landing gear extended
  - Alternative responses:
    - Landing gear extended safely, ***or***
    - Pilot warned of failure

Simple Event-B example

PressPedal $\widehat{=}$
when
    pedal = FALSE
then
    pedal := TRUE
end

ActuateBrake $\widehat{=}$
when
    pedal = TRUE
    brake = FALSE
then
    brake := TRUE
end

- Response event is enabled by execution of trigger event
- (For simplicity) we assume response event occurs before any subsequent occurrence of trigger event

Deadline declaration

PressPedal $\widehat{=}$
when
    pedal = FALSE
then
    pedal := TRUE
end

ActivateBrake $\widehat{=}$
when
    pedal = TRUE
    brake = FALSE
then
    brake := TRUE
end

- Extend Event-B model with deadline specification:

deadline(PressPedal, ActivateBrake, d)

- Meaning: if *PressPedal* event occurs at time t, then *ActivateBrake* event must occur no later than time t+d

Alternative responses

- Assume Event-B model has events A, $B_1..B_N$ (plus possibly other events)
- Assume execution of A enables one of B_1 to B_N
- Deadline declaration:

  deadline(A, $B_1 | \ldots | B_N$, d)

Event Sequencing

```
A ≙
when
      oA = FALSE
then
      oA : = TRUE
end
```

```
B ≙
when
      oA = TRUE
      oB = FALSE
then
      oB := TRUE
end
```

- Execution of A enables B

Adding deadline declaration

A ≙
when
oA = FALSE
then
oA : = TRUE
end

B ≙
when
oA = TRUE
oB = FALSE
then
oB := TRUE
end

deadline(A, B, d)

(for simplicity) d is a constant

Specification is Event-B model + deadline

Encoding deadline with clock variable

A $\triangleq$
when
    oA = FALSE
then
    oA : = TRUE
    trg : = clk
end

B $\triangleq$
when
    oA = TRUE
    oB = FALSE
then
    oB := TRUE
    rsp := clk
end

Tick $\triangleq$
when
    ???
then
    clk := clk+1
end

invariants

inv1: $0 \le trg \le clk \ \wedge \ 0 \le rsp \le clk$

inv2: $trg \le rsp \ \Rightarrow \ rsp\text{-}trg \le d$ // upper bound

Additional Invariant

A $\triangleq$
when
    oA = FALSE
then
    oA : = TRUE
    trg : = clk
end

B $\triangleq$
when
    oA = TRUE
    oB = FALSE
then
    oB := TRUE
    rsp := clk
end

Tick $\triangleq$
when
    ???
then
    clk := clk+1
end

invariants
    inv2: trg ≤ rsp $\Rightarrow$ rsp-trg ≤ d
    inv3: oA=TRUE $\wedge$ oB=FALSE $\Rightarrow$ clk−trg ≤ d

- inv3 is required to ensure that B maintains inv2

Tick guard

A ≙
when
    oA = FALSE
then
    oA : = TRUE
    trg : = clk
end

B ≙
when
    oA = TRUE
    oB = FALSE
then
    oB := TRUE
    rsp := clk
end

Tick ≙
when
    inv3(clk+1)
then
    clk := clk+1
end

invariants

inv2: $trg \le rsp \Rightarrow rsp\text{-}trg \le d$

inv3: $oA=TRUE \wedge oB=FALSE \Rightarrow clk-trg \le d$

Time must progress

- Event-B makes no fairness assumption
- Minimal progress assumption:
  - keep executing some enabled event unless all events are disabled
- Need to treat *Tick* event as special:
  - Every infinite trace has infinitely many *Tick* events

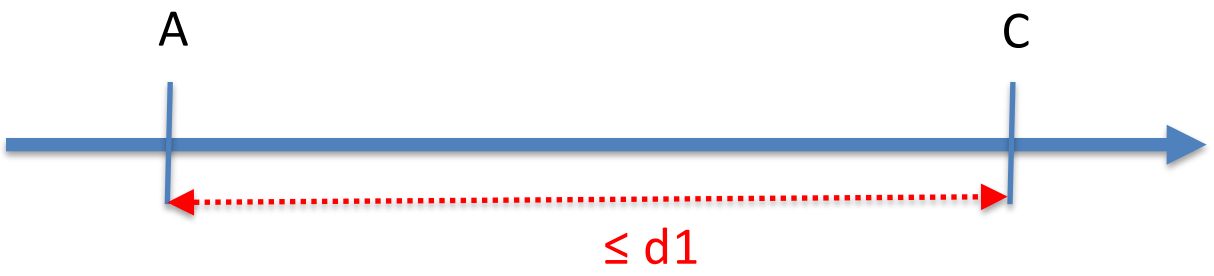
Enabledness of deadline response

- Deadline could cause *Tick* to become disabled forever
- Can prevent this with *enabledness* condition on response events:

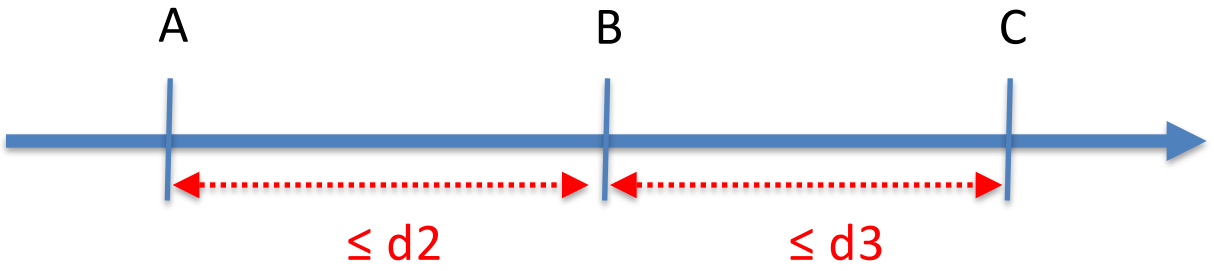
$$post(A) \Rightarrow grd(B_1) \vee \ldots \vee grd(B_N)$$

- In refinements of the model, we can introduce additional events between trigger and response

Deadline decomposition as refinement



Refine by:



Refining deadline declarations

deadline(A, C, d1)

Refined by

deadline(A, B, d2)

deadline(B, C, d3)

$d2+d3 \leq d1$

Decomposition with alternate responses

deadline(A, C_1|C_2, d1)

Refined by

deadline(A, B_1|B_2, d2)

deadline(B_1, C_1, d3)

deadline(B_2, C_2, d3)

d2+d3 ≤ d1

Periodic Events

```
B  ≙
when
      G
then
      M
      trg : = clk
end
```

```
skipB  ≙
when
      ¬G
Then
      trg := clk
end
```

```
Tick  ≙
when
      clk < trg+p
then
      clk  := clk+1
end
```

periodic(B|skipB, p)

Execute events B or skipB at least every p time units

Refinement: deadline to periodic

deadline(A, B, d)

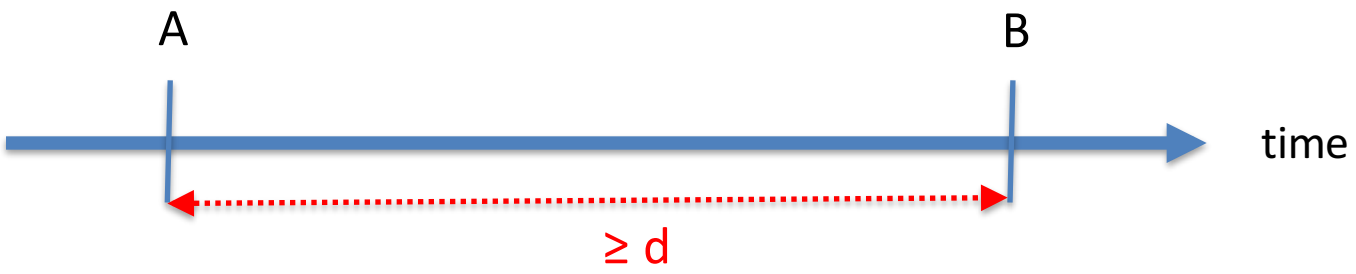
Refine by

periodic(B|skipB, p)

$p \leq d$

Delay

delay(A, B, d)



If A occurs at time t, then B cannot occur earlier than time t+d

Encoding delay

delay(A, B, d)

A ≙
when
    oA = FALSE
then
    oA := TRUE
    trg := clk
end

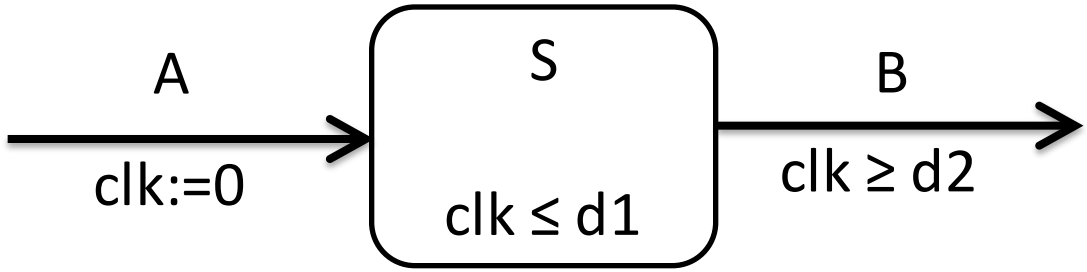
B ≙
when
    oA = TRUE
    oB = FALSE
    clk-trg ≥ d
then
    oB := TRUE
    rsp := clk
end

invariants

inv: trg ≤ rsp ⇒ rsp-trg ≥ d

Note: delay does not constrain Tick, so response might not occur

Timed automata idioms



Compare with

deadline(A, B, d1)
delay(A, B, d2)

Deadline and liveness

- Process p_0 to p_N can access a critical section
- *Liveness*: wish(p) **leads to** enter(p)
- *Timing*: deadline(wish(p), enter(p), w)
- Refine timing using a FIFO queue
  - deadline(enter(p), leave(p), c)
  - $w \geq N \times c$ // upper bound (worst case)

(Simplified) railway control

Safe state

We assume train braking distance is less than a block and train controller will apply brakes when necessary

Unsafe

Collision possible

Braking

Safe provided T1 starting braking in time

Abstract model

- Each train has a front and a back block (which might be the same):

  inv1: $front \in train \rightarrow BLOCK$

  inv2: $back \in train \rightarrow BLOCK$

- Each block is occupied by at most one train:

  inv3: $occupy = (front \cup back)^{-1}$

  inv4: $occupy \in BLOCK \nrightarrow train$

Abstract event

EnterBlock $\widehat{=}$
any t, b where
    front(t) = back(t)
    b = next(front(t))
    $b \notin dom(occupy)$ "magically" enforce safety
then
    front(t) := b
    occupy(b) := t
end

Timing assumptions

- *lower* bound on time it takes train to traverse a block
  - delay(EnterBlock(t), EnterBlock(t), d1)
- *upper* bound on time it takes train to stop once brakes applied:
  - deadline(Brake(t), Stop(t), d2)

Apply brakes asap

- deadline on time it takes apply brakes
  - deadline(EnterBlock(t), Brake(t), d3)
  - d3+d2 ≤ d1

- This is too strong!
  - requires braking even if next block is not occupied

Conditional deadline

Brake(t) $\widehat{=}$
when
    next(front(t)) $\in$ dom(occupy)
then
    braking := braking $\cup$ {b}
end

NoBrake(t) $\widehat{=}$
when
    next(front(t)) $\notin$ dom(occupy)
then
    skip
end

- deadline(EnterBlock(t), Brake(t) | NoBrake(t), d3)

Replacing the magic

EnterBlock(t,b) ≙
when
    front(t) = back(t)
    b = next(front(t))
    b ∉ dom(occupy)
then
    front(t) := b
    occupy(b) := t
end

refined EnterBlock(t,b) ≙
when
    front(t) = back(t)
    b = next(front(t))
    t ∉ stopped
then
    front(t) := b
    occupy(b) := t
end

delay(EnterBlock(t), EnterBlock(t), d1) \\ assumption
deadline(Brake(t), Stop(t), d2) \\ assumption
deadline(EnterBlock(t), Brake(t) | NoBrake(t), d3) \\ requirement
d3+d2 ≤ d1

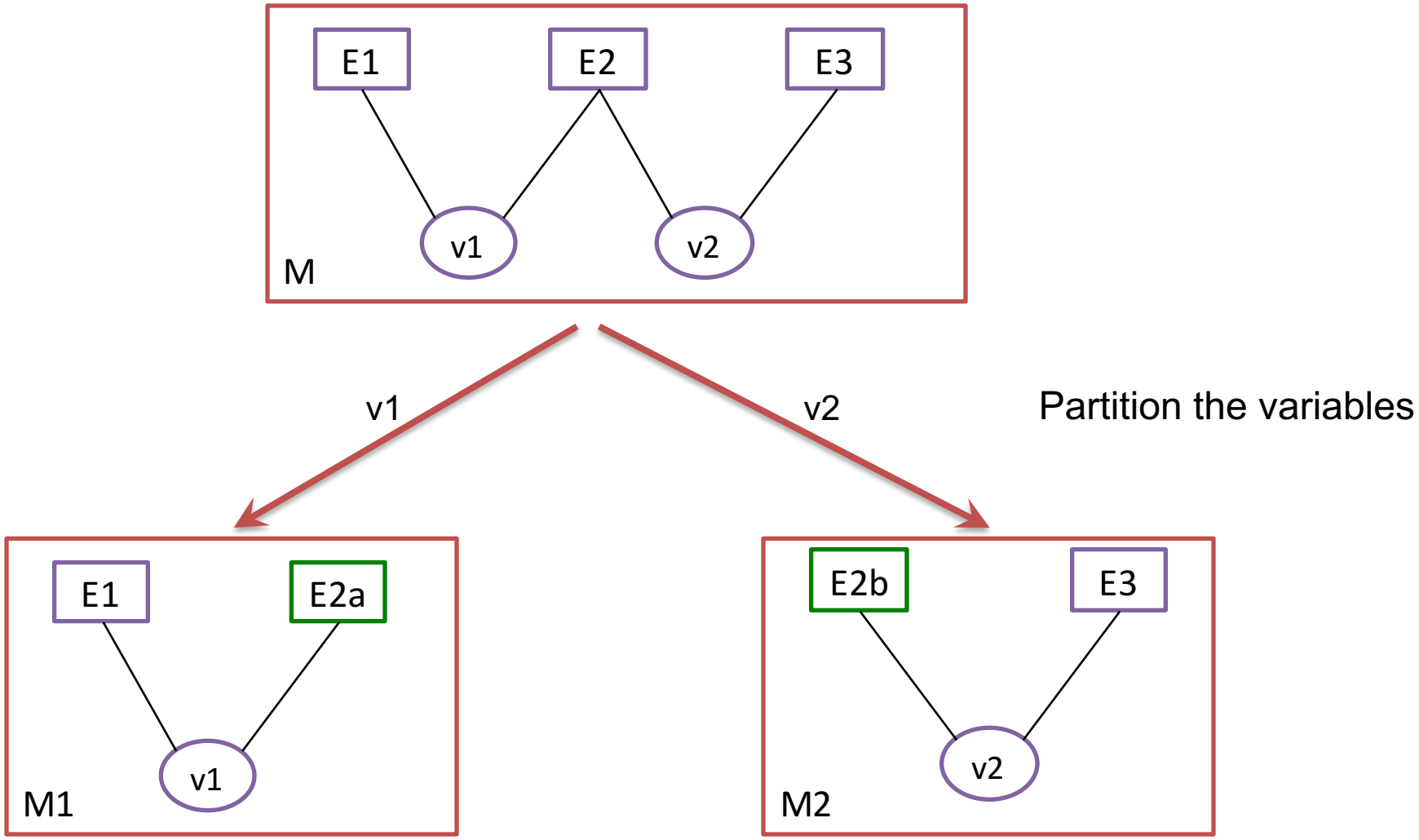
Invariants

$$\forall t \cdot t \notin stopped \ \wedge \ trgEnter(t)+d1 \leq clk$$
$$\Rightarrow next(front(t)) \notin dom(occupy)$$

$$\forall t \cdot t \in running \ \wedge \ trgEnter(t)+d3 \leq clk$$
$$\Rightarrow next(front(t)) \notin dom(occupy)$$

$$\forall t \cdot t \in braking \ \Rightarrow \ clk \leq trgBrake(t)+d2$$

Model Decomposition



Partitioning events

```
E =
any  p  where
      G1( v1, p )
      G2( v2, p )
then
      v1 := H1( v1, p )
      v2 := H2( v2, p )
end
```

```
Ex =
any  p  where
      G1( v1, p )
then
      v1 := H1( v1, p )
end
```

```
Ey =
any  p  where
      G2( v2, p )
then
      v2 := H2( v2, p )
end
```

Pre-partitioning

```
E =
any  p  where
     G1( v1, p, f(v2) )
     G2( v2, p )
then
     v1 := H1( v1, p, f(v2) )
     v2 := H2( v2, p )
end
```

```
E =
any  p, q  where
     q = f(v2)
     G1( v1, p, q )
     G2( v2, p )
then
     v1 := H1( v1, p, q )
     v2 := H2( v2, p )
end
```

Transform E to make it easier to split into *v1*-part and *v2*-part

Timing and machine decomposition

```
Tick  ≙
when
      G1(v1,clk)
      G2(v2,clk)
then
      clk  := clk+1
end
```

Decompose by partition variables v1, v2

How to decompose clk?

Timing and machine decomposition

Tick ≙
when
    G1(v1,clk)
    G2(v2,clk)
then
    clk := clk+1
end

refine

Tick ≙
when
    D1(v1,clk1)
    D2(v2,clk2)
then
    clk1 := clk1+1
    clk2 := clk2+1
end

Invariant
clk1 = clk2

decompose

Tick1 ≙
when
    D1(v1,clk1)
then
    clk1 := clk1+1
end

Tick2 ≙
when
    D2(v2,clk2)
then
    clk2 := clk2+1
end

Interleaving timing and functional refinement

- Functional and timing refinements can be intermingled, e.g.,
  1. End-to-end event flow
  2. Impose end-to-end deadline
  3. Refine end-to-end event flow, retain end-to-end deadline
  4. Refine end-to-end deadline to decomposed deadlines

PhD Thesis: Reza Sarshogh, 2013

- Augment Event-B with 3 discrete timing constructs:
  - deadline, delay, expiry
- Developed some refinement patterns
  - Sequential split of deadline
  - Sequential of deadline with multiple paths
  - Abstract ordering to delay + deadline
- Developed refinement patterns
- Developed prototype Rodin plug-in

- http://eprints.soton.ac.uk/354385/

Continuous time

Butler, Abrial, Banach (2016) Modelling and Refining Hybrid Systems in Event-B and Rodin

In, Petre, Luigia and Sekerinski, Emil (eds.) *From Action System to Distributed Systems: The Refinement Approach.* Taylor & Francis doi:10.1201/b20053-5