Timing diagrams add Requirements Engineering capability to Event-B Formal Development

Tossaporn Joochim

Supervisor : Dr. Michael R. Poppleton

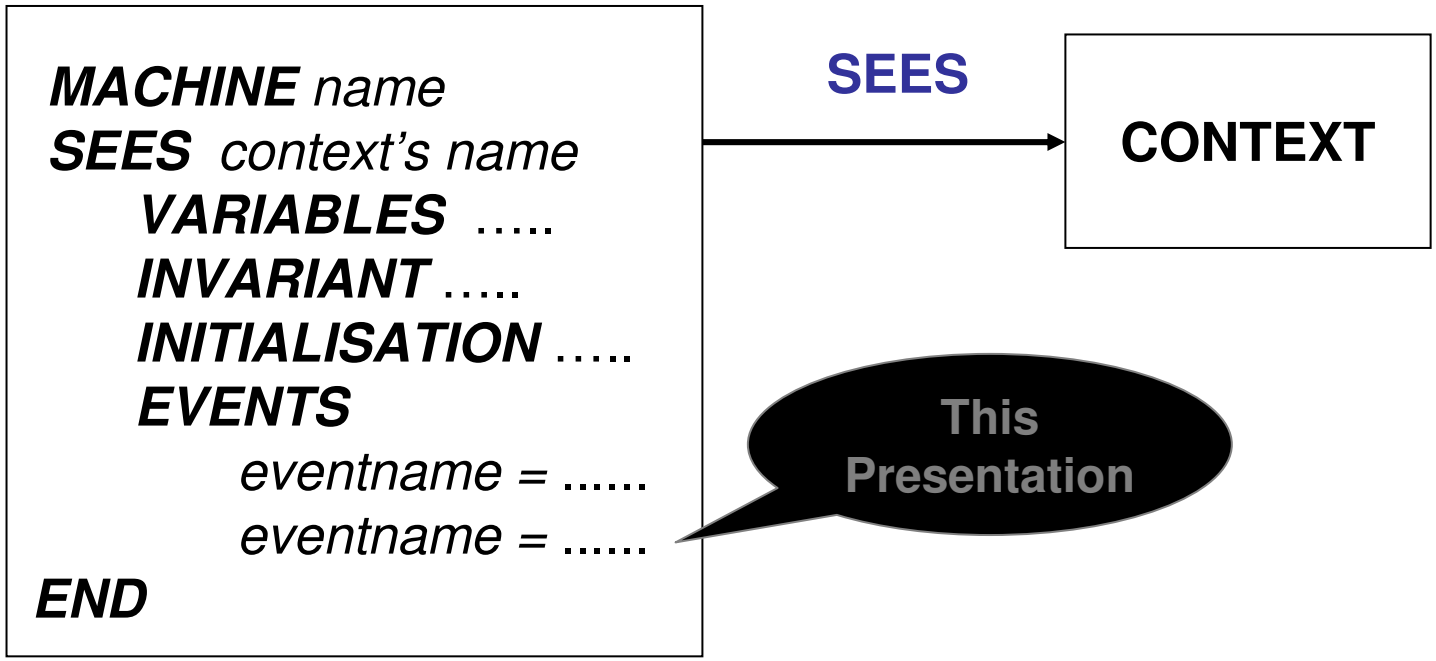
Dependable System and Software Engineering group
School of Electronics and Computer Science
University of Southampton

Outline

- Event-B model structure
- Timing diagrams
  - Notations
  - Case study : The Lift System
- Pattern to transform Timing diagrams into an Event-B model
  - BNF definitions
  - Translation rules
- Conclusions and Future work

Event-B

- Structure of Event-B



Event-B (cont')

- The general form of an event is

$$E = \textbf{ANY } l \textbf{ WHERE } G(l,v) \textbf{ THEN } S(l,v) \textbf{ END} \qquad (1)$$

- *A short form of an event omitting local variables is*

$$E = \textbf{WHEN } G(v) \textbf{ THEN } S(v) \textbf{ END} \qquad (2)$$

Case study : The Lift System

Some specifications are described as follows

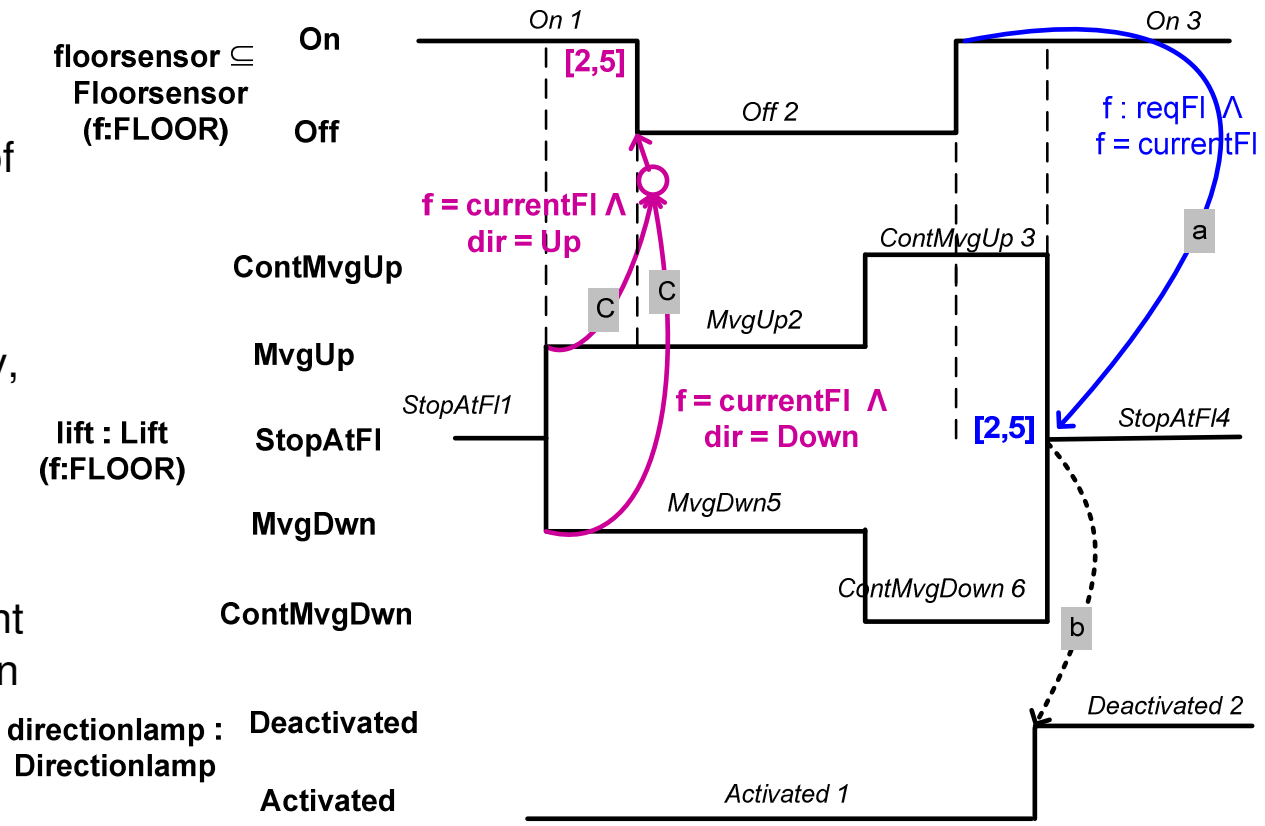
a) The moving lift will be stopped at the requested floor within a time interval of 2 – 5 seconds after floor sensor is set on.

b). If the lift becomes stationary, the direction lamp must be deactivated immediately.

c). Whenever the lift starts moving up/down, the current floor sensor will be off within a time interval of 2 – 5 seconds after the lift starts moving.

…….

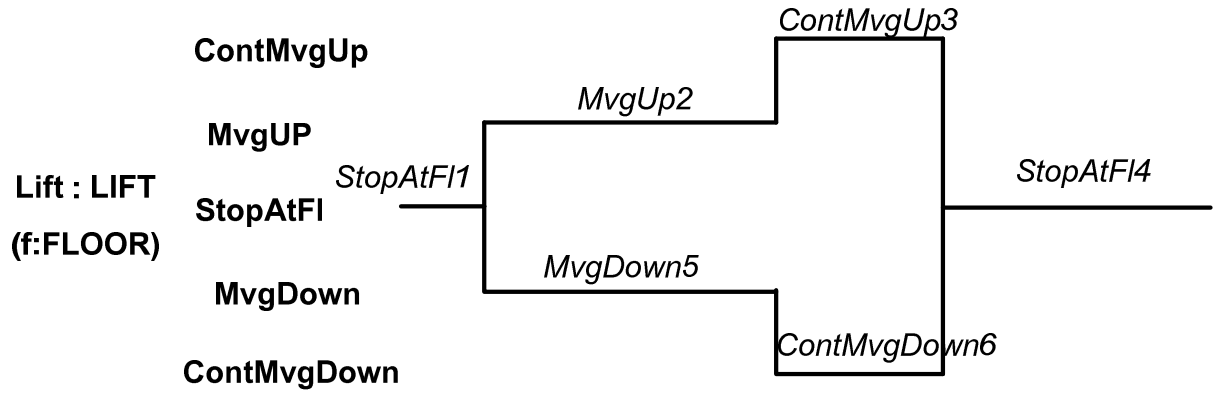
…….



Parts : BNF Timing Diagrams definitions

A Timeline comprises a chain of segments which individual segment represents the object state (Objst) and its position (Index) in the Timeline.

Timeline ::= Segment^{+}
Segment ::= Objst Index
Index ::= integer

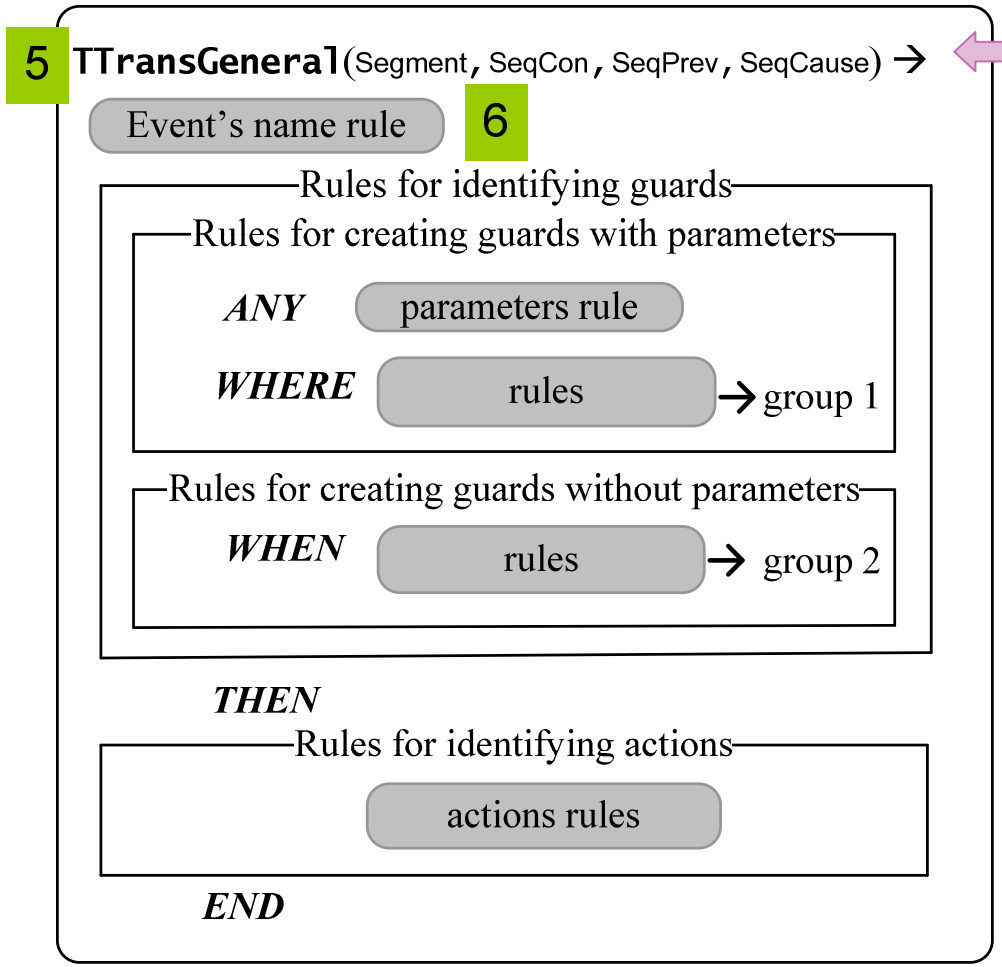


Translation rules

Basic rules :

Rule 1 : **TState** (Segment) → Objst; This rule gives the object state for an input segment.
Rule 2: **TObject**(Segment) → Obj; This rule gives the object for an input segment.
Rule 3: **TClass**(Obj) → Class This rule gives the class for an input object.
Rule 4: **TParam**(Class) → SqParam This rule gives the sequence of parameters for an input class.

Rules for creating abstract model's events



Rule 5 : rule for generating an abstract model's event.

TTransGeneral (Segment,SeqCon, SeqPrev, SeqCause)

Where :

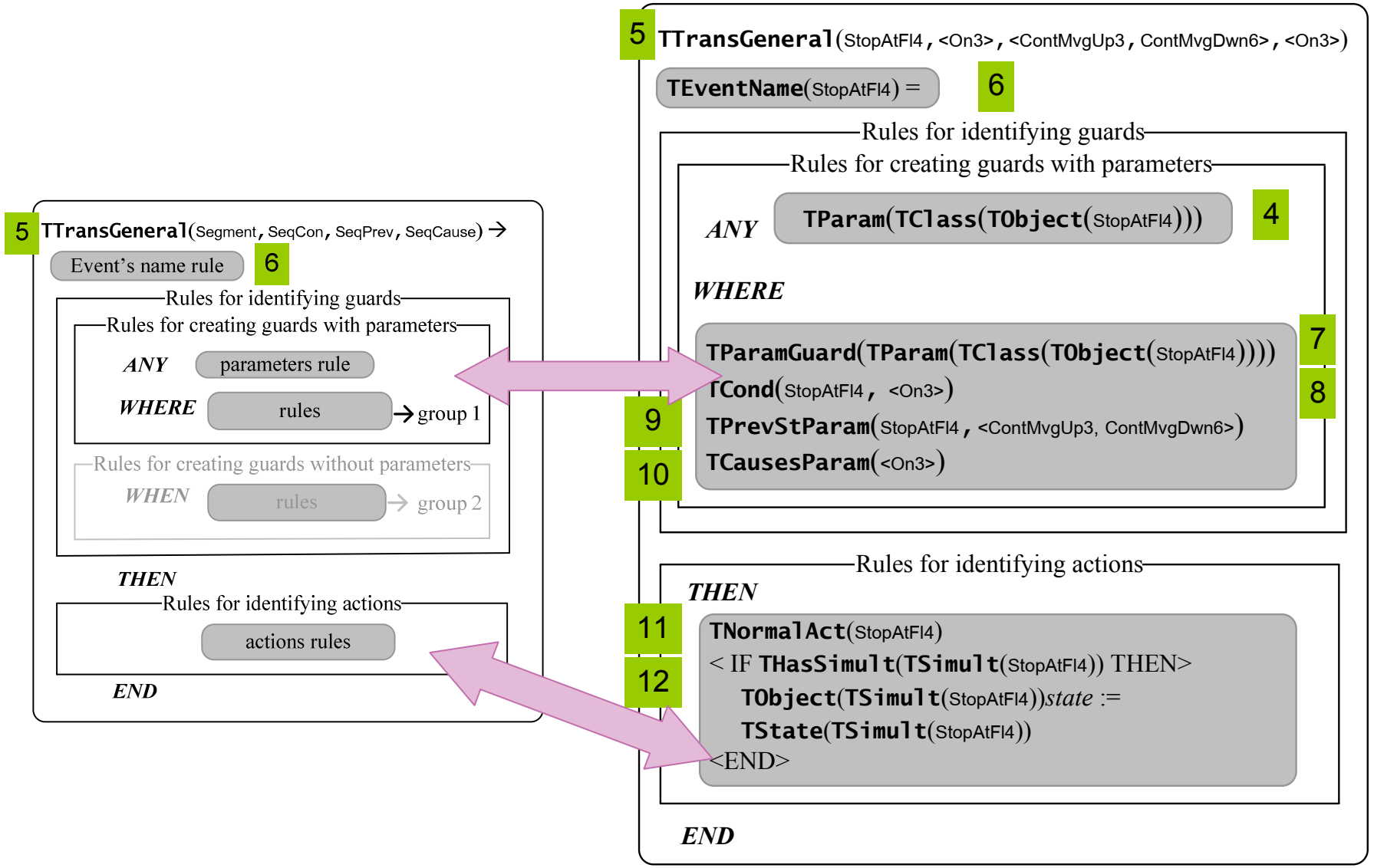
Segment : a segment,

SeqCon : a sequence of condition segments

SeqPrev : a sequence of previous segments

SeqCause : a sequence of cause segments

Example : Using rules for generating *liftStopAtFl* event



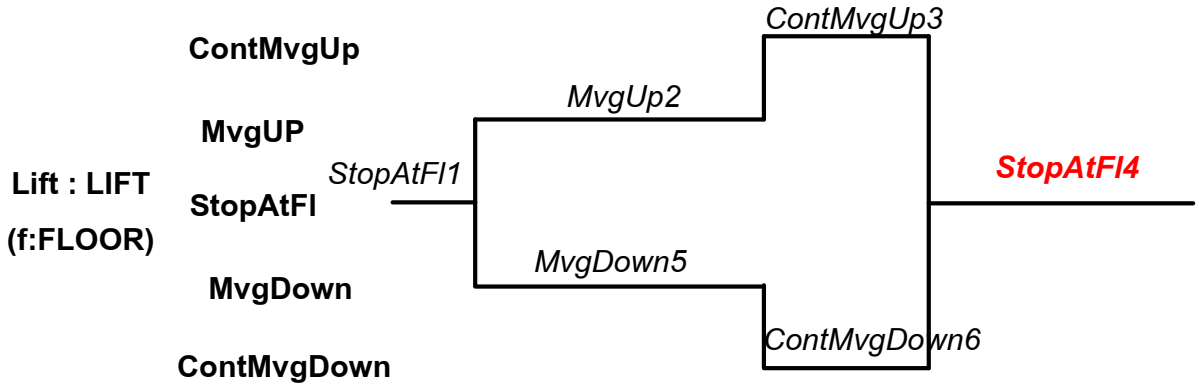
Rule 9 : TPrevStParam(Seqment, SeqPrev)

TPrevStParam(Segment, < >) = “ ”

TPrevStParam(Segment, SeqPrev) →
<IF **MultPrev**(Segment, SeqPrev) THEN >
TPrevStParam(Segment, SeqPrev) =
TPrevStParamR(Segment, Head : SegmSeqTail) →
TObject(Segment)*state (***TParamLt**(**TParam**(**TClass**(**TObject**(Segment)))) *)* =
TState(Head) ∨ **TPrevStParam**(Segment, SegmSeqTail)

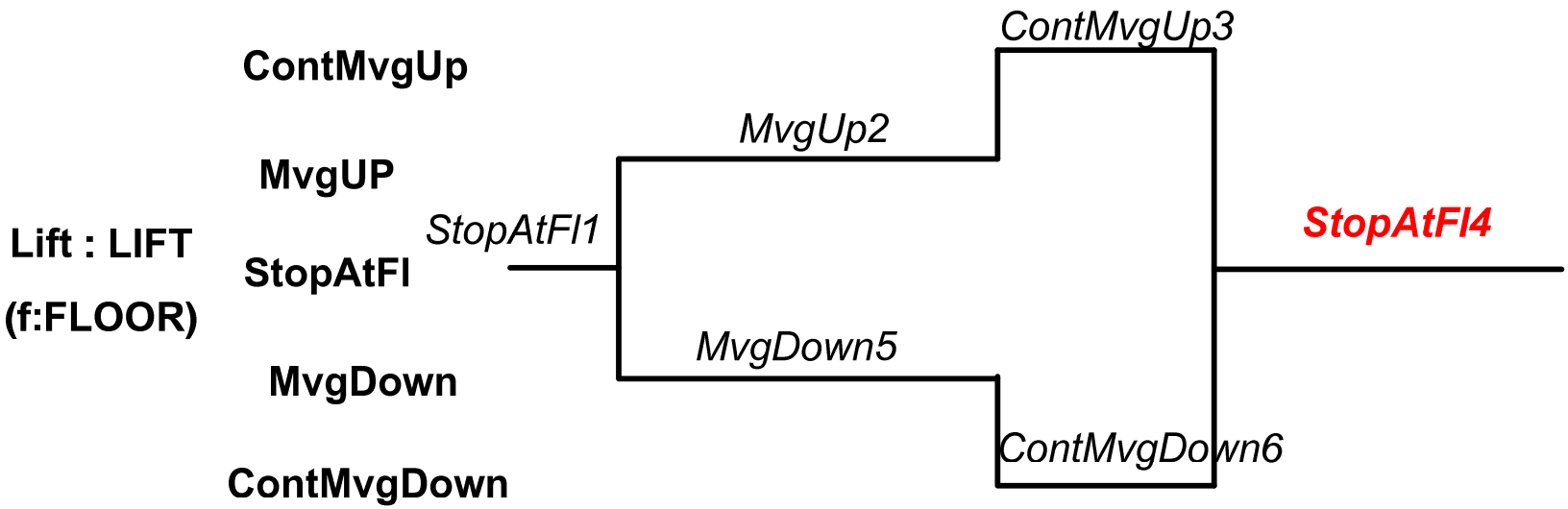
< ELSE>
TObject(Segment)*state (* (**TParamLt**(**TParam**(**TClass**(**TObject**(Segment)))) *)* =
TState(**Elem**(SegmSeqTail))

< END>



Seqment **SeqPrev**

TPrevStParam(StopAtFl4, < ContMvgUp3, ContMvgDwn6 >)



Example: Using rule 9 (Cont')

1st recursion

TPrevStParam(Segment, < >) = " "

TPrevStParam(StopAtFl4, < ContMvgUp3, ContMvgDwn6 >) →
<IF **MultPrev**(StopAtFl4, < ContMvgUp3, ContMvgDwn6 >) THEN >
TPrevStParam(StopAtFl4, < ContMvgUp3, ContMvgDwn6 >) =
TPrevStParamR(StopAtFl4, ContMvgUp3 : <ContMvgDwn6 >) →
TObject(StopAtFl4)*state (***TParamLt**(**TParam**(**TClass**(**TObject**(StopAtFl4)))) =
TState(ContMvgUp3) ⋁ **TPrevStParam**(StopAtFl4, <ContMvgDwn6 >)

2nd recursion

TPrevStParam(StopAtFl4, <ContMvgDwn6 >) →
< ELSE>
TObject(StopAtFl4)*state (* (**TParamLt**(**TParam**(**TClass**(**TObject**(StopAtFl4)))) *)* =
TState(**Elem**(<ContMvgDwn6 >)
< END>

Output: *liftstate(f) = ContMvgUp* ⋁ *liftstate(f) = ContMvgDwn*

Example : l*iftStopAtFl* Abstract model's event

liftStopAtFl =

ANY *f*

WHERE *f : FLOOR* */* rule 7*/*

f : reqFl f = currentFl */* rule 8*/*

liftstate(f) = ContMvgUp

liftstate(f) = ContMvgDown } ***/* rule 9*/***

floorsensorstate(f) = On */* rule 10*/*

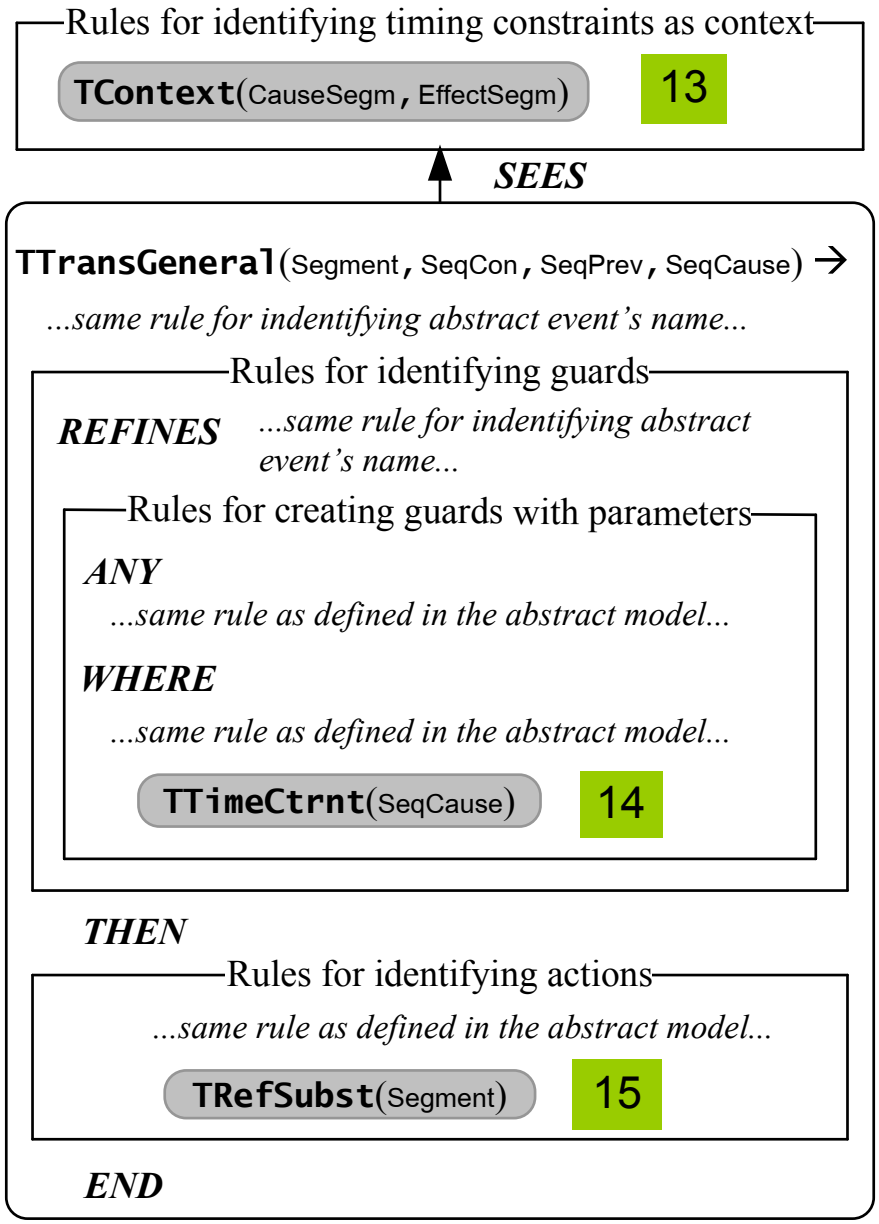
THEN

liftstate(f) := StopAtFl */* rule 11*/*

directionlampstate := Deactivated */* rule 12*/*

END

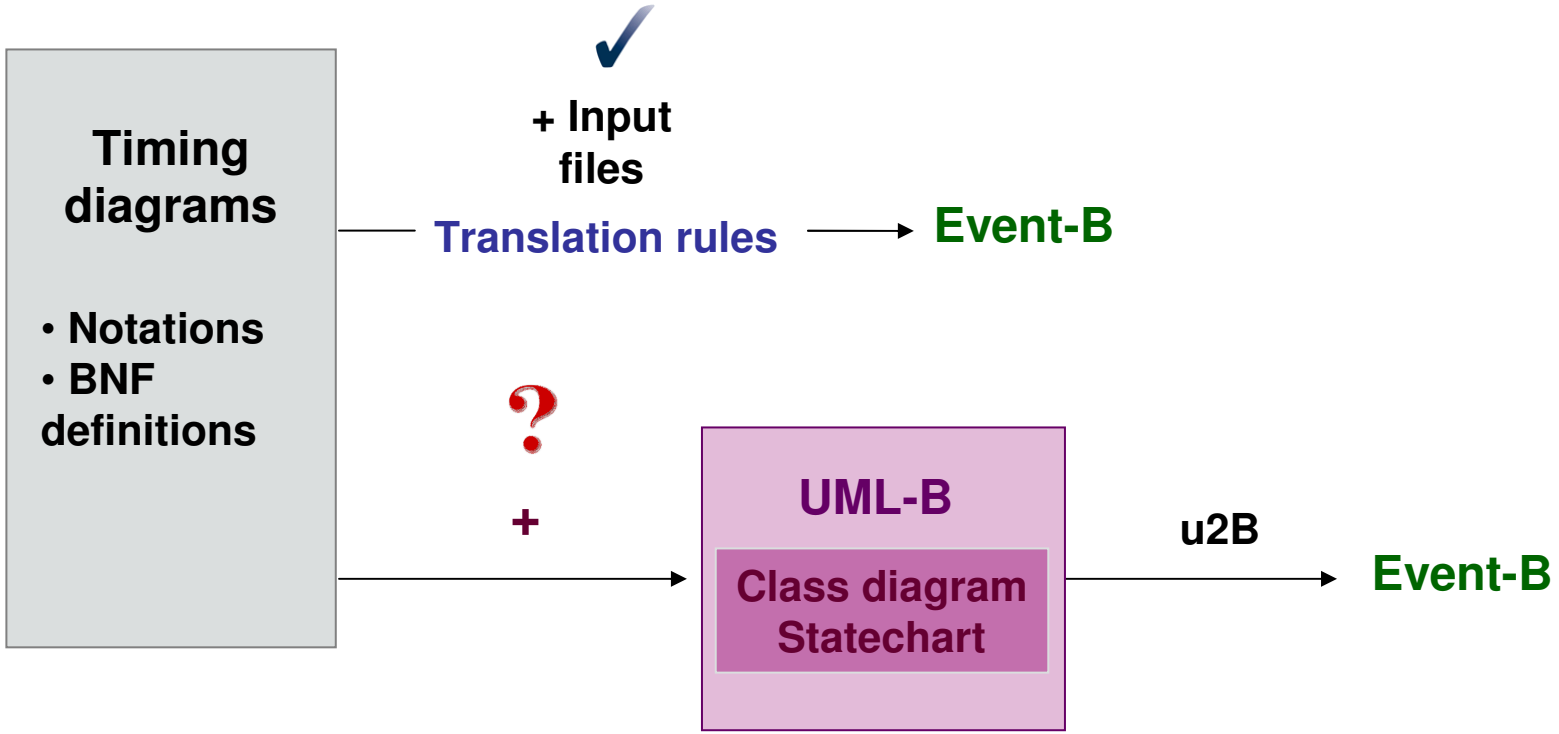
Rules for creating refinement model's events



Example : liftStopAtFl refinement model's event

liftStopAtFl =
REFINE *liftStopAtFl*
ANY *f*
WHERE
… same guard as defined in the abstract model…
(gclock – floorsensorOnTime)
LOWER_LIMIT_floorsensor &
(gclock – floorsensorOnTime)
UPPER_LIMIT_floorsensor
/ new guard from rule 14 */*
THEN
…same substitutions as defined in the abstract model…
liftStopAtFlTime := gclock
/ new substitution from rule 15 */*
END

Conclusions and Future work



Questions ?