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**

```
MACHINE name
SEES  context’s name
VARIABLES .....  
INVARIANT .....  
INITIALISATION .....  
EVENTS
  eventname = ......  
  eventname = ......  
END
```

SEES

CONTEXT

This Presentation
Event-B (cont’)

• The general form of an event is

\[ E = \textsc{any} \mid \textsc{where} \; G(l,v) \; \textsc{then} \; S(l,v) \; \textsc{end} \]  

(1)

• A short form of an event omitting local variables is

\[ E = \textsc{when} \; G(v) \; \textsc{then} \; S(v) \; \textsc{end} \]  

(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.

..................

..................
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:

<table>
<thead>
<tr>
<th>Rule</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule 1</td>
<td>TState</td>
<td>(Segment) → Objst; This rule gives the object state for an input segment.</td>
</tr>
<tr>
<td>Rule 2</td>
<td>TObject</td>
<td>(Segment) → Obj; This rule gives the object for an input segment.</td>
</tr>
<tr>
<td>Rule 3</td>
<td>TClass</td>
<td>(Obj) → Class; This rule gives the class for an input object.</td>
</tr>
<tr>
<td>Rule 4</td>
<td>TParam</td>
<td>(Class) → SqParam; This rule gives the sequence of parameters for an input class.</td>
</tr>
</tbody>
</table>
Rule 5 : rule for generating an abstract model’s event.

\[ \text{TTransGeneral} (\text{Segment, SeqCon, SeqPrev, SeqCause}) \rightarrow \]

**Event’s name rule**

Rules for identifying guards

- Rules for creating guards with parameters
  - \textit{ANY} \quad \text{parameters rule}
  - \textit{WHERE} \quad \text{rules} \rightarrow \text{group 1}

- Rules for creating guards without parameters
  - \textit{WHEN} \quad \text{rules} \rightarrow \text{group 2}

**THEN**

- Rules for identifying actions
  - \text{actions rules}

**END**

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 \textit{liftStopAtFl} event

\begin{itemize}
\item \texttt{TTransGeneral} (Segment, SeqCon, SeqPrev, SeqCause) \rightarrow
\item Event’s name rule
\item Rules for identifying guards
\item Rules for creating guards with parameters
\item ANY \quad parameters rule
\item \quad group 1
\item \quad rules
\item \quad \rightarrow group 2
\item WHEN \quad rules
\item THEN \quad Rules for identifying actions
\item \quad actions rules
\item END
\item \texttt{TTransGeneral}(\texttt{StopAtFl4, <On3>, <ContMvgUp3, ContMvgDwn6>, <On3>})
\item \texttt{TEventName}(\texttt{StopAtFl4}) =
\item Rules for identifying guards
\item ANY \quad \texttt{TParam}(\texttt{TClass}(\texttt{ TObject}(\texttt{StopAtFl4})))
\item \texttt{WHERE}
\item \texttt{TParamGuard}(\texttt{TParam}(\texttt{TClass}(\texttt{TObject}(\texttt{StopAtFl4}))))
\item \texttt{TCond}(\texttt{StopAtFl4, <On3>})
\item \texttt{TPrevStParam}(\texttt{StopAtFl4, <ContMvgUp3, ContMvgDwn6>})
\item \texttt{TCausesParam}(\texttt{<On3>})
\item \texttt{THEN}
\item \texttt{TNormalAct}(\texttt{StopAtFl4})
\item \texttt{< IF THasSimult(TSimult(StopAtFl4)) THEN>}
\item \texttt{TObject(TSimult(StopAtFl4)) state :=}
\item \texttt{TState(TSimult(StopAtFl4))}
\item \texttt{<END>}
\item END
\end{itemize}
Rule 9 : TPrevStParam( Segment, SeqPrev )

\[
T_{\text{PrevStParam}}(\text{Segment}, < >) = \text{""}
\]

\[
T_{\text{PrevStParam}}(\text{Segment}, \text{SeqPrev}) \rightarrow
<\text{IF} \text{ MultPrev}(\text{Segment}, \text{SeqPrev}) \text{ THEN } >
\]

\[
T_{\text{PrevStParam}}(\text{Segment}, \text{SeqPrev}) =
T_{\text{PrevStParamR}}(\text{Segment}, \text{Head} : \text{SegmSeqTail}) \rightarrow
T_{\text{Object}}(\text{Segment})_{\text{state}} (T_{\text{ParamLt}}(T_{\text{Param}}(T_{\text{Class}}(T_{\text{Object}}(\text{Segment})))) ) =
T_{\text{State}}(\text{Head}) \lor T_{\text{PrevStParam}}(\text{Segment}, \text{SegmSeqTail})
\]

\[
< \text{ELSE}>
T_{\text{Object}}(\text{Segment})_{\text{state}} ( (T_{\text{ParamLt}}(T_{\text{Param}}(T_{\text{Class}}(T_{\text{Object}}(\text{Segment})))) ) =
T_{\text{State}}(\text{Elem}(\text{SegmSeqTail}))
\]

\[
< \text{END}>
\]
TPrevStParam(StopAtFl4, < ContMvgUp3, ContMvgDwn6 >)
Example: Using rule 9 (Cont’)

1st recursion

\[ \text{TPrevStParam}(\text{Segment}, < >) = " " \]

\[ \text{TPrevStParam}(\text{StopAtFl4}, < \text{ContMvgUp3}, \text{ContMvgDwn6} >) \rightarrow \]
\[ < \text{IF MultPrev}(\text{StopAtFl4}, < \text{ContMvgUp3}, \text{ContMvgDwn6} >) \text{ THEN } > \]
\[ \text{TPrevStParam}(\text{StopAtFl4}, < \text{ContMvgUp3}, \text{ContMvgDwn6} >) = \]
\[ \text{TPrevStParamR}(\text{StopAtFl4}, \text{ContMvgUp3} : < \text{ContMvgDwn6} >) \rightarrow \]
\[ \text{TObject}(\text{StopAtFl4}) \text{ state (TParamLt(TParam(TClass(TObject(StopAtFl4)))))} = \]
\[ \text{TState}(\text{ContMvgUp3}) \lor \text{TPrevStParam}(\text{StopAtFl4}, < \text{ContMvgDwn6} >) \]

2nd recursion

\[ \text{TPrevStParam}(\text{StopAtFl4}, < \text{ContMvgDwn6} >) \rightarrow \]
\[ < \text{ELSE}> \]
\[ \text{TObject}(\text{StopAtFl4}) \text{ state (TParamLt(TParam(TClass(TObject(StopAtFl4))))) } = \]
\[ \text{TState}(\text{Elem(<ContMvgDwn6 >)}) \]
\[ < \text{END}> \]

Output: \( \text{liftstate}(f) = \text{ContMvgUp} \lor \text{liftstate}(f) = \text{ContMvgDwn} \)
Example : liftStopAtFl Abstract model’s event

\[\text{lif}t\text{StopAtFl} = \]
\[\text{ANY } f\]
\[\text{WHERE } f : \text{FLOOR} \quad /\ast \text{ rule 7}\ast/\]
\[f : \text{reqFl } f = \text{currentFl} \quad /\ast \text{ rule 8}\ast/\]
\[\text{liftstate}(f) = \text{ContMvgUp}\]
\[\text{liftstate}(f) = \text{ContMvgDown}\quad /\ast \text{ rule 9}\ast/\]
\[\text{floorsensorstate}(f) = \text{On} \quad /\ast \text{ rule 10}\ast/\]
\[\text{THEN}\]
\[\text{liftstate}(f) := \text{StopAtFl} \quad /\ast \text{ rule 11}\ast/\]
\[\text{directionlampstate} := \text{Deactivated} \quad /\ast \text{ rule 12}\ast/\]
\[\text{END}\]
Rules for creating refinement model’s events

Rules for identifying timing constraints as context

\[ TContext(CauseSegm, EffectSegm) \]

Rules for identifying guards

\[ TTransGeneral(Segment, SeqCon, SeqPrev, SeqCause) \rightarrow \]

...same rule for identifying abstract event’s name...

Rules for creating guards with parameters

\[ ANY \]

...same rule as defined in the abstract model...

\[ WHERE \]

...same rule as defined in the abstract model...

\[ TTimeCrtnt(SeqCause) \]

Rules for identifying actions

\[ TRefSubst(Segment) \]

...same rule as defined in the abstract model...

\[ END \]
Example: liftStopAtFl refinement model’s event

\[
\text{liftStopAtFl = } \\ \\
\text{REFINE liftStopAtFl} \ \\
\text{ANY f} \ \\
\text{WHERE} \ \\
\text{... same guard as defined in the abstract model...} \ \\
\begin{align*}
(g\text{clock} & \text{ – floorsensorOnTime}) \\
\text{LOWER\_LIMIT\_floorsensor} & \text{ &} \\
(g\text{clock} & \text{ – floorsensorOnTime}) \\
\text{UPPER\_LIMIT\_floorsensor} \\
\end{align*} \ \\
\text{THEN} \ \\
\text{...same substitutions as defined in the abstract model...} \ \\
liftStopAtFl\text{Time} := g\text{clock} \ \\
\text{END} \ \\
\]
Conclusions and Future work

Timing diagrams

- Notations
- BNF definitions

+ Input files

Translation rules → Event-B

UML-B

Class diagram
Statechart

u2B → Event-B
Questions ?