Building Event-B Interlocking Theories

Lessons Learned Using the Theory Plug-in

23/05/2016

Yoann Guyot    Renaud De Landtsheer    Christophe Ponsard

Centre d’Excellence en Technologies de l’Information et de la Communication
www.cetic.be
Who Am I?

Yet another formal methods research engineer…
...with a bit of interest in the event-b method.

Before @ Systerel
Coded for the SMT Solvers Plug-in of Rodin

Nowadays @ CETIC: Technology Transfert
Experimenting with the Theory Plug-in of Rodin
What is an Interlocking?

A Railway Signalling System
What is an Interlocking?

Request of routes → A Railway Signalling System
What is an Interlocking?

Request of routes

A Railway Signalling System

Controls railway network objects when requests are acceptable:
- sets points in expected positions
What is an Interlocking?

A Railway Signalling System

Controls railway network objects when requests are acceptable:
- sets points in expected positions
- ...
- opens/closes signals to give/deny access to the routes
What is an Interlocking?

A Railway Signalling System

- Controls railway network objects when requests are acceptable:
  - sets points in expected positions
  - ...
  - opens/closes signals to give/deny access to the routes

Trains move on the network without colliding with each others nor derailing...
Who Makes Interlockings?

Signalling engineers…

• possess the historical knowledge
• incrementally design new interlockings
• directly produce specific code
  based on generic programming rules
• validate by reviewing and testing
Who Makes Interlockings?

Signalling engineers...

- possess the historical knowledge
- incrementally design new interlockings
- directly produce specific code based on generic programming rules
- validate by reviewing and testing

What about mathematical proof of safety?

- are generally not fluent in formal methods
- are especially not event-b lovers
Outline

How to Bring These Worlds Together
*Use the Theory Plug-in of Rodin*¹

“Cover up those mathematical expressions which I cannot endure…”²

Defining Interlocking Theories

Reducing the Proof Effort

¹Modeling a Safe Interlocking Using the Event-B Theory Plug-in, M-T. Khuu, L. Voisin and F. Mejia
²(almost) Molière’s Tartuffe
Covering Mathematical Expressions
The Famous Train Example First Context

axm1: \( \text{blocks\_routes} \subseteq \text{BLOCKS} \leftrightarrow \text{ROUTES} \)

axm2: \( \text{dom(blocks\_routes)} = \text{BLOCKS} \)

axm3: \( \text{ran(blocks\_routes)} = \text{ROUTES} \)

axm4: \( \text{next} \in \text{ROUTES} \rightarrow (\text{BLOCKS} \not\subseteq \text{BLOCKS}) \)

axm5: \( \text{fst} \in \text{ROUTES} \rightarrow \text{BLOCKS} \)

axm6: \( \text{last} \in \text{ROUTES} \rightarrow \text{BLOCKS} \)

axm7: \( \text{fst} \subseteq \text{blocks\_routes} \)

axm8: \( \text{last} \subseteq \text{blocks\_routes} \)

axm9: \( \forall r \cdot r \in \text{ROUTES} \Rightarrow \text{fst}(r) \neq \text{last}(r) \)

axm10: \( \forall r \cdot r \in \text{ROUTES} \Rightarrow \text{next}(r) \in \text{blocks\_routes}^{-}[\{r\}] \setminus \{\text{last}(r)\} \not\subseteq \text{blocks\_routes}^{-}[\{r\}] \setminus \{\text{fst}(r)\} \)

axm11: \( \forall r \cdot r \in \text{ROUTES} \Rightarrow (\forall S \cdot S \subseteq \text{next}(r)[S] \Rightarrow S = \varnothing) \)

axm12: \( \forall r1, r2 \cdot r1 \in \text{ROUTES} \land r2 \in \text{ROUTES} \land r1 \neq r2 \)
\[ \Rightarrow \text{fst}(r1) \notin \text{blocks\_routes}^{-}[\{r2\}] \setminus \{\text{fst}(r2), \text{last}(r2)\} \]

axm13: \( \forall r1, r2 \cdot r1 \in \text{ROUTES} \land r2 \in \text{ROUTES} \land r1 \neq r2 \)
\[ \Rightarrow \text{last}(r1) \notin \text{blocks\_routes}^{-}[\{r2\}] \setminus \{\text{fst}(r2), \text{last}(r2)\} \]
The Famous Train Example First Context

axm1: blocks_routes ∈ BLOCKS ↔ ROUTES
axm2: dom(blocks_routes) = BLOCKS
axm3: ran(blocks_routes) = ROUTES

axm4: next ∈ ROUTES → (BLOCKS ⊖ BLOCKS)
axm5: fst ∈ ROUTES → BLOCKS
axm6: last ∈ ROUTES → BLOCKS
axm7: fst̸∈ blocks_routes
axm8: last̸∈ blocks_routes

axm9: ∀ r · r ∈ ROUTES ⇒ fst(r) ≠ last(r)
axm10: ∀ r · r ∈ ROUTES ⇒ next(r) ∈ blocks_routes−[{r}] \ {last(r)} ⊖ blocks_routes−[{r}] \ {fst(r)}
axm11: ∀ r · r ∈ ROUTES ⇒ (∀ S · S ⊆ next(r)[S] ⇒ S = ∅)
axm12: ∀ r1, r2 · r1 ∈ ROUTES ∧ r2 ∈ ROUTES ∧ r1 ≠ r2
        ⇒ fst(r1) ∉ blocks_routes−[{r2}] \ {fst(r2), last(r2)}
axm13: ∀ r1, r2 · r1 ∈ ROUTES ∧ r2 ∈ ROUTES ∧ r1 ≠ r2
        ⇒ last(r1) ∉ blocks_routes−[{r2}] \ {fst(r2), last(r2)}
What Is Our Goal?

Be as close as possible to the signaling engineers usage:

- Implicit objects definitions (blocks, routes, signals…)
- Implicit operators definitions (reserve, lock, open…)
- Rich DSL
- Implicit objects basic properties

Just model *this* interlocking.
The Famous Train Example First Context

Axiom 1: \( \text{blocks\_routes} \subseteq \text{BLOCKS} \leftrightarrow \text{ROUTES} \)

Axiom 2: \( \text{dom(blocks\_routes)} = \text{BLOCKS} \)

Axiom 3: \( \text{ran(blocks\_routes)} = \text{ROUTES} \)

Axiom 4: \( \text{next} \subseteq \text{ROUTES} \rightarrow (\text{BLOCKS} \not\subseteq \text{BLOCKS}) \)

Axiom 5: \( \text{fst} \subseteq \text{ROUTES} \rightarrow \text{BLOCKS} \)

Axiom 6: \( \text{last} \subseteq \text{ROUTES} \rightarrow \text{BLOCKS} \)

Axiom 7: \( \text{fst} \not\subseteq \text{blocks\_routes} \)

Axiom 8: \( \text{last} \not\subseteq \text{blocks\_routes} \)

Axiom 9: \( \forall r \cdot r \in \text{ROUTES} \Rightarrow \text{fst}(r) \neq \text{last}(r) \)

Axiom 10: \( \forall r \cdot r \in \text{ROUTES} \Rightarrow \text{next}(r) \subseteq \text{blocks\_routes} \backslash \{r\} \not\subseteq \text{blocks\_routes} \backslash \{r\} \)

Axiom 11: \( \forall r \cdot r \in \text{ROUTES} \Rightarrow (\forall S \cdot S \subseteq \text{next}(r)[S] \Rightarrow S = \emptyset) \)

Axiom 12: \( \forall r_1, r_2 \cdot r_1 \in \text{ROUTES} \land r_2 \in \text{ROUTES} \land r_1 \neq r_2 \Rightarrow \text{fst}(r_1) \not\subseteq \text{blocks\_routes} \backslash \{r_2\} \not\subseteq \text{blocks\_routes} \backslash \{r_2\} \)

Axiom 13: \( \forall r_1, r_2 \cdot r_1 \in \text{ROUTES} \land r_2 \in \text{ROUTES} \land r_1 \neq r_2 \Rightarrow \text{last}(r_1) \not\subseteq \text{blocks\_routes} \backslash \{r_2\} \not\subseteq \text{blocks\_routes} \backslash \{r_2\} \)
Using Our Theories

\textbf{axm1:} \quad ROUTES \subseteq routes(BLOCKS)

\textbf{axm2:} \quad wellFormedRoutes(ROUTES)
Using Our Theories

Concrete set of routes (event-b constant)

Definition of *routes* constructor is hidden in the theories

Concrete set of blocks (event-b constant)

**axm1:** \( \text{ ROUTES} \subseteq \text{ routes(BLOCKS)} \)

**axm2:** \( \text{ wellFormedRoutes(ROUTE)} \)

Additional properties of routes
- routes don’t start in the middle of others
- routes don’t end in the middle of others
The Famous Train Example First Machine

Without Theories

route_reservation:
ANY r WHERE
  r ∉ res_routes
  blocks_routes~[{r}] \cap res_blocks = ∅
THEN
  res_routes = res_routes \cup \{r\}
  resblk_resrt = resblk_resrt \cup \text{blocks_routes} \parr \{r\}
  res_blocks = res_blocks \cup \text{blocks_routes}~[{\{r\}]}
END

Using Our Theories

route_reservation:
ANY r WHERE
  r \in ROUTES
  ~ isReserved(r, res_routes)
  noReservedBlocks(r, res_routes)
THEN
  res_routes = res_routes \cup \text{completeRes}(r)
END
Defining Interlocking Theories
Interlocking Theories Dependencies

Interlocking Theories
- Trains
  - Route Reservation
    - Points
      - Routes

Intermediate Theories
- SubChains
  - SubSeqs
    - MoreSeqs
      - Seq

Basic Theories (official package)
Chains

Railway network is made of chains of blocks.
\{ c \mid c \in \text{seq}(S) \land c^\sim \in T \otimes \mathbb{Z} \}

Equivalent to iseq guarantees that there is no cycle

Operators:
- chains (constructor)
- chainSize
- emptyChain
- chainFst / Last / Tail
- chainPrev / Next
- chainPrepend / Append
- ...

Theorems:
- chainIsFinite
- chainIsMonotonic
- nextIsInRange
- chainTailIsChain
- chainPrependIsChain
- ...

www.cetic.be
Routes

Routes are chains of at least two blocks.
\{ c \mid c \subseteq \text{chains(blocks)} \land \text{card}(c) > 1 \} 

Operators:
- routes (constructor)
- blcks
- routeLength
- routeFst / Last / Tail
- routePrev / Next
- starts / endsInTheMiddle
- wellFormedRoutes
- ...

Theorems:
- routeIsFinite
- routeIsMonotonic
- routeIsFunc
- routeDomain
- routeIsChain
- ...
Routes (How To Use It ?)

Routes are chains of at least two blocks.
{ c | c ∈ chains(blocks) ∧ card(c) > 1 }

• r is a route
• b is a block of r
• first block of r
• block after b on r
• r1 starts in the middle of r2
• R is a set of routes
• no route in R ends in the middle of another

r ∈ routes(BLOCKS)
b ∈ blocks(r)
routeFst(r)
routeNext({b}, r)
startsInTheMiddle(r1, r2)
R ⊆ routes(BLOCKS)
routesDontEndInTheMiddle(R)
Subchains

Routes reservations and trains are subchains of routes.
\{ \text{sub} \mid \text{sub} \in \text{chains}(T) \land \text{subChain(sub, c)} \}\n
Operators:
- subChain (constructor)
- subChains (constructor)
- frontSubChains (constructor)
- backSubChains (constructor)

Theorems:
- subChainIsChain
- frontSubChainIsSubChain
- backSubChainIsSubChain
- chainTailIsABackSubChain
- chainTail
- ...
Subchains (How To Use It ?)

Routes reservations and trains (routes occupation) are subchains of routes.

\{
  \text{sub} \mid \text{sub} \subseteq \text{chains}(T) \land \text{subChain}(\text{sub}, c)
\}

- \text{r1} is a subchain of \text{r2}
- set of subchains of \text{r}
- set of “front subchains” of \text{r}
- set of “back subchains” of \text{r}
Routes Reservations

A theory for maintaining the set of routes reservations.
\{ r \mapsto b \mid r \in \text{routes}(\text{blocks}) \land b \in \text{backSubChains}(r) \}
Routes Reservations (How To Use It ?)

A theory for maintaining the set of routes reservations.
\{ r \mapsto b \mid r \in \text{routes(blocks)} \land b \in \text{backSubChains}(r) \}

- the set of route reservations on the track
- only one reservation can be made for each route
- a block cannot be reserved twice

res \in \text{validRouteRes(BLOCKS, ROUTES)}
onlyOneResByRoute(res)
compatibleRoutesOnly(res)
Routes Reservations (How To Use It ?)

A theory for maintaining the set of routes reservations.
\[ \{ r \mapsto b \mid r \in \text{routes}(\text{blocks}) \land b \in \text{backSubChains}(r) \} \]

\[ r = \{ b_1; b_2; \ldots; b_n \} \]

Events
- Initialisation
- Reserve \( r \)
- Enter route; Front moves...
- Back move
- Back moves...
- Free \( r \)

Route Reservations
- \( \{ \} \)
- \( \{ r \mapsto \{ b_1; b_2; \ldots; b_n \} \} \)
- \( \{ r \mapsto \{ b_1; b_2; \ldots; b_n \} \}
  \quad \{ r \mapsto \{ b_2; \ldots; b_n \} \}
  \quad \{ r \mapsto \{ \} \}\)
  \quad \{ \} \)
Reducing the Proof Effort?

Conclusion
Current Status

Theorems defined = proof factorized.
Manual proof is easier, sometimes even trivial.

But only 40% POs automatically discharged in our theories and models.

Defining theories does not naturally simplify the proof.
The Right Theorems… (future work)

How to define the right theorems?

Suggestion:
1. define more theorems… a lot of them!
2. generalization of the theorems
3. simplification of the theorems
Proof Strategies? (future work)

Theorems not automatically applied: use strategies?
How to define them?
Discussion (1/2)

- Infix predicates?
  - `startsWithInTheMiddle(r1, r2) => r1 startsInTheMiddle r2`

- Type inference?
  - `r ∈ ROUTES`
    - `isReserved(r, res_routes)`

- Local theories? Using symbols that are local for a given project.
  - `~ isReserved(r, res_routes)`

- Assignment operators?
  - `res_routes := res_routes ∪ completeRes(r)`
Discussion (2/2)

• Theory instanciation?
  • ex: Instantiate the Routes theory with the constant BLOCKS

• What about a Rodin plug-in for generating Domain Specific Platforms (DSP)?
  1. Define the DSL using the Theory plug-in
  2. Validate the DSL with signaling engineers
  3. Generate the DSP
  4. Let signaling engineers model their system
Thanks

Yoann Guyot
Senior Research Engineer
yoann.guyot@cetic.be

twitter.com/@CETIC
twitter.com/@CETIC_be
linkedin.com/company/cetic
info@cetic.be
+32 71 490 700