Generic Instantiation and Tool Support

Thai Son Hoang

Institute of Information Security, ETH Zurich, Switzerland

Rodin Workshop 2013, Turku, Finland
11th June 2013

(joint work between ETH Zurich and Hitachi Ltd.)
Abrial and Hallerstede (2007)

... modeling a large and complex computer system results in a large and complex model. ... proofs will be more and more difficult to perform as models become inevitably larger and larger.

... We present three techniques, refinement, decomposition, and instantiation, that we consider indispensable for modeling large and complex systems.
Refinement, Decomposition, and Instantiation

Current Status

- **Refinement**: An integral part of Event-B
- **Decomposition**: Shared-event / shared-variable decomposition
Refinement, Decomposition, and Instantiation

Current Status

- **Refinement**: An integral part of Event-B
- **Decomposition**: Shared-event / shared-variable decomposition
- **Generic instantiation**:
  - R. Silva and M. Butler: Supporting Reuse of Event-B Developments through Generic Instantiation. (ICFEM 2009)
  - Ulyana Tikhonova at. al. (Rodin Workshop 2013) (this morning)
The development is parameterised by $S$ and $c$.

Reuse model: Instantiating to $S = E(T)$ and $c = F(T, d)$.

$B(T, d) \Rightarrow A(E(T), F(T, d))$.
Generic Instantiation

(Generic) Sets $S$, constants $c$
Axioms $A(S, c)$

- The development is parameterised by $S$ and $c$
Generic Instantiation

- The development is parameterised by $S$ and $c$
- Reuse model: Instantiating to $S = E(T)$ and $c = F(T, d)$

$$B(T, d) \Rightarrow A(E(T), F(T, d))$$
Instantiating Sets and Constants. An Example

Generic context

**sets**: MESSAGE

**constants**: maxsize

**axioms**:
- finite(MESSAGE)
- maxsize \(\in\) \(\mathbb{N}_1\)

Specific context

**sets**: ID, INFO

**axioms**:
- finite(ID)
- finite(INFO)

- Instantiation: MESSAGE = ID \times INFO, maxsize = 3

- To be proved: finite(ID) \land finite(INFO) \Rightarrow finite(ID \times INFO) \land 3 \in \mathbb{N}_1
Instantiating Sets and Constants. An Example

Generic context

**sets**: MESSAGE

**constants**: maxsize

**axioms**:
finite(MESSAGE)
maxsize ∈ N1

Specific context

**sets**: ID, INFO

**axioms**:
finite(ID)
finite(INFO)
Instantiating Sets and Constants. An Example

Generic context

- **sets**: `MESSAGE`
- **constants**: `maxsize`
- **axioms**: 
  - `finite(MESSAGE)`
  - `maxsize ∈ ℕ1`

Specific context

- **sets**: `ID, INFO`
- **axioms**: 
  - `finite(ID)`
  - `finite(INFO)`

- Instantiation: `MESSAGE = ID × INFO, maxsize = 3`

- To be proved:
  
  `finite(ID) ∧ finite(INFO) ⇒ finite(ID × INFO) ∧ 3 ∈ ℕ1`
A Generic Instantiation Tool
A Generic Instantiation Tool

\[ M_0(S, c) \]

\[ M_n(S, c) \]

\[ \text{refines} \]

\[ \text{sees} \]

\[ A(S, c) \]

\[ C_i \]

\[ B(T, d) \]

\[ D_j \]

\[ \text{GenInst} = E(T) \]

\[ c = F(T, d) \]

\[ M_0(E(T), F(T, d)) \]

\[ M_n(E(T), F(T, d)) \]

\[ \text{refines} \]

\[ \text{theorems} \]

\[ \text{extends} \]

\[ \text{generates} \]
A Generic Instantiation Tool

\[ M_0(S, c) \]
\[ M_n(S, c) \]
\[ S = E(T) \]
\[ c = F(T, d) \]

\[ \text{refines} \]

\[ \text{sees} \]

\[ \text{GenInst} \]

\[ \text{extends} \]

\[ D_j \]
A Generic Instantiation Tool

\[ S = E(T) \]
\[ c = F(T, d) \]

\[ M_0(S, c) \]
\[ M_n(S, c) \]

\[ C_i \]
\[ A(S, c) \]

\[ M_0(E(T), F(T, d)) \]
\[ M_n(E(T), F(T, d)) \]

\[ \text{refines} \]
\[ \text{sees} \]

\[ \text{generates} \]
\[ \text{extends} \]

\[ \text{GenInstCtx theorems} \]
\[ A(E(T), F(T, d)) \]
The Usefulness of Generic Instantiation

The similarity between refinement and generic instantiation
The Usefulness of Generic Instantiation

The similarity between refinement and generic instantiation

The Usefulness of Refinement

The ability to perform abstraction with state variables.
The Usefulness of Generic Instantiation
The similarity between refinement and generic instantiation

The Usefulness of Refinement
The ability to perform abstraction with state variables.

The Usefulness of Generic Instantiation
The ability to perform abstraction with sets and constants
Train Control Example (1)

Typical modelling style using variables

\[
\begin{align*}
\text{trains} & \subseteq \text{TRAIN\_ID} \\
\text{head} & \in \text{trains} \rightarrow \text{SECTION} \\
\text{rear} & \in \text{trains} \rightarrow \text{SECTION} \\
\text{area} & \in \text{trains} \rightarrow \mathbb{P}(\text{SECTION}) \\
\text{connection} & \in \text{trains} \rightarrow (\text{SECTION} \leftrightarrow \text{SECTION})
\end{align*}
\]
Train Control Example (2)

- Trains can be represented by some abstract data type (e.g., sequence of sections `SEQUENCE`).

- Operations with sequences:
  - Extend head: `extend ∈ SEQUENCE × SECTION ↦ SEQUENCE`
  - Remove rear: `front ∈ SEQUENCE ↦ SEQUENCE`
  - Disjointness: `s_1 ↦ s_2 ∈ disjoint`
  - Sub-sequence: `s_1 ↦ s_2 ∈ subset`

- Properties of sequences

\[ \forall s_1, s_2, s_3 \cdot \]
\[ s_1 ↦ s_2 ∈ subset \land s_2 ↦ s_3 ∈ disjoint \Rightarrow s_1 ↦ s_3 ∈ disjoint \]

\[ \forall s_1, s_2 \cdot s_1 ↦ s_2 ∈ disjoint \Rightarrow s_2 ↦ s_1 ∈ disjoint \]
Summary

- Plug-in on sourceforge:
  http://sourceforge.net/projects/gen-inst/

- Instantiation enables abstraction with models’ parameters.

- Context instantiation as a part of the development?