Constraint-Aware Model Merging

Alessandro Rossini\textsuperscript{1}, Adrian Rutle\textsuperscript{2}, Florian Mantz\textsuperscript{2}, Yngve Lamo\textsuperscript{2}, Uwe Wolter\textsuperscript{1}

\textsuperscript{1}Department of Informatics, University of Bergen, Norway
\textsuperscript{2}Faculty of Engineering, Bergen University College, Norway

11 November 2010
Nordic Workshop on Programming Theory, Turku, Finland
Outline

Introduction and Motivation

Diagram Predicate Framework

Constraint-Aware Model Merging
Model-driven engineering (MDE)

- *Model* in software engineering
  - abstract representation of a software system
  - typically graph-based structure, e.g. UML
Model-driven engineering (MDE)

- **Model** in software engineering
  - abstract representation of a software system
  - typically graph-based structure, e.g. UML
- Model-driven engineering (MDE)
  - models as primary artefacts of software development
  - generation of systems by model transformations
Model-driven engineering (MDE)

- **Model** in software engineering
  - abstract representation of a software system
  - typically graph-based structure, e.g. UML

- Model-driven engineering (MDE)
  - models as primary artefacts of software development
  - generation of systems by model transformations

- Complex evolution of models
  - need for techniques and tools to support version control
Optimistic version control

- A local copy of artefacts for each developer
  - independent and parallel modifications
- Merging of modifications
  - centralised (copy-modify-merge)
  - distributed
Three-way model merging

- Merging two versions of a model relying on their common ancestor

\[ V_1 \rightarrow V_{1}' \rightarrow V_2 \rightarrow V_{1}'' \rightarrow V_1 \]
Three-way model merging

- Merging two versions of a model relying on their common ancestor

\[ V_1 \quad \rightarrow \quad V_2 \quad \leftarrow \quad V_1' \quad \rightarrow \quad V_1'' \]

- Detection of conflicts
  - possibly resolution
Challenges

- Mainstream version control systems, e.g. Subversion
  - target text-based artefacts
  - not suitable for graph-based structures
- Prototype model version control systems, e.g. AMOR
  - lack of formal underpinning
- State-of-the-art of research
  - constraint on model elements not considered
Proposed solution

- Diagram Predicate Framework (DPF)
- Formal approach to MDE based on category theory
  - (meta)modelling
  - model transformation
  - model management
Outline

Introduction and Motivation

Diagram Predicate Framework

Constraint-Aware Model Merging
Sample Object-Oriented Modelling Hierarchy

- Modelling language: graph + ...
Sample Object-Oriented Modelling Hierarchy

- Modelling language: graph + signature
Sample Object-Oriented Modelling Hierarchy

- Model (specification): graph + constraints
- Nodes interpreted as sets
- Arrows interpreted as multi-valued functions

<table>
<thead>
<tr>
<th>p</th>
<th>α_S2(p)</th>
<th>Proposed vis.</th>
<th>Intended semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>![mult(m,n)]</td>
<td>![a ↦ 2]</td>
<td>![f ↦ [m..n]]</td>
<td>∀x ∈ X : m ≤</td>
</tr>
<tr>
<td>![surjective]</td>
<td>![a ↦ 2]</td>
<td>![f ↦ surj]</td>
<td>f(X) = Y</td>
</tr>
<tr>
<td>![inverse]</td>
<td>![a ↦ 2]</td>
<td>![f ↦ inv]</td>
<td>∀x ∈ X, ∀y ∈ Y : y ∈ f(x) iff x ∈ g(y)</td>
</tr>
<tr>
<td>![image-inclusion]</td>
<td>![a ↦ 2]</td>
<td>![f ↦ ⊑]</td>
<td>∀x ∈ X : f(x) ⊆ g(x)</td>
</tr>
<tr>
<td>![composition]</td>
<td>![a ↦ 2]</td>
<td>![f ↦ comp]</td>
<td>∀x ∈ X : h(x) = {g(y)</td>
</tr>
</tbody>
</table>

- Diagram Predicate Framework
- Constraint-Aware Model Merging

---

- Model (specification): graph + constraints
- Nodes interpreted as sets
- Arrows interpreted as multi-valued functions
• Req 1: “an employee must work for at least one department”
• Req 2: “a department may have none or many employees”
• Req 3: “a project may involve none or many employees”
• Req 4: “a project must be controlled by at least one department”
● Req 5: “an employee involved in a project must work in the controlling department”
Sample Object-Oriented Modelling Hierarchy

\[ \frac{\text{Data Type}}{\text{Class}} \quad \frac{\text{Reference}}{\text{Employee}} \quad \frac{\text{Department}}{\text{Project}} \]

\[ \text{Proposed vis.} \quad \text{Intended semantics} \]

<table>
<thead>
<tr>
<th>( p )</th>
<th>( \alpha^{22}(p) )</th>
<th>Proposed vis.</th>
<th>Intended semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>([\text{mult}(m,n)])</td>
<td>1 ( \xrightarrow{a} ) 2</td>
<td>( X \xrightarrow{f} [m..n] Y )</td>
<td>( \forall x : X \leq</td>
</tr>
<tr>
<td>([\text{surjective}])</td>
<td>1 ( \xrightarrow{a} ) 2</td>
<td>( X \xrightarrow{f} [\text{surj}] Y )</td>
<td>( f(X) = Y )</td>
</tr>
<tr>
<td>([\text{inverse}])</td>
<td>1 ( \xrightarrow{a} ) 2</td>
<td>( X \xrightarrow{f} [\text{inv}] Y )</td>
<td>( \forall x \in X, \forall y \in Y : y \in f(x) \iff x \in g(y) )</td>
</tr>
<tr>
<td>([\text{image-inclusion}])</td>
<td>1 ( \xrightarrow{a} ) 2</td>
<td>( X \xrightarrow{f} [\text{⊑}] Y )</td>
<td>( \forall x \in X : f(x) \subseteq g(x) )</td>
</tr>
<tr>
<td>([\text{composition}])</td>
<td>1 ( \xrightarrow{a} ) 2</td>
<td>( X \xrightarrow{f} [\text{comp}] Y )</td>
<td>( \forall x \in X : h(x) = {g(y) \mid y \in f(x)} )</td>
</tr>
</tbody>
</table>

- **Instance**
Sample Object-Oriented Modelling Hierarchy

- Invalid instance
Specification entailment

\[ \mathcal{L} \models \mathcal{R} \]

\[ \alpha([\text{surjective}]) \alpha([\text{inverse}]) \models \alpha([\text{mult}(1,\infty)]) \]

\[
\forall x \in X \text{ and } \forall y \in Y : x \in g(y) \text{ iff } y \in f(x) \quad f, g \text{ inverse}
\]

\[
\forall x \in X : \bigcup \{g(y)\} = X \quad g \text{ surjective}
\]

\[
\Rightarrow \forall x \in X \exists y \in Y : x \in g(y)
\]

\[
\Rightarrow \forall x \in X \exists y \in Y : y \in f(x)
\]

\[
\Rightarrow \forall x \in X : |f(x)| \geq 1 \quad f \text{ total}
\]
Embedding of specification entailment

\[ \mathcal{S} \vdash \mathcal{S} \cup \phi(\mathcal{R}) \]
\[ \alpha([\text{surjective}]) \alpha([\text{inverse}]) \vdash \alpha([\text{mult}(1,\infty)]) \]
Outline

Introduction and Motivation

Diagram Predicate Framework

Constraint-Aware Model Merging
Example 1: multiplicity
Example 1: multiplicity

Repository

Alice’s copy

Bob’s copy

Timeline

Repository

Alice

Bob
Example 1: multiplicity
Example 1: multiplicity

Repository

Alice’s copy

Bob’s copy

Timeline

Repository

Alice

Bob
Example 1: multiplicity

**Repository**

- Alice's copy:
  - $X \rightarrow Y [1..3]$
- Bob's copy:
  - $X \rightarrow Y [2..4]$

**Timeline**

- Repository
  - $V_1$
  - $V_2$
- Alice
  - $A_1$
- Bob
  - $B_1$
Example 1: multiplicity

Repository

Alice’s copy

Bob’s copy

Timeline
Example 1: multiplicity

Repository

Alice’s copy

Bob’s copy

Timeline

Repository

Alice

Bob

Example 1: multiplicity
Example 1: multiplicity

- Model merging
  - push out [Rossini et al., 2010]
- Conflict detection
Example 1: multiplicity

- Model merging
  - push out [Rossini et al., 2010]
- Conflict detection
- Conflict resolution
Example 2: multiplicity and surjectivity

Repository  Alice’s copy  Bob’s copy

\[ \begin{align*}
X & \xrightarrow{f} Y \\
\xrightarrow{g} & 
\end{align*} \]

Timeline

\[ \begin{align*}
\mathcal{V}_1 & \\
\mathcal{A}_1 & \quad \mathcal{B}_1 \\
\mathcal{V}_2 & \\
\mathcal{A}_1 & \quad \mathcal{B}_1 \\
\end{align*} \]
Example 2: multiplicity and surjectivity
Example 2: multiplicity and surjectivity
Example 2: multiplicity and surjectivity

Repository

Alice’s copy

Bob’s copy

Timeline

Example 2: multiplicity and surjectivity

Repository

Alice’s copy

Bob’s copy

Timeline
Example 2: multiplicity and surjectivity

Repository

Alice’s copy

Bob’s copy

Timeline

Repository

Alice

Bob
Example 2: multiplicity and surjectivity
Example 2: multiplicity and surjectivity

- Model merging
- Conflict detection
Example 2: multiplicity and surjectivity

- Model merging
- Conflict detection
- Conflict resolution
Summary

• Extension of DPF
  • specification entailments

• Constraint-aware model merging
  • detection of semantic conflicts
  • resolution of conflicts
Future work

- Fully-fledged logic for predicates
  - Deduction calculus
- Prototype tool
- Real-size case study
Related work

- [Westfechtel, 2010] Three-way merging of Ecore models based on set theory and predicate logic
- [Taentzer et al., 2010] Merging of typed attributed graphs based on graph transformations and category theory
- [da Silva et al., 2010] Automatic generation of repair plans for a given inconsistent model
Related work

- [Westfechtel, 2010] Three-way merging of Ecore models based on set theory and predicate logic
- [Taentzer et al., 2010] Merging of typed attributed graphs based on graph transformations and category theory
- [da Silva et al., 2010] Automatic generation of repair plans for a given inconsistent model
- Constraints on model elements not considered
Thank you!

Questions?

For more information and list of publications visit:
http://dpf.hib.no/
Resolution pattern

\[(\text{mult}(1, 3), \delta_3) \lor (\text{mult}(2, 4), \delta_3) \equiv (\text{mult}(1, 4), \delta_3)\]
Resolution pattern

\[
([\text{mult}(m_1, n_1)], \delta) \lor ([\text{mult}(m_2, n_2)], \delta)
\]

\[
\equiv \begin{cases} 
([\text{mult}(m_1, n_1)], \delta) & \text{if } m_1 \leq m_2 \leq n_2 \leq n_1 \\
([\text{mult}(m_2, n_2)], \delta) & \text{if } m_2 \leq m_1 \leq n_1 \leq n_2 \\
([\text{mult}(m_1, n_2)], \delta) & \text{if } m_1 \leq m_2 \leq n_1 \leq n_2 \\
([\text{mult}(m_2, n_1)], \delta) & \text{if } m_2 \leq m_1 \leq n_2 \leq n_1 
\end{cases}
\]