Enriching Megamodel Management with Collection-Based Operators

RICK SALAY, SAHAR KOKALY, ALESSIO DI SANDRO, MARSHA CHECHIK

MODELS 2015

OCT. 1, 2015
A Megamodel is not just a really big model!
A Megamodel represents models and their relationships at a high level of abstraction to facilitate model management.

Context: Large software projects ⇒ proliferation of model artifacts
- Need to manage this “accidental complexity”
Motivating Example

Auto Inc. megamodel

Problem: Dev dept. decides a construct in some models is undesirable

Fix procedure

1. Identify (bad) models containing the construct
2. Refactor bad models to remove the construct
3. Merge the modified models with remaining good models for analysis

These steps require working with a megamodel as a collection
Collection operators in programming

Filter\(<P>(L)\)
- Extract list of elements of L that satisfies prop P
- E.g., Filter<Odd>([1,2,3,4])=[1,3]

Map\(<F>(L)\)
- Apply function F to elements of list L
- E.g., Map<Double>([1,2,3,4])=[2,4,6,8]

Reduce\([F](L)\)
- Fold elements of L into a single value using binary function F
- E.g., Reduce<Plus>([1,2,3,4])=10
Collection operators for megamodels

Filter, Map and Reduce (FMR) can be used with megamodels

Example fix procedure with collection operators

1. Identify (bad) models containing construct
   ◦ use Filter with a property to identify the construct

2. Refactor bad models to remove the construct
   ◦ use Map with refactoring transformation

3. Merge the modified models with remaining good models for analysis
   ◦ use Reduce with a merge transformation
FMR for megamodels

Key Challenges

1. Need FMR to correctly manipulate entire graphs of related models rather than just sets of models.
   - Graph edges (i.e., relationships) have content

2. Map and Reduce must work with transformations
   - Transformations accept graphs of models and rels as input and output

Contributions:

- Adapt FMR to megamodels addressing challenges
- Implement FMR in model management tool (MMINT)
Operator: Filter

Filter[P](X)

Extracts models/relationships satisfying property P from megamodel X.

Assumptions:
- Property language for P is given
- P can be either a model property or relationship property

Behaviour details:
- Model property: all relationships between extracted models kept
- Relationship property: all endpoint models kept
Example
Operator: Map

\text{Map}[F](\{X\})

Applies transformation F to each matching site in the input megamodels.

Assumptions:

- Input is a set of megamodels that conform to input signature of F
- Output is set of megamodels that conform to output signature of F

Behaviour details:

- If F is commutative then isomorphic sites for F are omitted
- Each site is split across input megamodels
Example: Applying Map to CDMatch

A:CD
B:CD
C:CD

:map[CDMatch]

R1:CDrel
R2:CDrel
R3:CDrel
Operator: Reduce

\textbf{Reduce}[F](X)

Applies a transformation F to reduce the content of a megamodel X.

\textbf{Assumptions:}
\begin{itemize}
  \item Relationship composition operators exist.
  \item F is commutative and associative
    \begin{itemize}
      \item To ensure \textit{confluence}.
    \end{itemize}
  \item F must be strictly reducing in output types
    \begin{itemize}
      \item To ensure \textit{termination}.
    \end{itemize}
\end{itemize}

\textbf{Behaviour details:}
\begin{itemize}
  \item Applies F to arbitrary matching sites until it can no longer be applied
Example: Applying Reduce to CDMerge

[Diagram showing the application of Reduce to CDMerge]
Complexity Analysis

Worst case complexity of the operators:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>map(<a href="%7BX%7D">F</a>)</td>
<td>(O(n^k \times C_F(m)))</td>
</tr>
<tr>
<td>reduce(<a href="X">F</a>)</td>
<td>(O(n^2 \times C_F(m)))</td>
</tr>
<tr>
<td>filter(<a href="X">P</a>)</td>
<td>(O(n^q \times C_P(m)))</td>
</tr>
</tbody>
</table>

Operators scale reasonably for certain classes of application scenarios.

Complexity is no worse than quadratic (modulo the transformation/property complexity) in the size of the input megamodel:

- when \textbf{map} is applied to a transformation with two or less input models
- in all cases for \textbf{reduce}
- when \textbf{filter} is applied to a model property or a binary relationship property
Problem: Dev dept. decides CD’s with public attributes are undesirable

Fix procedure

1. Identify CD’s containing public attributes
2. Refactor these CD’s using AddGetter to remove public attributes
3. Merge the refactored CD’s with remaining CD’s for analysis
Using FMR

Fix procedure

1. Identify CD’s containing public attributes (filter with prop PubAtt)
   a. Identify remaining CD’s by filtering with complement prop NoPubAtt
2. Refactor these CD’s by mapping AddGetter to remove public attributes
   a. Union refactored CD’s with remaining CD’s
3. Merge the refactored CD’s with remaining CD’s for analysis
Scenario: CD to ER

Transform a megamodel or related CD’s to a megamodel of related ER’s
Tooling

Implemented FMR for megamodels using MMINT (Model Management INTeractive) workbench

Preliminary scalability study shows implementation is consistent with complexity analysis.

Demo was yesterday! (But re-demos can be arranged..)
Related Work
Summary of Contributions

1. Adapted Filter-Map-Reduce collection operators to megamodels addressing challenges
2. Analyzed the complexity of the operators
3. Implemented operators and scenarios in MMINT
Future Work

- Integrate FMR into an MDE workflow language
- Extend approach to take into account hierarchical megamodel structure
- Improve scalability
Questions