Compositional development framework for bidirectional model transformations based on structural recursion on graphs

Soichiro Hidaka*
National Institute of Informatics

BIRS-BX 2013
December 4, 2013

Bidirectional Graph Transformation
BX on Graphs

Models: Graphs
Bidirectional Model Transformation: Bidirectional Graph Transformation
Can We Design a Language for BiG?
Example of BX on Graphs

- Replace ‘a’ by ‘d’ and removes ‘c’
WPutGet (Weak PutGet)
Towards a General Solution …

- How to deal with termination of graph transformation?
  ➔ Structural Recursion and its bulk semantics

- How to deal with equality of two graphs?
  ➔ Bisimulation (graphs as regular trees)

- How to correctly reflect changes on the view to the source?
  ➔ Traceability based on Bulk Semantics
GRoundTram: A General Functional Framework

• It is compositional (functional)
  – Based on the existing graph query language UnQL
• It is well-behaved
  – Built upon bidirectional UnCAL: a graph algebra with clear bidirectional semantics
• It is an integrated development environment
  – Graph editor, graph validation, graph transformation checking, visualizations of bidirectional behavior
Overview of GRoundTram

Our Approach: Bidirectionalization

UnCAL graph algebra
structural recursion
[Buneman et al., VLDBJ00]

Bidirectionalization
• adding trace information
• narrowing

source graph
Bidirectional Interpreter
Operation based updating

target graph

Architecture of GRoundTram

Model Transformation in UnQL+ (Compositional and Functional)

UnQL+ to UnCAL Translation [SAC]

Source/Target Models

Bidirectional Graph Contraction [ASE, PI]

Frontend of Bidirectional

UnCAL Graph Algebra (Structural Recursion)

UnCAL

UnCAL Graphs

Backend Engine for Bidirectional UnCAL

Bidirectional Interpreter [ICFP]

Optimizer [LOPSTR]

Transformation Verifier [PPDP]

Graph Verifier [SAC]

Update Checker [JSSST]

Application [ICSE, BX]
UnCAL (Unstructured CALculus) [Buneman et al. VLDB J. 2000] features:

1. Regular tree transformation
2. Structural Recursion
3. Termination and finiteness-preserving
UnCAL-1. Regular Tree Transformation

- \{\text{finite graphs}\}/\text{bisimilarity} \cong \{\text{regular trees}\}
  
  \rightarrow \text{Graph transformations as regular tree transformations}

- All UnCAL queries are well-defined w.r.t. bisimilarity

\[ g_1 \sim g_2 \Rightarrow f(g_1) \sim f(g_2) \]
Graph Model

$G = (V, B, I)$ where

$V = \{1, 2, 3, 4\}$,
$B = \{1 \mapsto \{E(d, 2), E(a, 4)\}$,
$2 \mapsto \{E(c, 3)\}$,
$3 \mapsto \{E(d, 2)\}$,
$4 \mapsto \{E(b, 3), O(&y)\}$,
$I = \{(&l \mapsto 1)\}$

**Graph Equivalence**

bisimulation

$= \text{strong bisim.} + \varepsilon$-elimination

Graph Model

Graph Equivalence
Graph Constructors

\[
\emptyset \quad \& \\
\{ \} \quad \&
\]

\[
\begin{align*}
\& x_1 \& x_2 \\
& x_1 \& x_2
\end{align*}
\]

\[
\emptyset \quad \&
\]

\[
\begin{align*}
\& x \& \& x_1 \& x_2 \\
& y_1 \& y_2
\end{align*}
\]

\[
\begin{align*}
\& x \& \& x_1 \& x_2 \\
& y_1 \& y_2
\end{align*}
\]

\[
\begin{align*}
\& x \& \& x_1 \& x_2 \\
& y_1 \& y_2
\end{align*}
\]

\[
\begin{align*}
\& x \& \& x_1 \& x_2 \\
& y_1 \& y_2
\end{align*}
\]

\[
\begin{align*}
\& x \& \& x_1 \& x_2 \\
& y_1 \& y_2
\end{align*}
\]

\[
\begin{align*}
\& x \& \& x_1 \& x_2 \\
& y_1 \& y_2
\end{align*}
\]

\[
\begin{align*}
\& x \& \& x_1 \& x_2 \\
& y_1 \& y_2
\end{align*}
\]

\[
\begin{align*}
\& x \& \& x_1 \& x_2 \\
& y_1 \& y_2
\end{align*}
\]

\[
\begin{align*}
\& x \& \& x_1 \& x_2 \\
& y_1 \& y_2
\end{align*}
\]

\[
\begin{align*}
\& x \& \& x_1 \& x_2 \\
& y_1 \& y_2
\end{align*}
\]

\[
\begin{align*}
\& x \& \& x_1 \& x_2 \\
& y_1 \& y_2
\end{align*}
\]

\[
\begin{align*}
\& x \& \& x_1 \& x_2 \\
& y_1 \& y_2
\end{align*}
\]

\[
\begin{align*}
\& x \& \& x_1 \& x_2 \\
& y_1 \& y_2
\end{align*}
\]

\[
\begin{align*}
\& x \& \& x_1 \& x_2 \\
& y_1 \& y_2
\end{align*}
\]

\[
\begin{align*}
\& x \& \& x_1 \& x_2 \\
& y_1 \& y_2
\end{align*}
\]

\[
\begin{align*}
\& x \& \& x_1 \& x_2 \\
& y_1 \& y_2
\end{align*}
\]

\[
\begin{align*}
\& x \& \& x_1 \& x_2 \\
& y_1 \& y_2
\end{align*}
\]

\[
\begin{align*}
\& x \& \& x_1 \& x_2 \\
& y_1 \& y_2
\end{align*}
\]
UnCAL-2: Structural Recursion

- Functional style programming /
  For example:
  For $f = \text{src}(e)$

  \[
  f \{\} = \{\}
  \]

  \[
  f \{l:G\} = e(l,G)@ f \ G
  \]

  \[
  f (G_1 \cup G_2) = (f \ G_1) \cup (f \ G_2)
  \]
UnCAL-3: Termination and Finiteness-Preserving

- Bulk semantics

\[ a2d_{xc} = \text{srec}(\lambda(l,g). \text{ if } l=a \text{ then } \{d:\&\} \text{ else if } l=c \text{ then } \{\varepsilon:\&\} \text{ else } \{l:\&\}) \]
Example: A Customer-Order Graph
Compositional User Language

• SQL-like graph query language

```sql
select {tables : $table} where $persistentClass in
  (* select classes *)
  (select $class where
    {Association.(src|dest).Class : $class} in $db,
    {is_persistent : {Boolean : true}} in $class),
$table in
  (* replace Attribute *)
  (replace attrs -> $g
  by (select {Column : $a} where
    {attrs.Attribute : $a} in $persistentClass)
  in $persistentClass))
```
GUI of the GRoundTRam Implementation

- Fwd/bwd transformation by 1 click
- Invalid modification on the target is reported.
- Corresponding source and target nodes are highlighted
Applications
Co-evolution of models and codes

**Background:** Codes generated from models are often modified by the programmers. Models are also updated.

**Problem:** Changes by the programmer are lost when the codes are regenerated from the updated model.

**Proposal:** ‘Undo’ of the programmer’s change is represented by the forward transformation of BX. Model updates are propagated using backward transformation.

**Contribution:** Consistent evolution of models and codes. Integrated into Eclipse.

---

“Invoking GrondTram... doing backward transformation ...”

Collaborative development of Bio-models

Collaborative development of Bio-models

Bidirectional graph transformation (non-bijective)  Common model representation

Bijective transformation (conversion/translation)  Edge-labelled graph representation

UnQL query (ex: extract interested parts)  stochastically simulable representation

High-level query  refined view

shared knowledge base

UnQL query (ex: extract interested parts)  refined view

visual model representation

Feedback of verification results
to designed activity

Integration with Unidirectional Transformation

- BXing model transformation language ATL (INRIA, Shibaura Institute of Technology/NII)


Cf. BiQuery/iGRT by Zan Tao et al.
Challenges
From Unordered to Ordered Graphs

**Unordered Graph**

\[ B : V \to \mathcal{P}(L_\varepsilon \times V + Y) \]

\[ B(1) = \{E(a, 4), E(d, 2)\}, \ldots \]

**Ordered Graph**

\[ B : V \to \text{List}(L_\varepsilon \times V + Y) \]

\[ B(1) = [E(a, 4), E(d, 2)], \ldots \]
Ordered branches and mixed

- Our preliminary works
  - Simple bidirectional transformations on ordered graphs
  - Unidirectional transformations with more expressive sibling transformations


Feature-Based Classification of Bidirectional Transformation Approaches

Joint work with Massimo Tisi, Jordi Cabot (Ecole des Mines de Nantes/INRIA), Zhenjiang Hu (NII)

- Domain Analysis of BX
- Feature model on BX
- Clarification of BX design space
- Apply tools to the Feature Model
- Proposals of future research directions/themes

Existing tools and literatures

Exhaustive tool survey has not been the main focus

Existing tools

Analysis of missing/unexplored features
Conclusion

- Framework of *compositional* and well-behaved BX of *graphs*
- Implemented *BX platform for graphs* GRoundTram
  - Used by research groups beside us
- *Applied to model-code co-evolution, synthetic biology,* and other research projects
- Research challenges include BX on *ordered* graphs and *mixed*
Thank you very much for your kind attentions.

More information can be found at http://research.nii.ac.jp/~hidaka/ and http://www.biglab.org/
Appendix


