A Coupled Transformations Chrestomathy

Based on

Compilable and Executable Megamodels

Ralf Lämmel
Software Languages Team
University of Koblenz-Landau, Germany
http://www.softlang.org/
What’s a coupled transformation?

Consistency, e.g., conformance

Transformation, often in the sense of evolution

Artifacts ‘typed’ by languages

---

One transformation necessitates another transformation.
What’s a coupled transformation chrestomathy?

It’s a collection of CX useful for learning.

© 2016, Software Languages Team, University of Koblenz-Landau, Ralf Lämmel
What’s a megamodel and why in the world do we need to compile it?

Let’s see later.
Let’s instantiate the megamodel!
CX by mapping

FRL — Family ("by reference") Language
MML — Meta Modeling Language
DDL — Data Definition Language
TDL — Term Difference Language

Everything is linked to artifacts!
Why such a chrestomathy?

❖ CX are relatively complex transformations.
❖ Notions covered: deltas, incrementality, consistency, …
❖ The chrestomathy helps capturing the essence.
❖ Different CX forms and scenarios can be compared.

How to do such a chrestomathy?

❖ Use a software language repository (SLR).
❖ Use technology-neutral metaprogramming.
❖ We use the (Prolog-centric) YAS: Yet Another SLR.
❖ We use megamodeling for executable patterns of CX.
Demo time

git clone https://github.com/softlang/yas.git
cd yas
make // if you have SWI-Prolog installed
make view // if you have GraphViz/dot installed
find . -name “*.egl” // This lists EGL grammars.

© 2016, Software Languages Team, University of Koblenz-Landau, Ralf Lämmel
CX by \textit{incremental mapping}

\begin{itemize}
  \item \(a : L_1\)
  \item \(b : L_2\)
  \item \(\Delta\)
  \item \(l(\Delta)\)
  \item \(c : L_1\)
  \item \(d : L_2\)
\end{itemize}

Let's instantiate the megamodel!
CX by incremental mapping

FRL — Family ("by reference") Language
MML — Meta Modeling Language
DDL — Data Definition Language
MMDL — MetaModel Difference Language

FRL/mm.mml : MML

FRL/ll/mm.mml

\( \Delta \) : MMDL

mmdlToDdl(\( \Delta \))

FRL/dd.ddl : DDL

FRL2/mm.mml : MML

FRL2/dd.ddl : DDL

Everything is linked to artifacts!
CX by invariant consistency

Let's instantiate the megamodle!

\[ a : L_1 \]
\[ b : L_2 \]
\[ c : L_1 \]

\[ I_1(t) \]
We only permit the subset of EGTL which serves language extension. See here.
CX by co-transformation

Let's instantiate the megamodel!
CX by co-transformation

BSL — Basic Signature Language
Term — Terms conforming to signature
BSTL — Basic Signature Transformation Language

Everything is linked to artifacts!
Delta mapping
State-based lenses
Delta-based lenses
Lenses with complements
...
High-level megamodel CX by co-transformation

LAL megamodel \texttt{cx.transformation}

\begin{verbatim}
reuse coupling
reuse interpretation [ L_2 \mapsto L_1, \text{Any}_2 \mapsto \text{Any}_1 ]
reuse interpretation [ L_1 \mapsto L_2, \text{Any}_1 \mapsto \text{Any}_2 ]
axiom consistency \{ \forall t \in XL. \forall a, c \in L_1. \forall b, d \in L_2.
\quad \begin{align*}
& \text{consistent}(a, b) \\
& \land \text{interpret}(t, a) = c \\
& \land \text{interpret}(t, b) = d \Rightarrow \text{consistent}(c, d) \}
\end{align*}
\end{verbatim}
Low-level megamodel CX by co-transformation

Ueber megamodell BSTL/tests/trafo1.ueber

[ elementOf('trafo1.term',bstl(term)),
  elementOf('term1.term',term),
  elementOf('term2.term',term),
  elementOf('sig1.term',bsl(term)),
  elementOf('sig2.term',bsl(term)),
  relatesTo(conformsTo,['term1.term','sig1.term']),
  mapsTo(interpret,['trafo1.term','term1.term'],['term2.term'])),
  mapsTo(interpret,['trafo1.term','sig1.term'],['sig2.term'])),
  relatesTo(conformsTo,['term2.term','sig2.term']) ].

© 2016, Software Languages Team, University of Koblenz-Landau, Ralf Lämmel
Megamodel compilation

• Forall becomes exists
• Implication becomes conjunction
• ...
• Instantiate languages, artifacts, functions, relations.
• Rely on interpretations at low level.
End of Talk — Thanks!

Work on megamodelling is joint work at softlang with:

- Andrei Varanovich
- Marcel Heinz
- Lukas Härtel
- Johannes Härtel

Thanks also to the SLE course 2016, specifically:

- Lukas Debald
- Christopher Held
- Philipp Seifer