Coupled Software Transformations

Revisited

Ralf Lämmel
Software Languages Team
University of Koblenz-Landau, Germany
http://www.softlang.org/
A long time ago (at an unknown workshop (SET’04)) …

Problems with the past:
• CX (or BX) has developed ever since.
• We don’t like figures without meaning anymore.
• Things shall be illustrated, validated, reproducible.
Coupled Software Transformations—Revisited

Ralf Lämmel

Software Languages Team, http://www.softlang.org/
University of Koblenz-Landau
Germany

"Everything" is linked to the repo!
What’s a coupled transformation (CX)?

- **x : L**: Artifacts ‘typed’ by languages
- **Transformation**, often in the sense of **evolution**
- Consistency, e.g., conformance

Changes imply co-changes to reestablish consistency.

© 2016, Software Languages Team, University of Koblenz-Landau, Ralf Lämmel
• **What are we doing?**
  • Model ‘patterns’ of CX.
  • Capture properties of transformations.
  • Instantiate ‘patterns’ as test cases.

• **Why are we doing it?**
  • Provide a CX chrestomathy (‘useful for learning …’).
  • Introduce a logic-based form of testable megamodels.

• **How are we doing it?**
  • Design a domain-specific predicate logic.
  • Design and implement a logic-based test framework.
  • Implement CX examples in Prolog (so it happens).
Run YAS
(Yet Another SLR
(Software Language Repository))

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 "*.lal" // This lists megamodels.

© 2016, Software Languages Team, University of Koblenz-Landau, Ralf Lämmel
How do the megamodels look like?

sort Any // The universe to draw elements from
sort L ⊆ Any // A language as a subset of the universe

reuse language [ L ↦ MathML, Any ↦ XML ]
link MathML to 'https://www.w3.org/TR/MathML3'
link XML to 'https://www.w3.org/XML'

reuse language // The defined language
reuse language [ L ↦ DefL, Any ↦ DefAny ]
constant defL : DefL // The language definition
relation conformsTo : Any × DefL
axiom { ∀x ∈ Any. x ∈ L ⇔ conformsTo(x, defL) }

reuse conformance [ Any ↦ XML, DefAny ↦ XML,
L ↦ MathML, DefL ↦ XSD, defL ↦ MathMLSchema ]
link XML to 'https://www.w3.org/XML'
link XSD to 'https://www.w3.org/XML/Schema'
link MathML to 'https://www.w3.org/TR/MathML3'
link MathMLSchema to 'https://www.w3.org/Math/XMLSchema'

© 2016, Software Languages Team, University of Koblenz-Landau, Ralf Lämmel
The ‘pattern’ of CX by mapping

Let’s instantiate the pattern!
An ‘instance’ of CX by *mapping*

FRL — Family ... Language  
MML — Metamodeling Language  
DDL — Data Definition Language  
TDL — Term Difference Language  

Everything is linked to artifacts!
The ‘pattern’ of CX by incremental mapping

\[ \Delta \]

\[ l(\Delta) \]
An ‘instance’ of CX by *incremental mapping*

FRL — Family … Language
MML — Metamodeling Language
DDL — Data Definition Language
MMDL — Metamodel Difference Language

```
mmdlToDdl(\Delta)
```

![Diagram showing the mapping process](image-url)
The ‘pattern’ of CX by *invariant consistency*
An ‘instance’ of CX by *invariant consistency*

BNL — Binary Number Language  
BGL — Basic Grammar Language  
EGL — Extended Grammar Language  
EGTL — Extended Grammar Transformation Language

We only permit the subset of EGTL which serves language extension. See [here](#).
The ‘pattern’ of CX by co-transformation

\[ a : L_1 \quad l_1(t) \quad c : L_1 \]
\[ b : L_2 \quad l_2(t) \quad d : L_2 \]
An ‘instance’ of CX by co-transformation

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

sig1.bsl : BSL

bstlSig(trafo1.term)

sig2.bsl : BSL

bstlTerm(trafo1.term)

term1.term : Term

conformance

conformance

term2.term : Term
More CX

Lenses
Complements
Symmetry

...
Higher level megamodel for CX by co-transformation

\[
\begin{align*}
& a : L_1 & b : L_2 \\
& l_1(t) & l_2(t) \\
& c : L_1 & d : L_2
\end{align*}
\]

**LAL megamodel cx.cotransformation**

- **reuse coupling**
- **reuse interpretation** \[ L_2 \leftrightarrow L_1, \text{Any}_2 \leftrightarrow \text{Any}_1 \]
- **reuse interpretation** \[ L_1 \leftrightarrow L_2, \text{Any}_1 \leftrightarrow \text{Any}_2 \]
- **axiom consistency** \{ \forall t \in XL. \forall a, c \in L_1. \forall b, d \in L_2.
  \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*} \}
Lower level megamodel CX by co-transformation

Ueber megamodel BSTL/tests/trafo1.ueber

[ elementType('trafo1.term', bstl(term)),
          elementType('term1.term', term),
          elementType('term2.term', term),
          elementType('sig1.term', bsl(term)),
          elementType('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']) ].
Megamodel compilation for CX by co-transformation

**LAL megamodel**

- **reuse coupling**
- **reuse interpretation** \[ L_2 \leftrightarrow L_1, \text{Any}_2 \leftrightarrow \text{Any}_1 \]
- **reuse interpretation** \[ L_1 \leftrightarrow L_2, \text{Any}_1 \leftrightarrow \text{Any}_2 \]
- **axiom consistency** \{ \forall t \in XL. \forall a, c \in L_1. \forall b, d \in L_2.\]
  \[
  \text{consistent}(a, b) \\
  \land \text{interpret}(t, a) = c \\
  \land \text{interpret}(t, b) = d \Rightarrow \text{consistent}(c, d) \}

**Ueber megamodel**

\[
\begin{align*}
\text{elementOf}(&'\text{trafo1.term}', \text{bstl(term)}), \\
\text{elementOf}(&'\text{term1.term}', \text{term}), \\
\text{elementOf}(&'\text{term2.term}', \text{term}), \\
\text{elementOf}(&'\text{sig1.term}', \text{bsl(term)}), \\
\text{elementOf}(&'\text{sig2.term}', \text{bsl(term)}), \\
\text{relatesTo}(\text{conformsTo}, ['\text{term1.term}', '\text{sig1.term}']), \\
\text{mapsTo}(\text{interpret}, ['\text{trafo1.term}', '\text{term1.term}'], ['\text{term2.term}']), \\
\text{mapsTo}(\text{interpret}, ['\text{trafo1.term}', '\text{sig1.term}'], ['\text{sig2.term}']), \\
\text{relatesTo}(\text{conformsTo}, ['\text{term2.term}', '\text{sig2.term}'])
\end{align*}
\]
Configuration of compilation from higher to lower level megamodel

LAL configuration cx.cotransformation

```
[ sort('L1', term),
sort('Any1', term),
sort('L2', bsl(term)),
sort('Any2', term),
sort('XL', bstl(term)),
sort('XAny', term),
relation(consistent, conformsTo),
axiom(consistency, [
    (t, 'trafo1.term'),
    (a, 'term1.term'),
    (b, 'sig1.term'),
    (c, 'term2.term'),
    (d, 'sig2.term') ])].
```
Summary of megamodel compilation

- A **limited** subset of predicate logic is considered.
- For all becomes exists
- Implication becomes conjunction
- ...
- Instantiate languages, artifacts, functions, relations.
- Rely on interpretations at low level.
Parsing

Inlining modulo substitution

Well-formedness checking

Test execution

Translation

Unparsing

Mega-model (LAL)

AST (LAL)

Problems

Test cases (Ueber)

Configuration

Mega-model (LAL)

AST (LAL)

Software Language Repository (YAS)
The basic skill set of software language engineering

Ralf Lämmel
Software Languages Team
University of Koblenz-Landau
Germany

This is sort of an advertisement for the upcoming textbook on software languages:

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 2016 course, specifically:

• Lukas Debald
• Christopher Held
• Philipp Seifer