Logic-based, Executable ~~Megamodels of Coupled~~ Transformations

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

<u>Tool demo</u> on YAS

(Yet Another SLR

(Software Language Repository))

with applications to logic-based, executable megamodeling of CX

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

What's a *coupled transformation* (CX)?

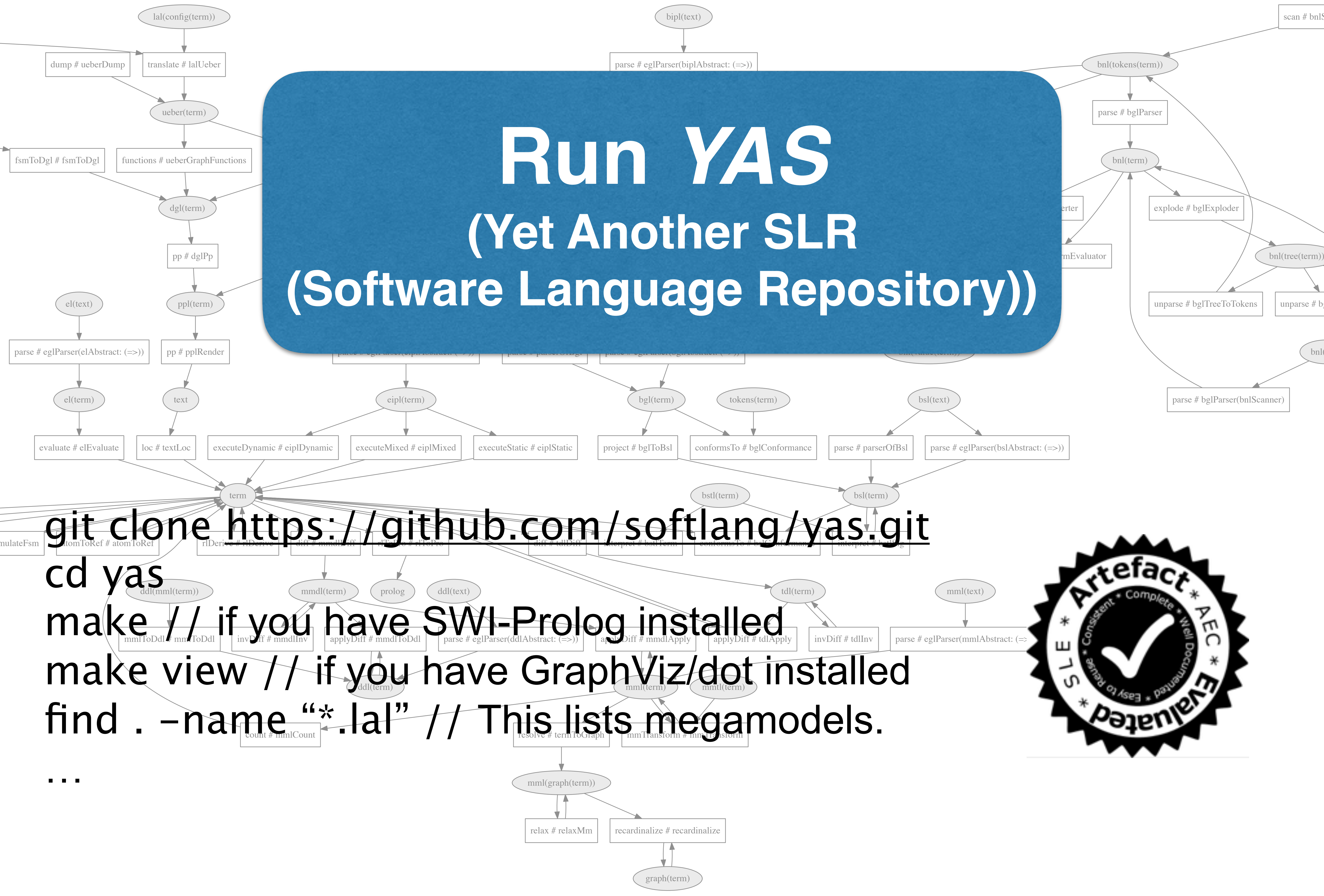
$\boxed{x : L}$	Artifacts 'typed' by languages
$\longrightarrow$	Transformation, often in the sense of **evolution**
$- - - - -$	Consistency, e.g., conformance

(See SLE 2016 paper.)

$$\begin{tikzcd}
\boxed{a : L_1} \arrow[r, dash, dashed] \arrow[d] & \boxed{b : L_2} \arrow[d] \\
\boxed{c : L_1} \arrow[r, dash, dashed] & \boxed{d : L_2}
\end{tikzcd}$$

Changes imply co-changes to reestablish consistency.

- **<u>What</u> are we doing?**
  - Model ‘patterns’ of CX.
  - Capture properties of transformations.
  - Instantiate ‘patterns’ as test cases.
- **<u>Why</u> are we doing it?**
  - Provide a CX chrestomathy (‘useful for learning …’).
  - Provide a logic-based form of testable megamodels.
- **<u>How</u> are we doing it?**
  - Set up a suitable predicate logic.
  - Set up a declarative test framework.
  - Implement all 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.
...
```

How do the *megamodels* look like?

LAL megamodel
language

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

LAL megamodel
language.mathml

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

LAL megamodel
conformance

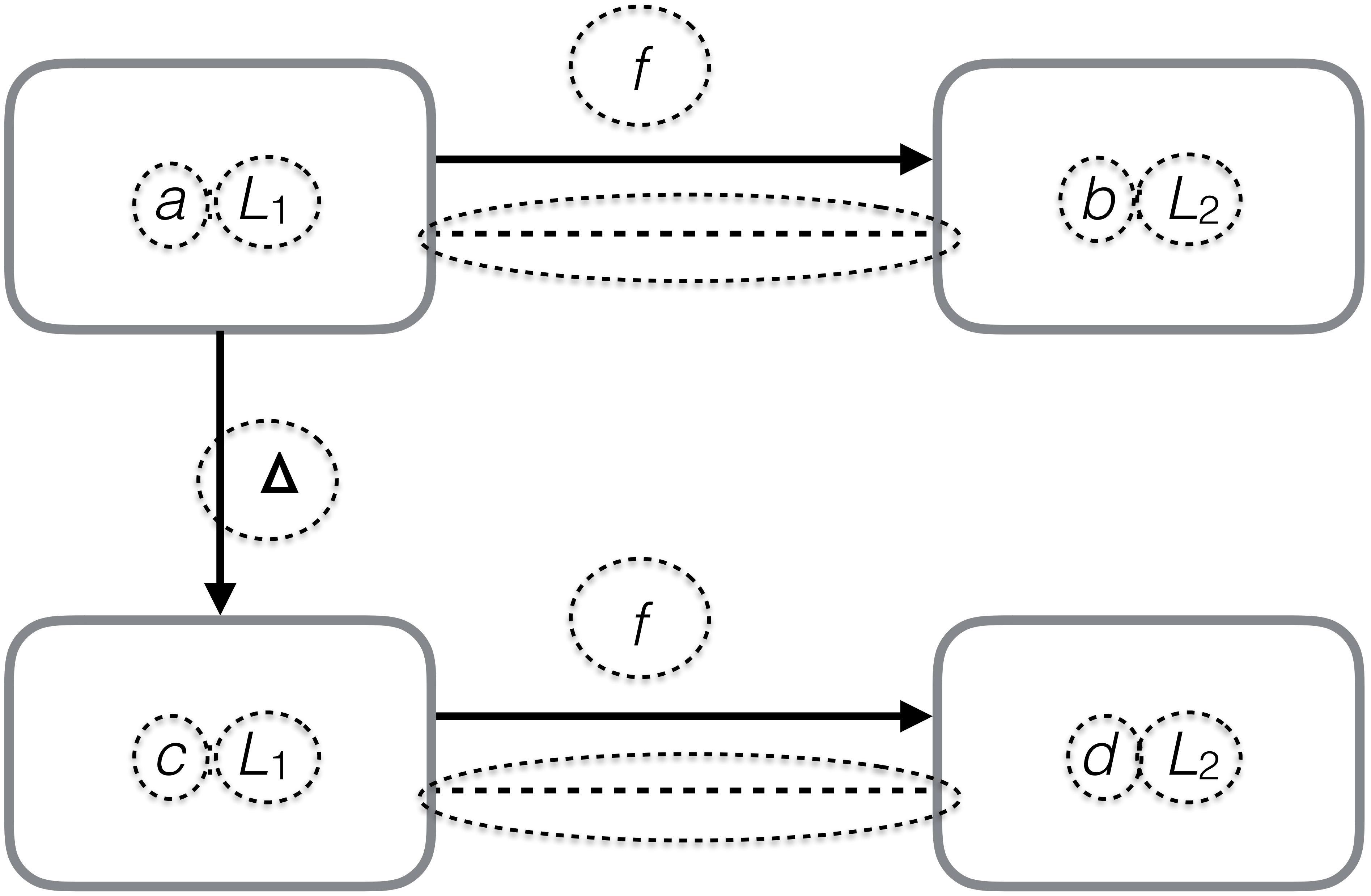
```
reuse language // The defined language
reuse language [ L ↦ DefL, Any ↦ DefAny ]
constant defL : DefL // The language definition
relation conformsTo : Any × DefL
axiom { ∀x ∈ Any. x ∈ L ⇔ conformsTo(x, defL) }
```

LAL megamodel
conformance.mathml

```
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'
```

The ‘pattern’ of CX by *mapping*

Let’s instantiate the pattern*!*

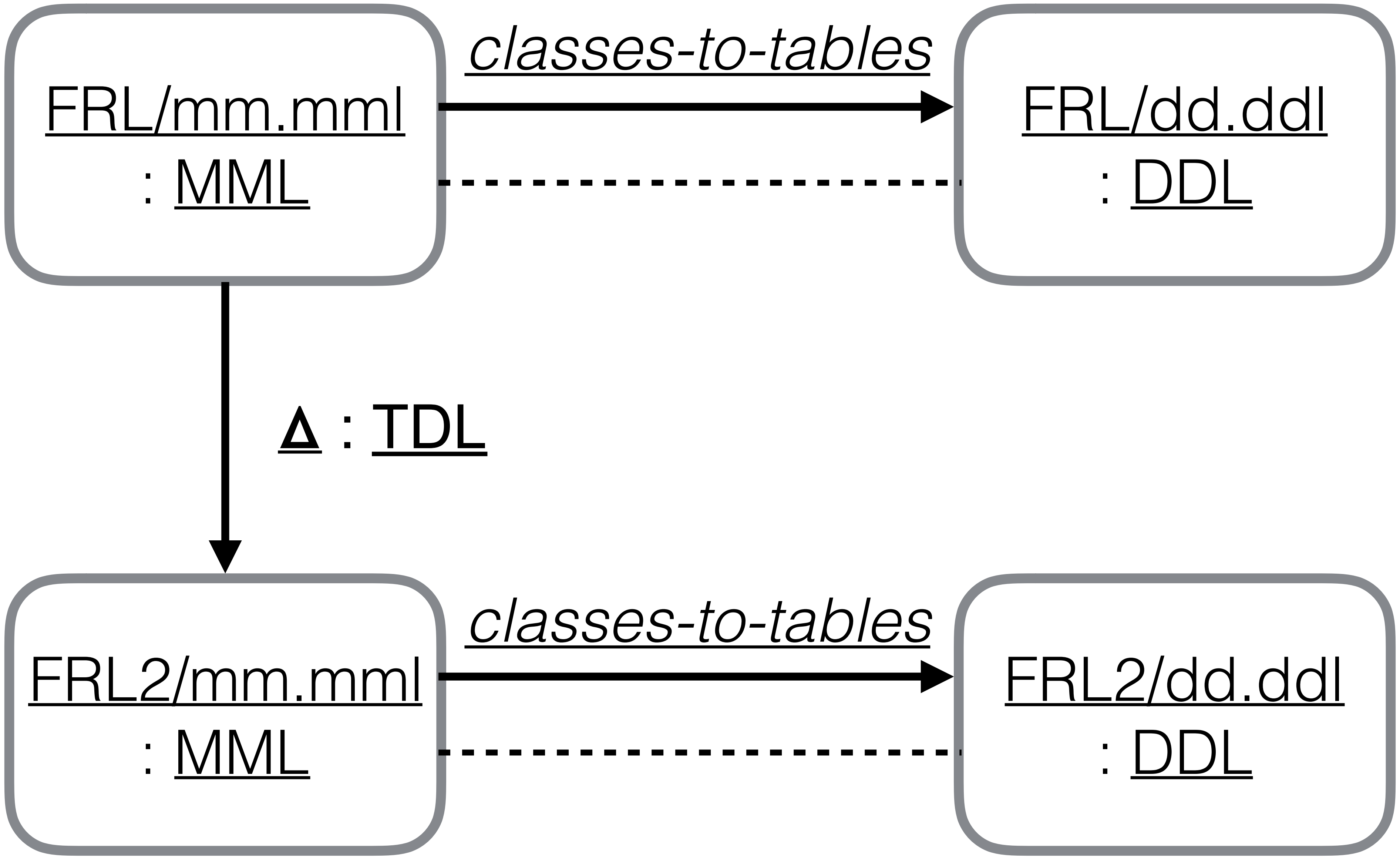

$$\begin{tikzcd} a : L_1 \arrow[r, "f"] \arrow[d, "\Delta"] & b : L_2 \\ c : L_1 \arrow[r, "f"] & d : L_2 \end{tikzcd}$$

1/4

An ‘instance’ of CX by *mapping*

Everything is linked to artifacts*!*

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


$$\begin{tikzcd}
\boxed{\text{FRL/mm.mml : MML}} \arrow[r, "\textit{classes-to-tables}"] \arrow[r, dashed, bend right] \arrow[d, "\Delta : \text{TDL}"] & \boxed{\text{FRL/dd.ddl : DDL}} \\
\boxed{\text{FRL2/mm.mml : MML}} \arrow[r, "\textit{classes-to-tables}"] \arrow[r, dashed, bend right] & \boxed{\text{FRL2/dd.ddl : DDL}}
\end{tikzcd}$$

2/4

The ‘pattern’ of CX by *incremental mapping*

$$\begin{tikzcd}
\boxed{a : L_1} \arrow[r, dash, dashed] \arrow[d, "\Delta"] & \boxed{b : L_2} \arrow[d, "I(\Delta)"] \\
\boxed{c : L_1} \arrow[r, dash, dashed] & \boxed{d : L_2}
\end{tikzcd}$$

2/4

An ‘instance’ of CX by *incremental mapping*

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

$$\begin{tikzcd}
\text{FRL/mm.mml : MML} \arrow[r, dash, dashed, "\text{correspondence}"] \arrow[d, "\Delta : \text{MMDL}"] & \text{FRL/dd.ddl : DDL} \arrow[d, "\text{mmdlToDdl}(\Delta)"] \\
\text{FRL2/mm.mml : MML} \arrow[r, dash, dashed, "\text{correspondence}"] & \text{FRL2/dd.ddl : DDL}
\end{tikzcd}$$

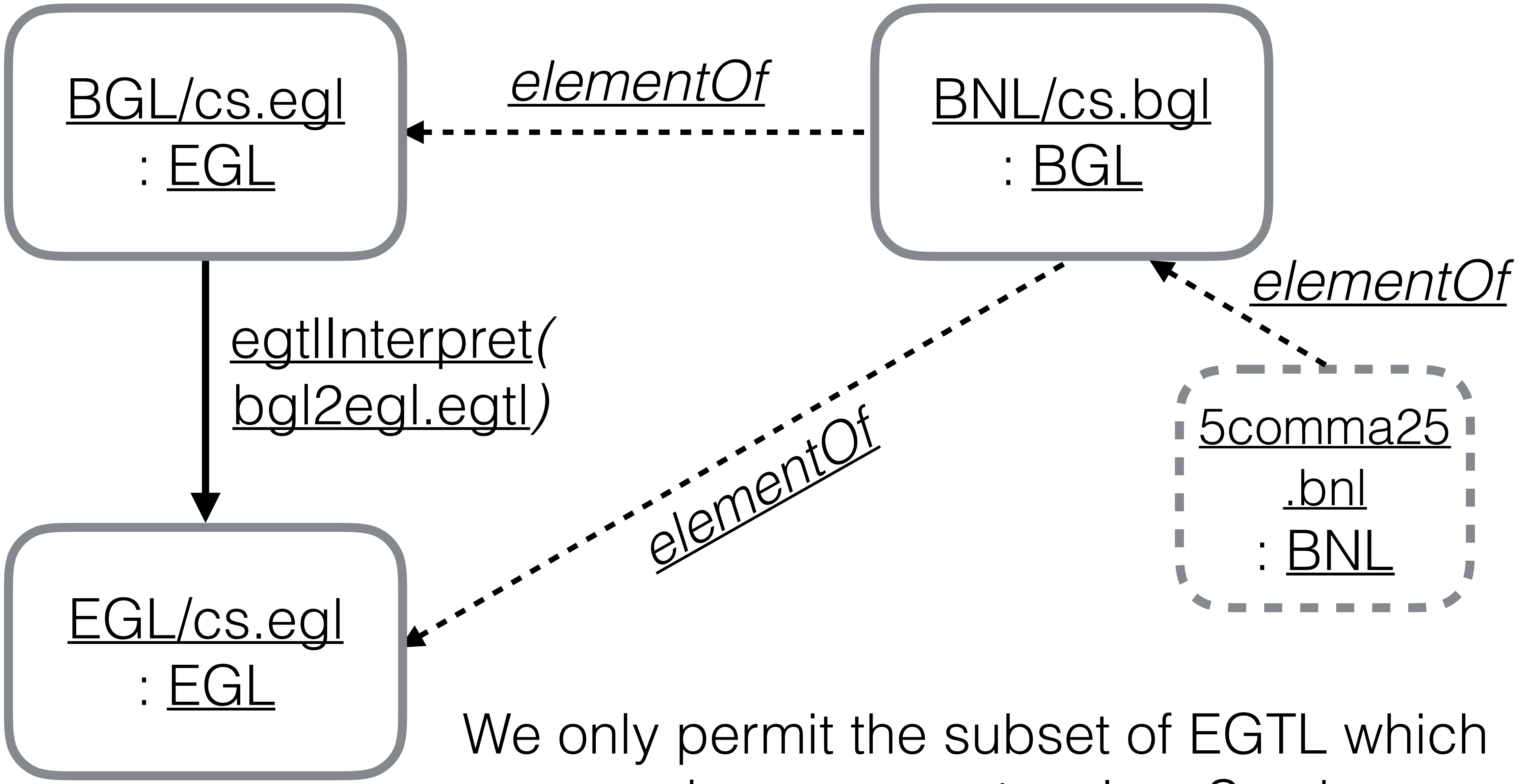
3/4

The ‘pattern’ of CX by *invariant consistency*

$$\begin{tikzcd}
a : L_1 \arrow[r, dash, dashed] \arrow[d, "l_1(t)"] & b : L_2 \\
c : L_1 \arrow[ur, dash, dashed] &
\end{tikzcd}$$

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.

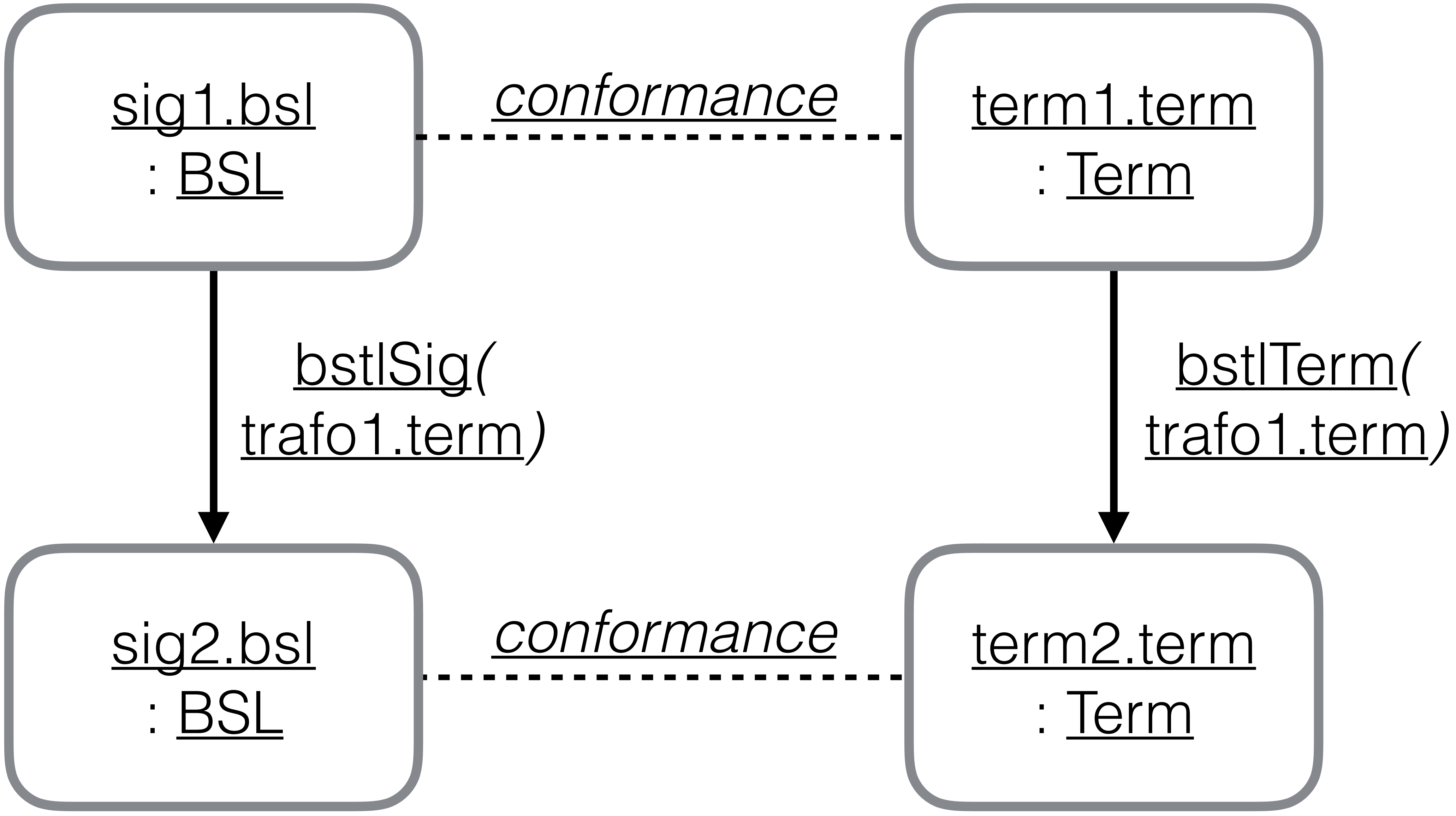
4/4

The ‘pattern’ of CX by *co-transformation*

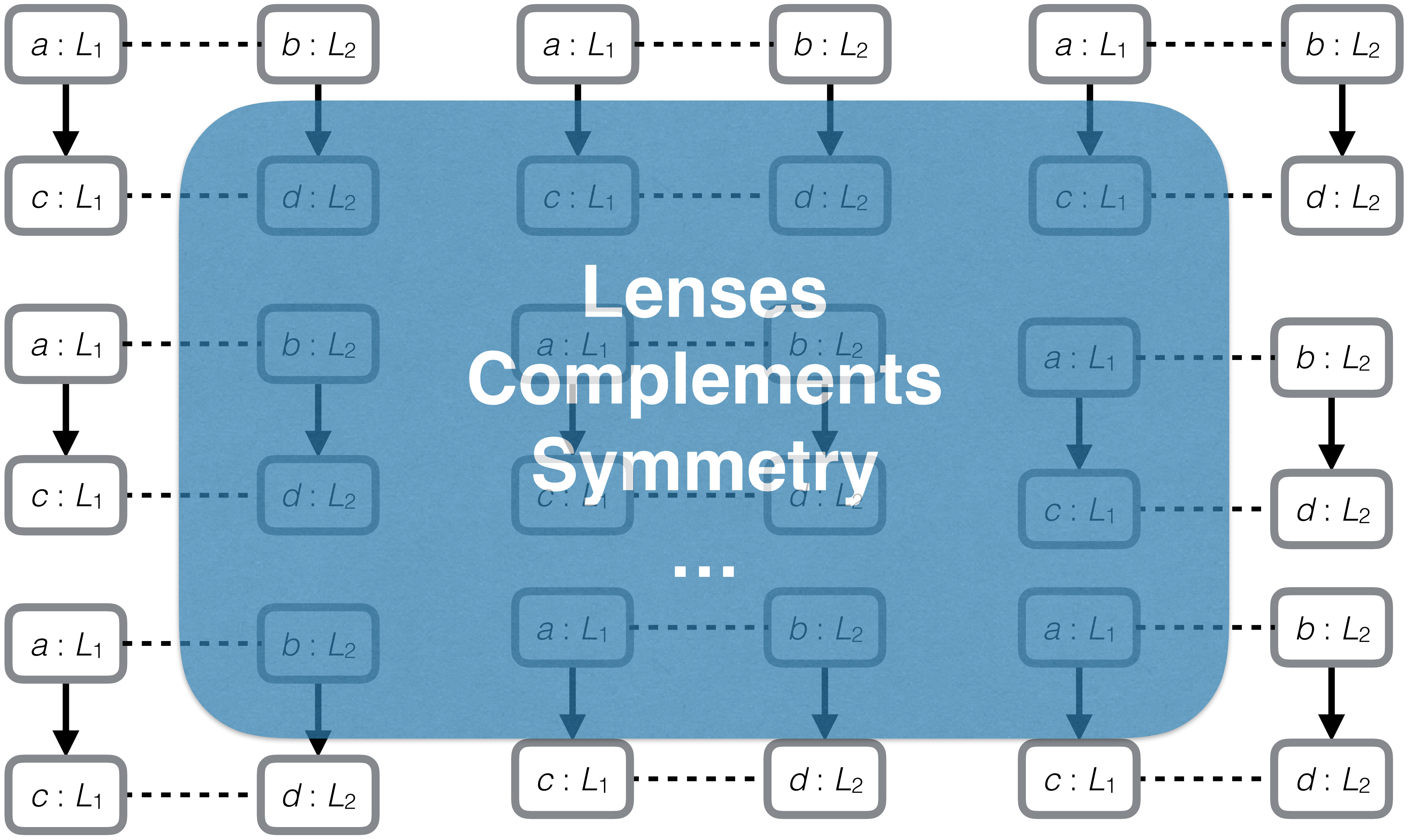
$$\begin{tikzcd}
a : L_1 \arrow[r, dash, dashed] \arrow[d, "l_1(t)"] & b : L_2 \arrow[d, "l_2(t)"] \\
c : L_1 \arrow[r, dash, dashed] & d : L_2
\end{tikzcd}$$

An ‘instance’ of CX by *co-transformation*

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



More CX



Lenses
Complements
Symmetry
…

…

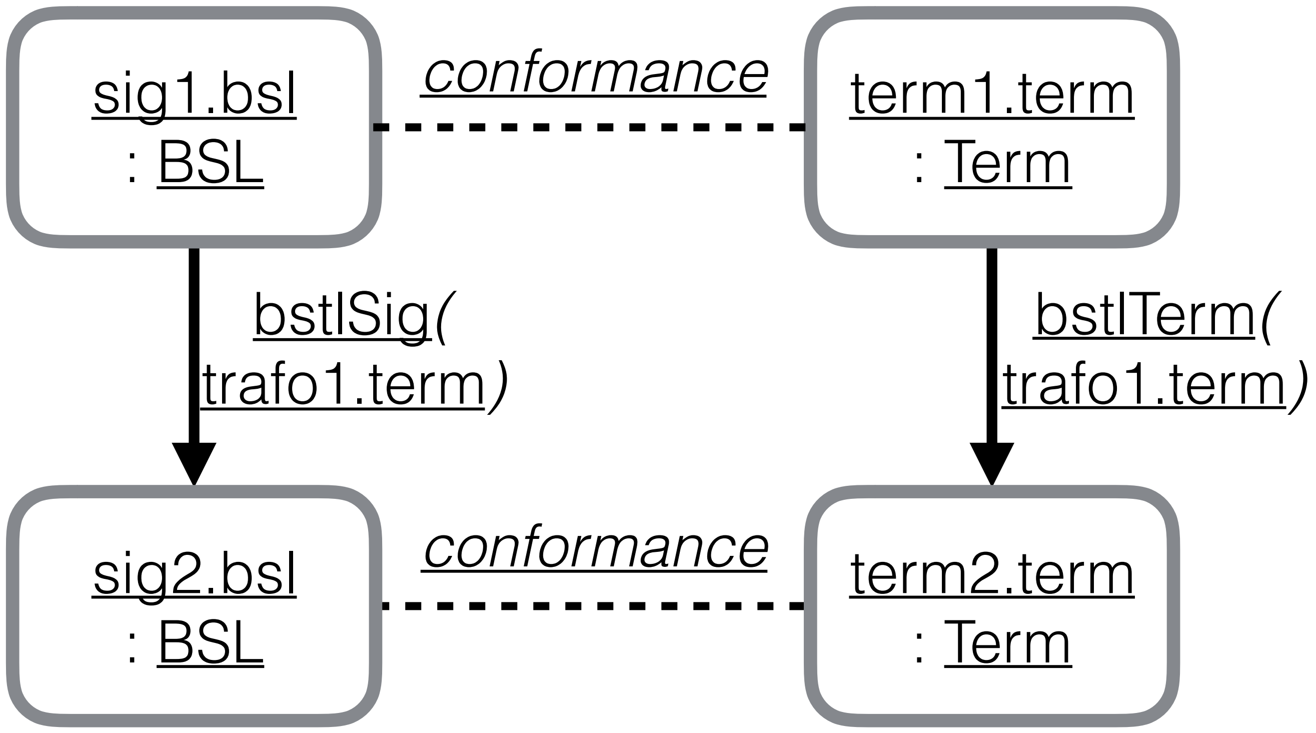
Higher level megamodel for CX by *co-transformation*

$$\begin{tikzcd}
a : L_1 \arrow[r, dash, dashed] \arrow[d, "I_1(t)"] & b : L_2 \arrow[d, "I_2(t)"] \\
c : L_1 \arrow[r, dash, dashed] & d : L_2
\end{tikzcd}$$

LAL megamodel cx.cotransformation

reuse coupling
reuse interpretation $[\ L_2 \mapsto L_1,\ Any_2 \mapsto Any_1\ ]$
reuse interpretation $[\ L_1 \mapsto L_2,\ Any_1 \mapsto Any_2\ ]$
axiom consistency $\{\ \forall t \in XL.\ \forall a, c \in L_1.\ \forall b, d \in L_2.$
  consistent(a, b)
  $\wedge$ interpret(t, a) $=$ c
  $\wedge$ interpret(t, b) $=$ d $\Rightarrow$ consistent(c, d) $\}$

Lower level megamodel CX by *co-transformation*



Ueber megamodel 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']) ].
```

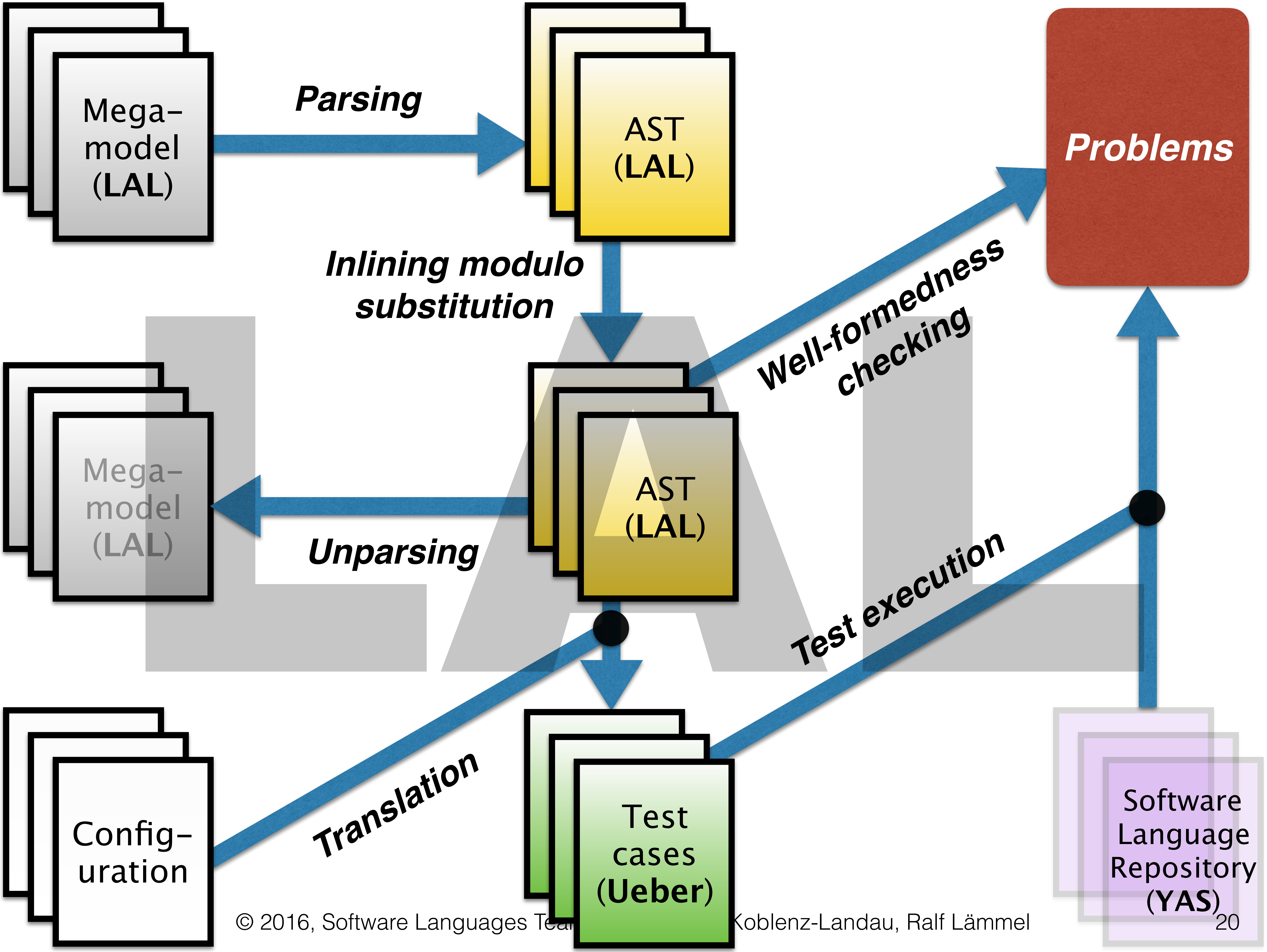
Configuration of compilation from higher to lower level megamodel

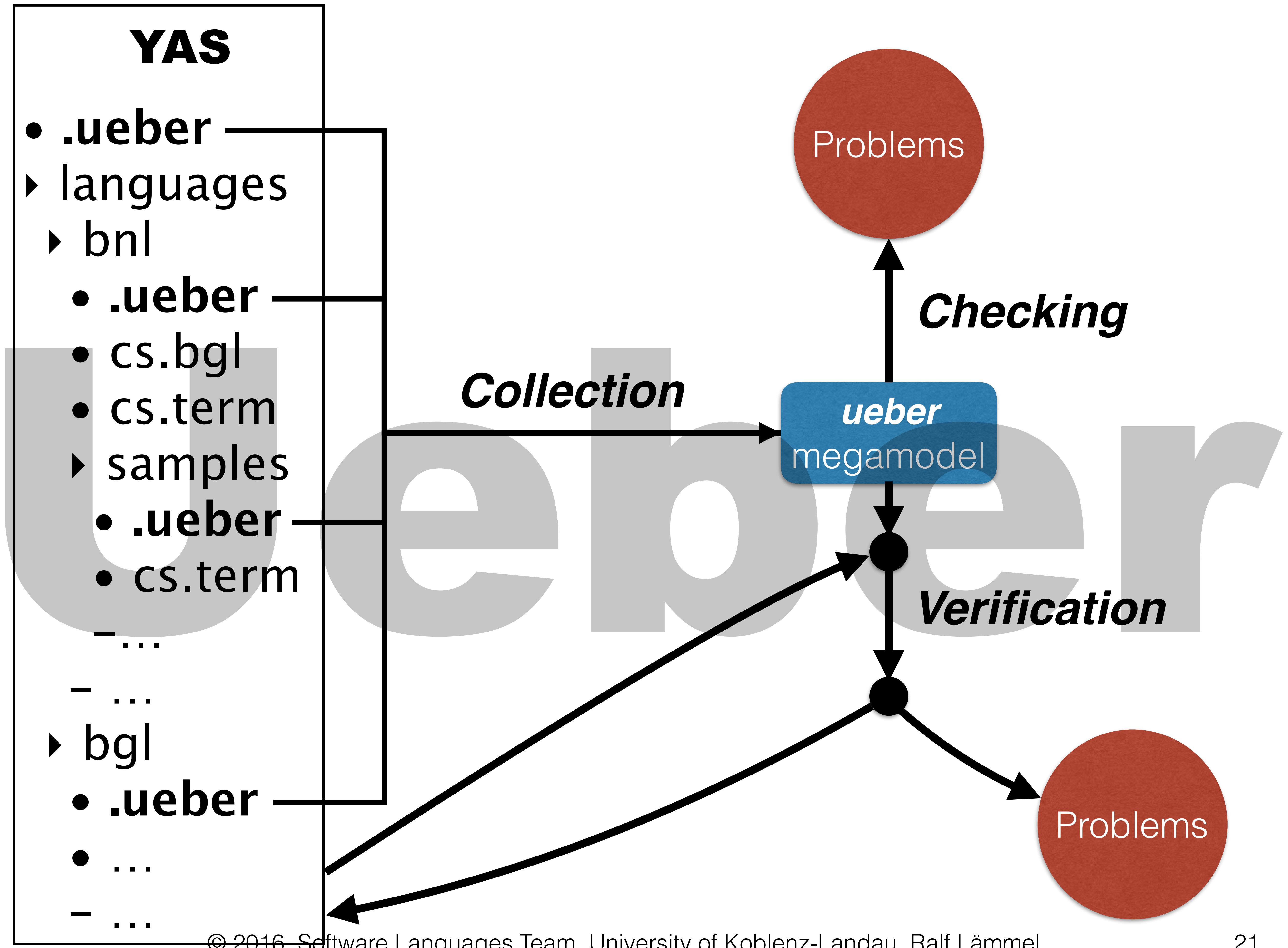
LAL configuration cx.cotransformation

```
[ 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') ])].
```

Megamodel compilation

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


Mega-model (LAL)
Parsing
AST (LAL)
Problems
Inlining modulo substitution
Well-formedness checking
Mega-model (LAL)
Unparsing
AST (LAL)
Test execution
Config-uration
Translation
Test cases (Ueber)
Software Language Repository (YAS)
LAL


YAS
• .ueber
‣ languages
‣ bnl
• .ueber
• cs.bgl
• cs.term
‣ samples
• .ueber
• cs.term
– …
– …
‣ bgl
• .ueber
• …
– …
Collection
ueber megamodel
Checking
Problems
Verification
Problems
ueber

End of Talk — Thanks*!*

Work on megamodeling 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