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)) …

Coupled Software Transformations

— Extended Abstract —

Ralf Lämmel

VUA & CWI, Amsterdam, The Netherlands

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.

Today (SLE 2016)

Coupled Software Transformations—Revisited

Ralf Lämmel

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

“Everything” is linked to the repo!

...

3.1 Languages

Let us express that a language L is a subset of a suitable universe Any (such as ‘all’ strings, trees, or graphs

LAL megamodel *language*

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

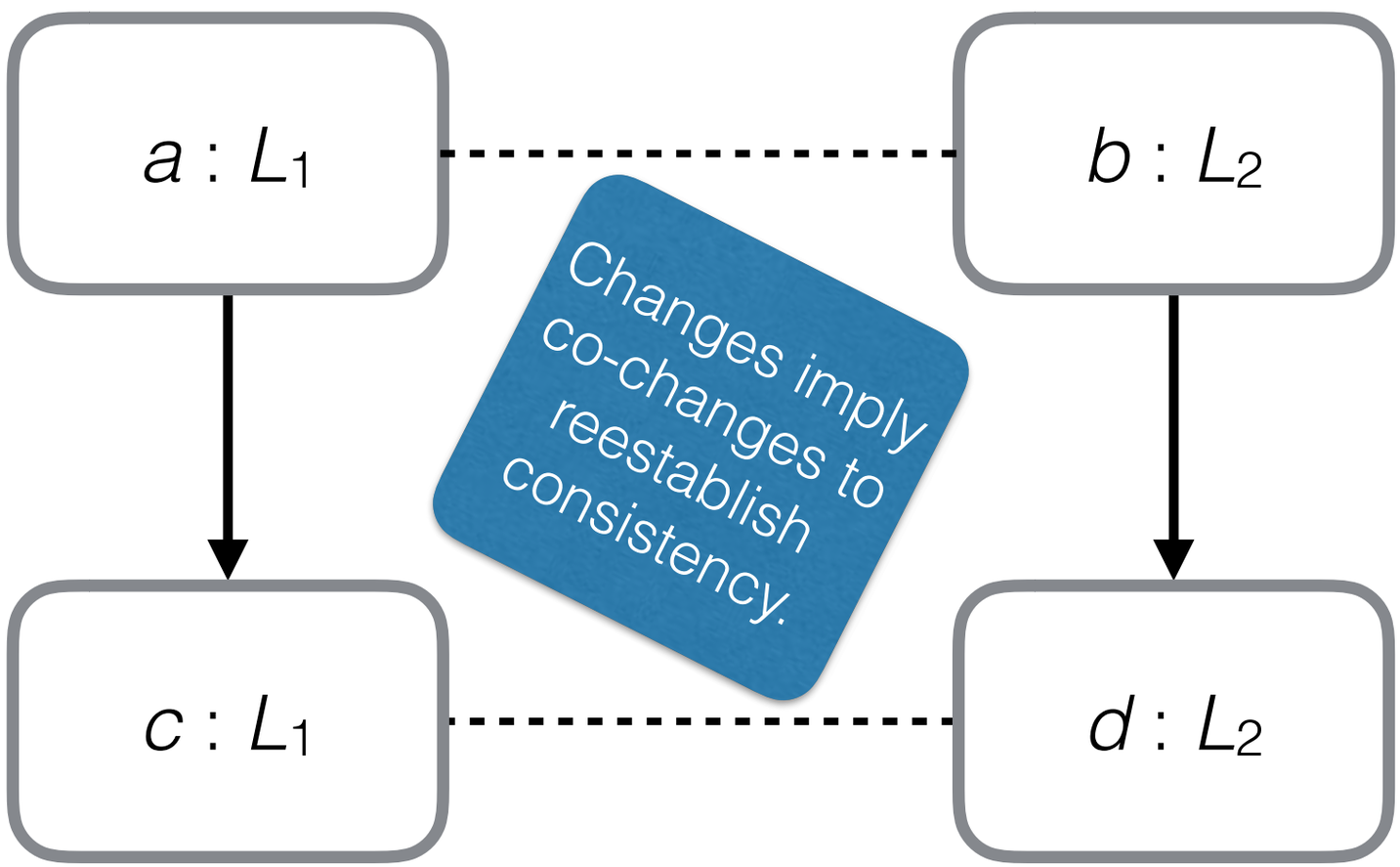
...

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

Artifacts 'typed' by languages

Transformation, often in the sense of **evolution**

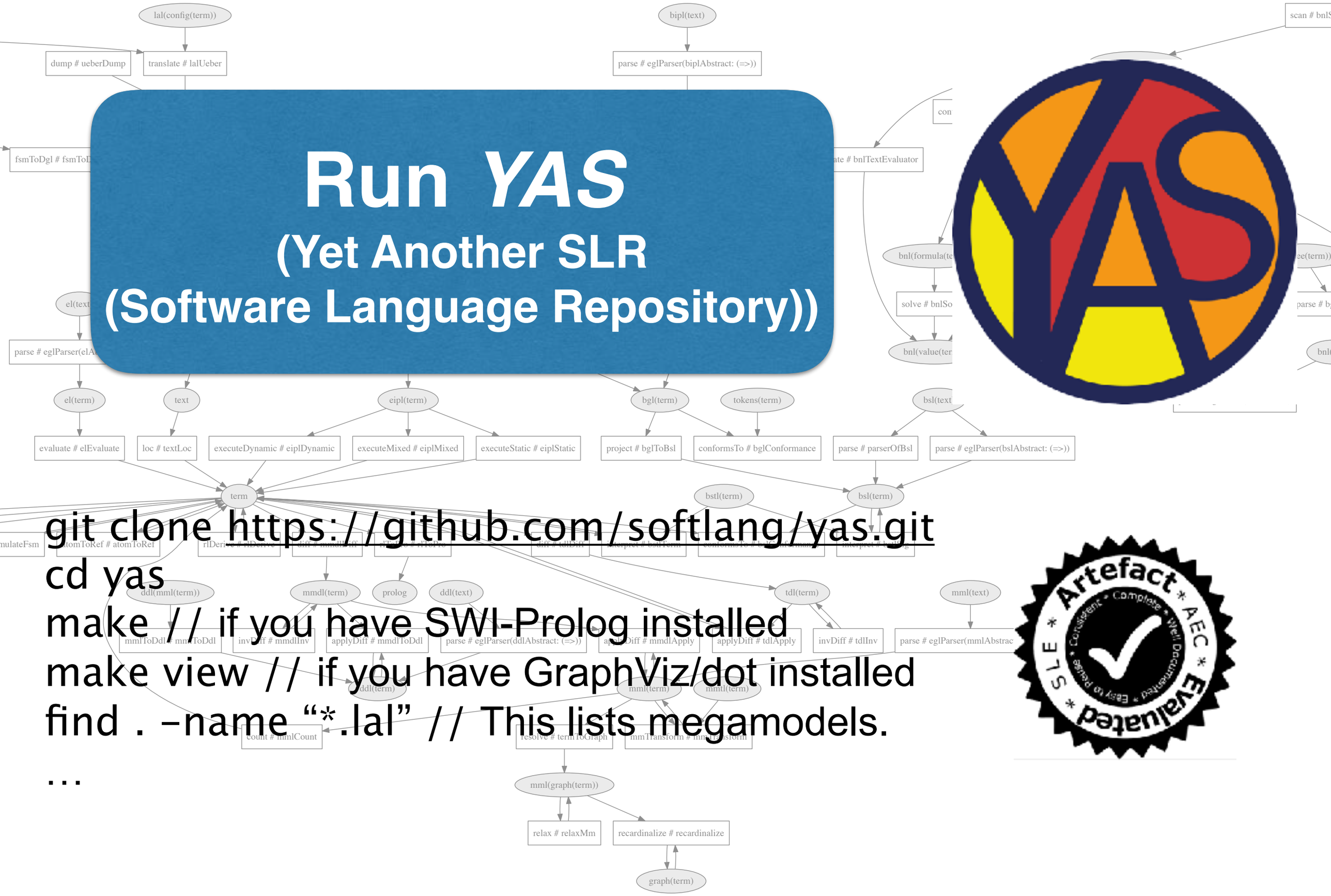
Consistency, e.g., conformance



- **<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 …’).
  - Introduce a logic-based form of testable megamodels.
- **<u>How</u> 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.
...
```

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
```

LAL megamodel
language

```
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
language.mathml

```
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) }
```

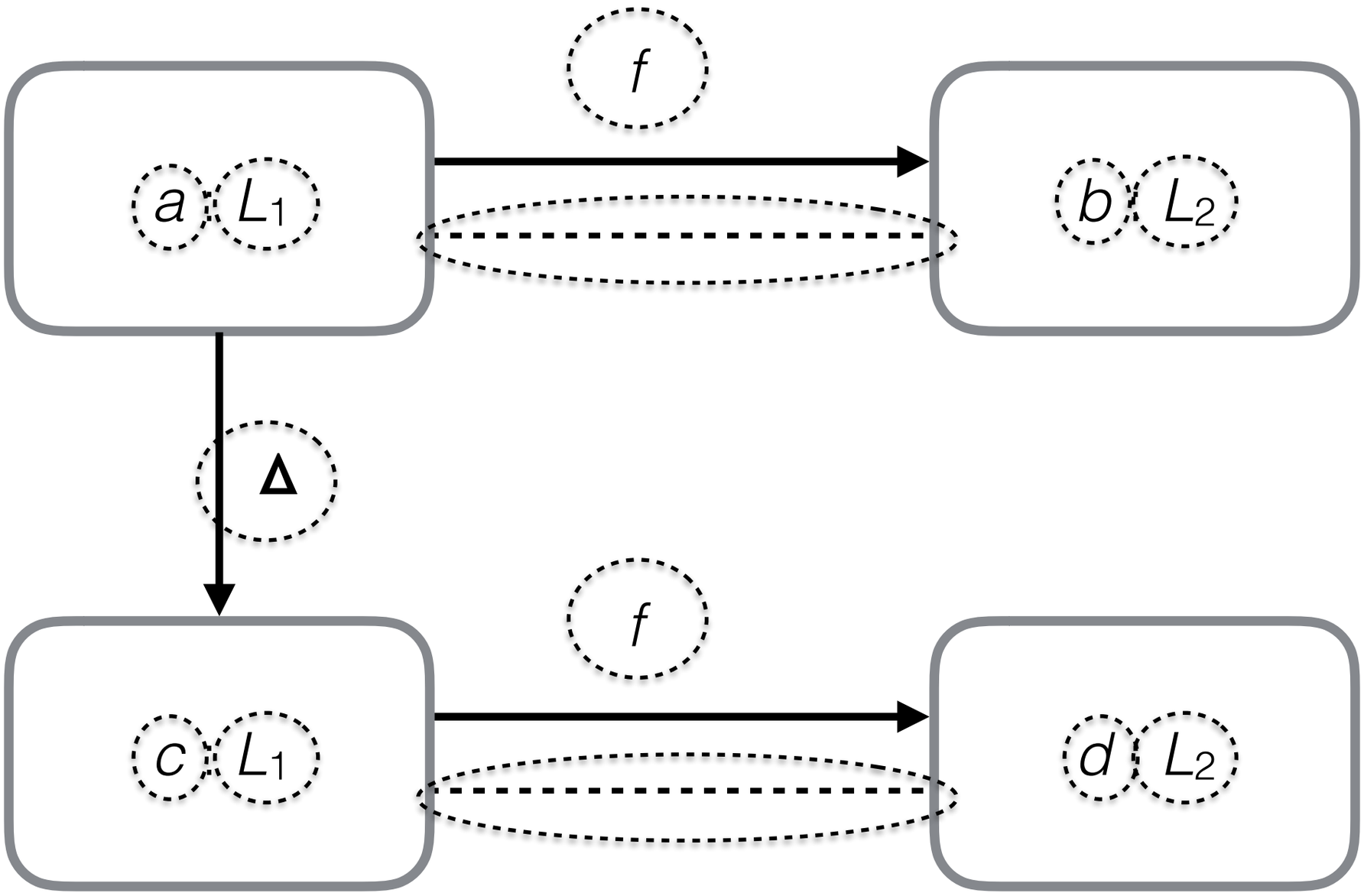
LAL megamodel
conformance

```
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'
```

LAL megamodel
conformance.mathml

The ‘pattern’ of CX by *mapping*

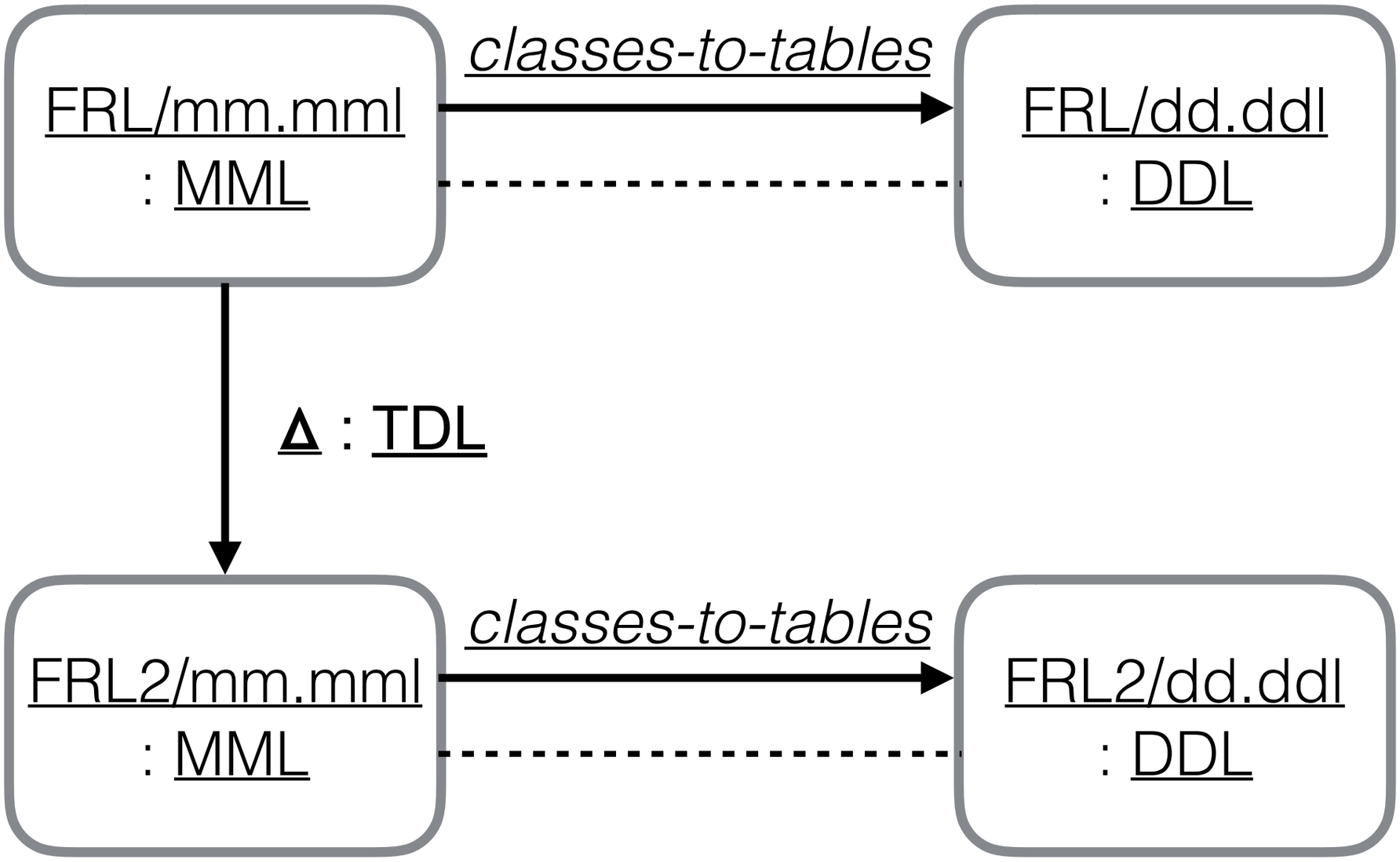
Let’s instantiate the pattern!



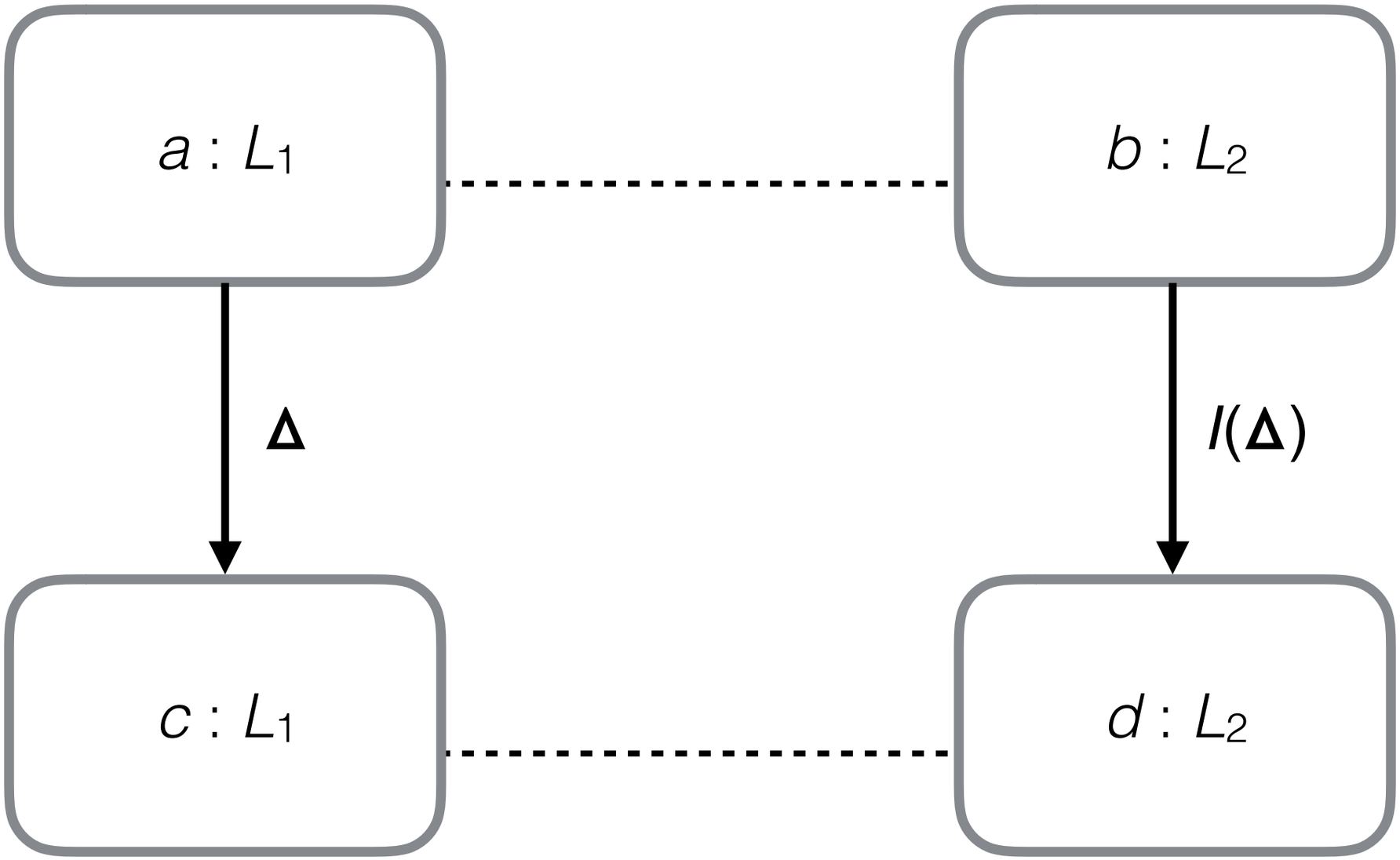
An 'instance' of CX by *mapping*

Everything is linked to artifacts!

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

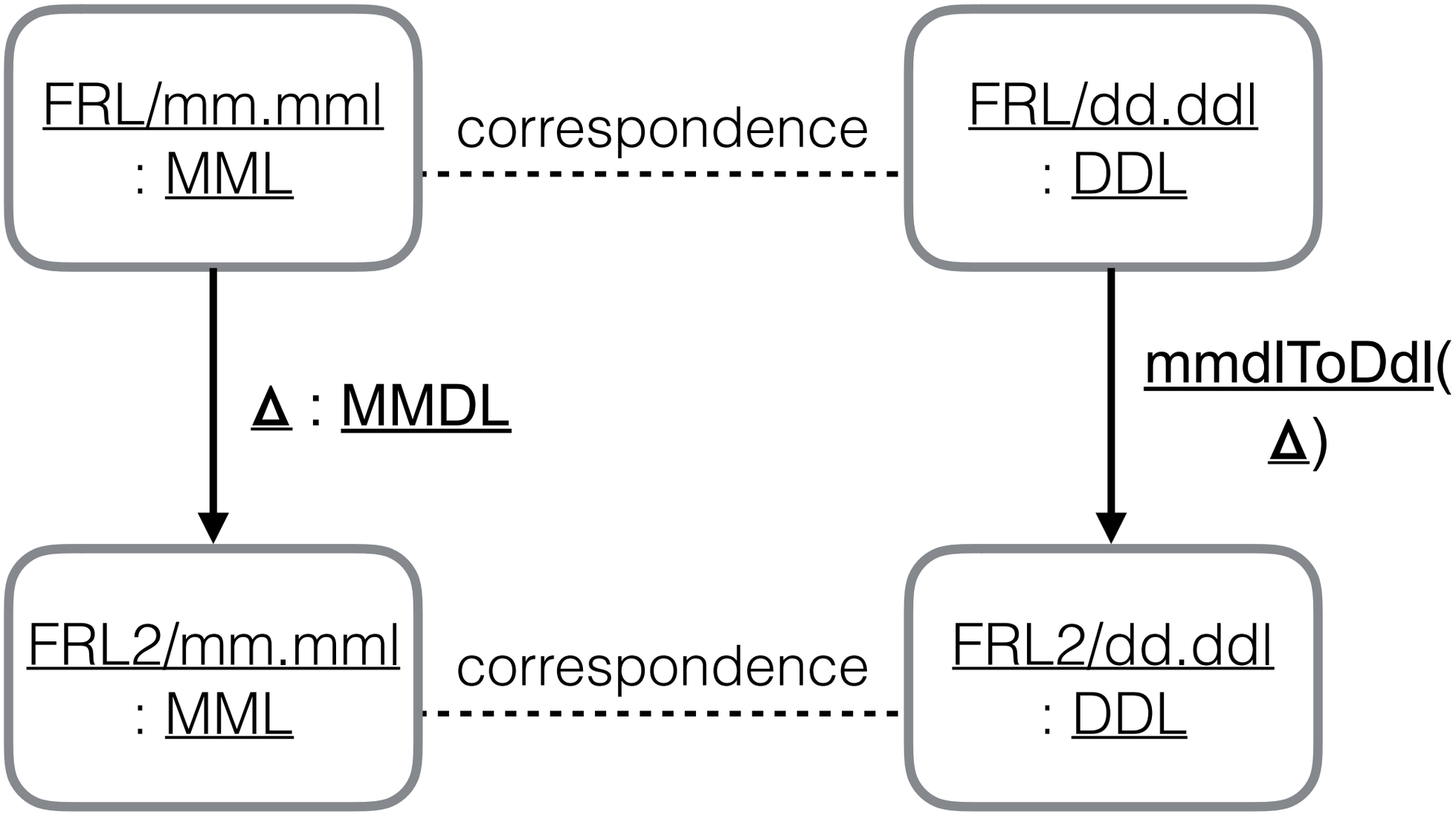


The ‘pattern’ of CX by *incremental mapping*

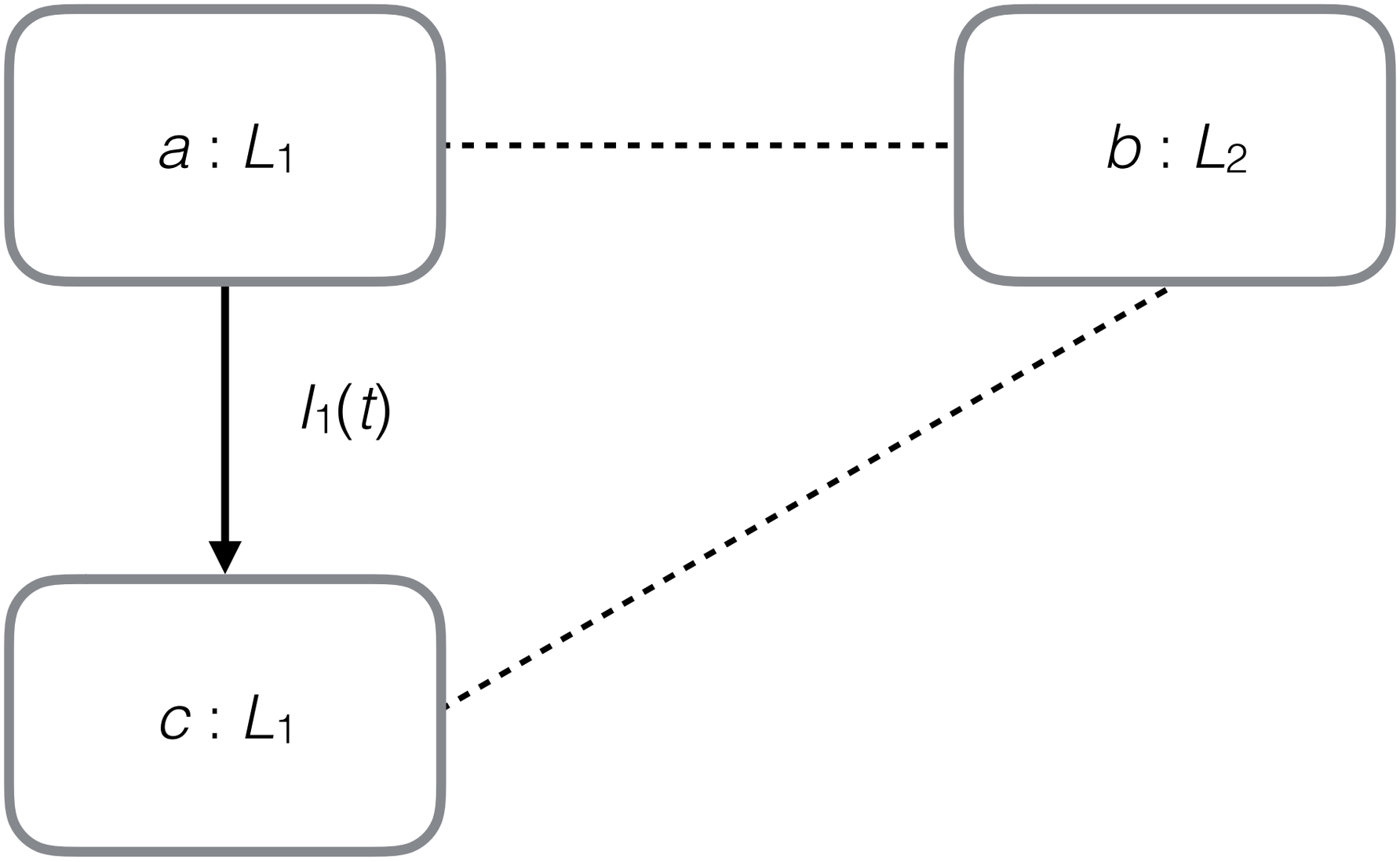


An ‘instance’ of CX by *incremental mapping*

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

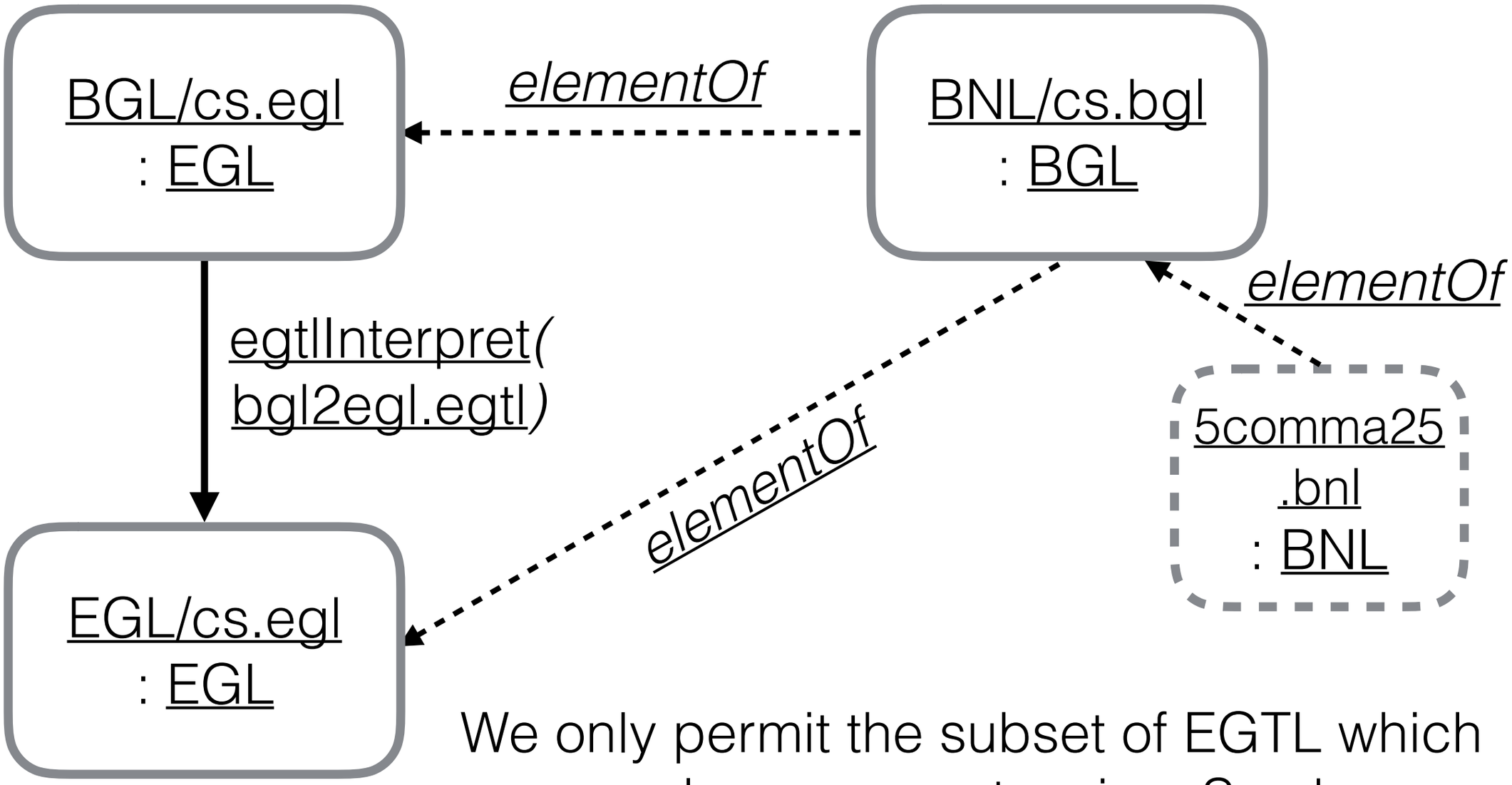


The ‘pattern’ of CX by *invariant consistency*



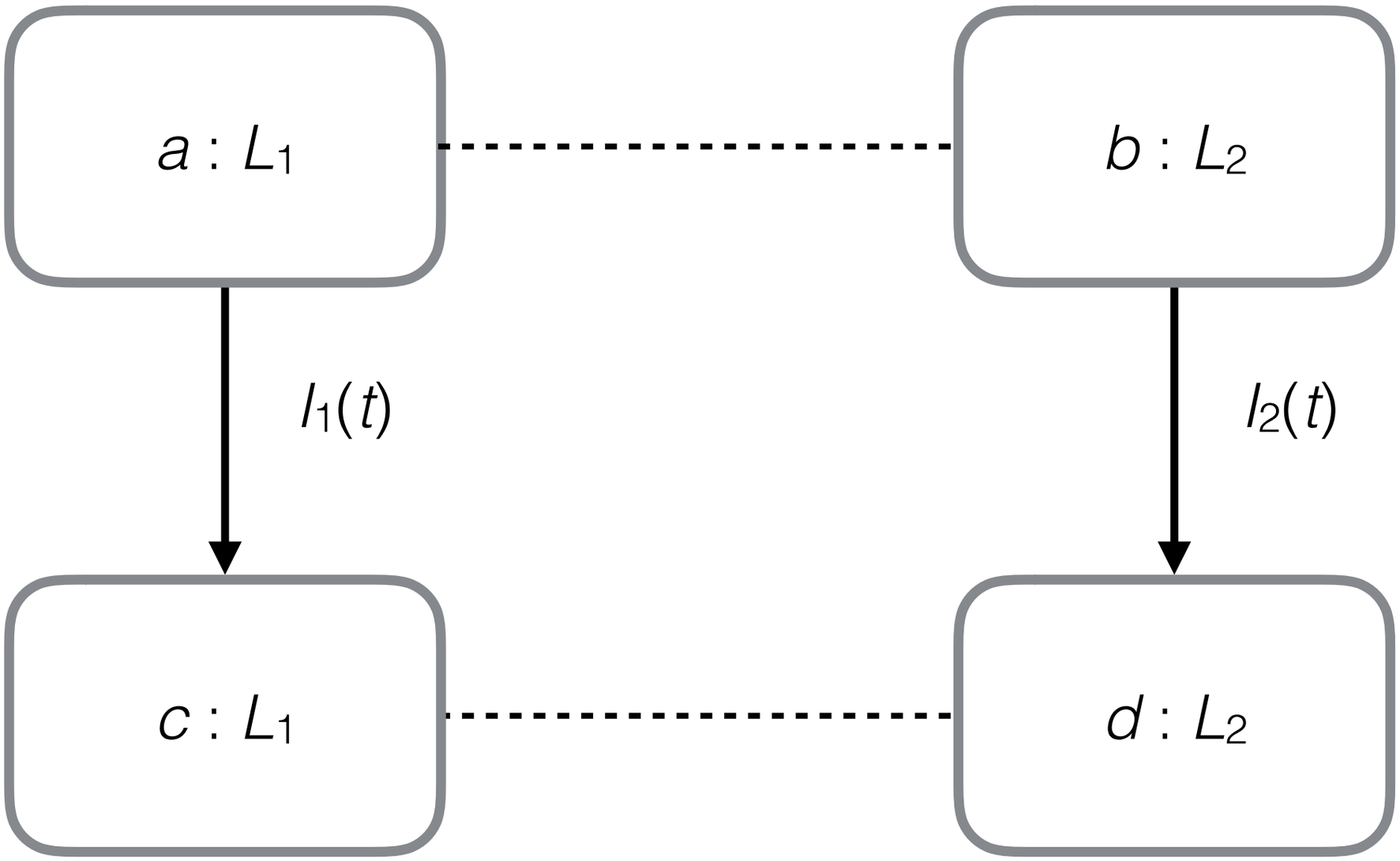
An 'instance' of CX by *invariant consistency*

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



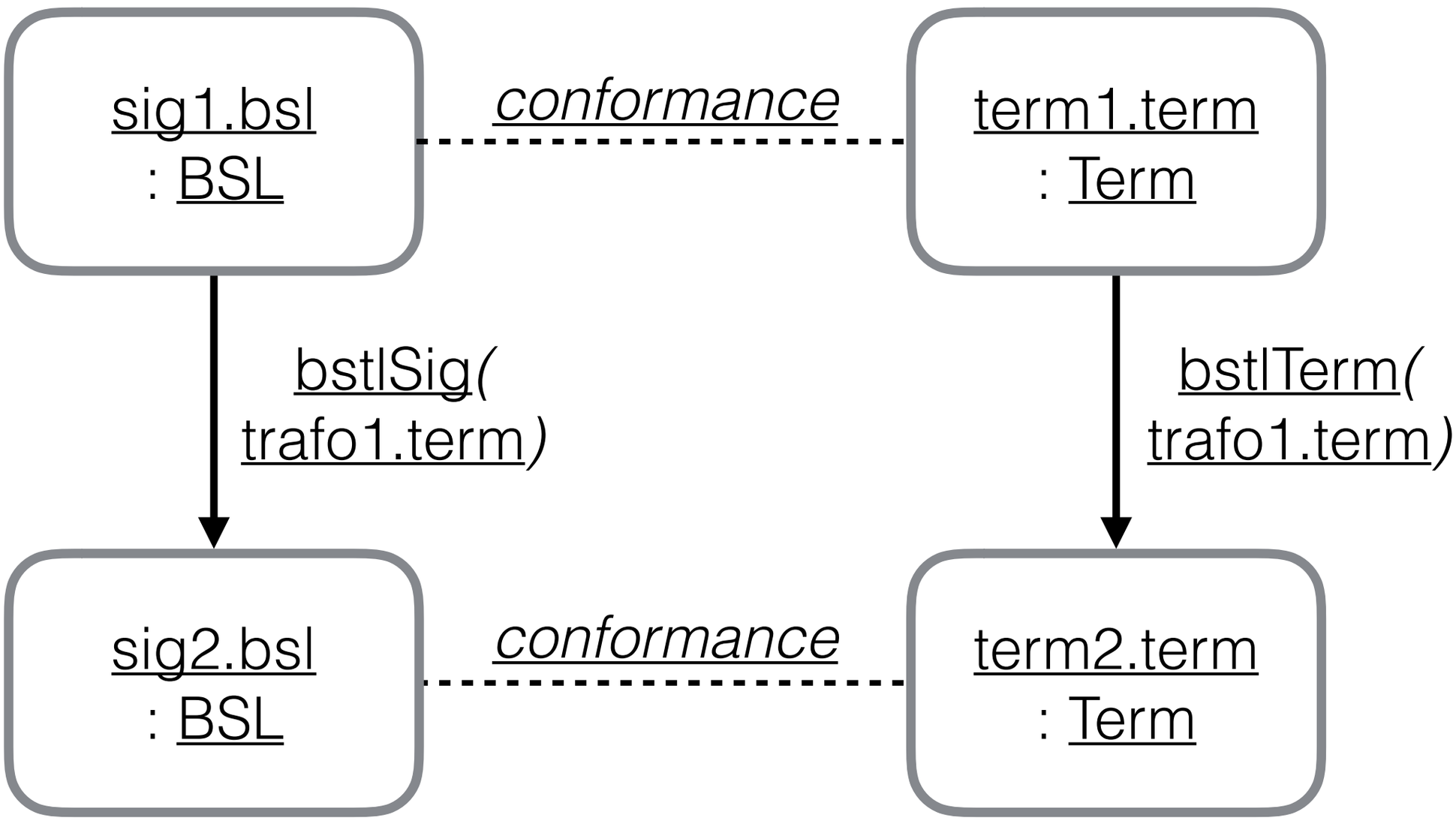
We only permit the subset of EGTL which serves language extension. See here.

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

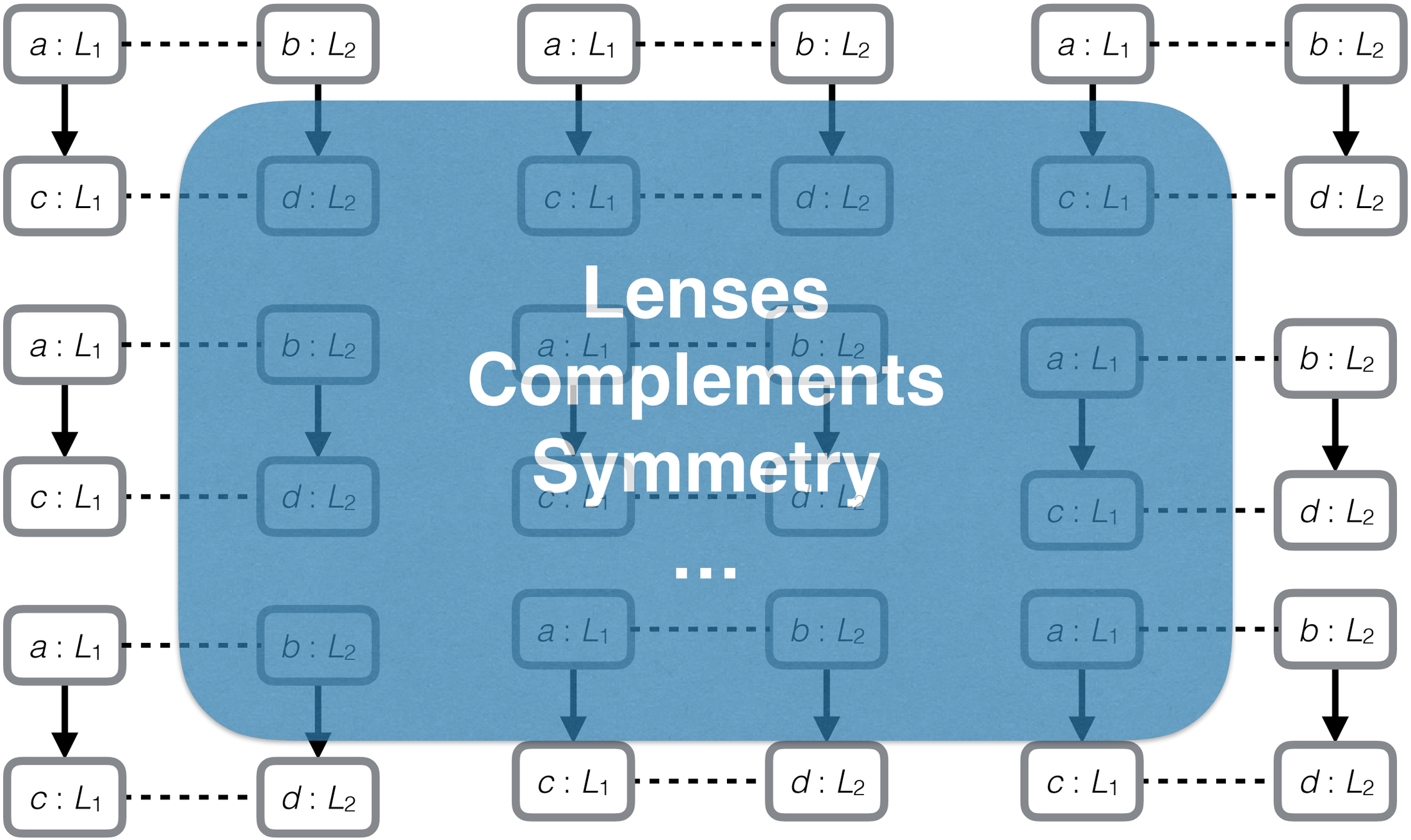


An ‘instance’ of CX by *co-transformation*

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

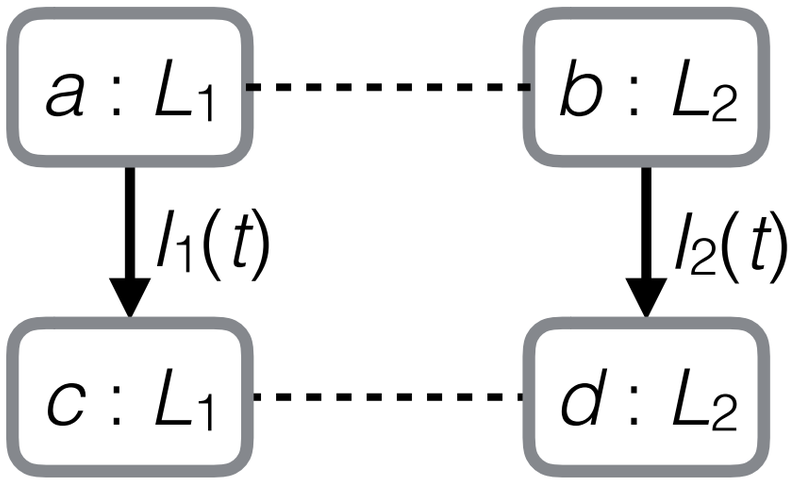


More CX



...

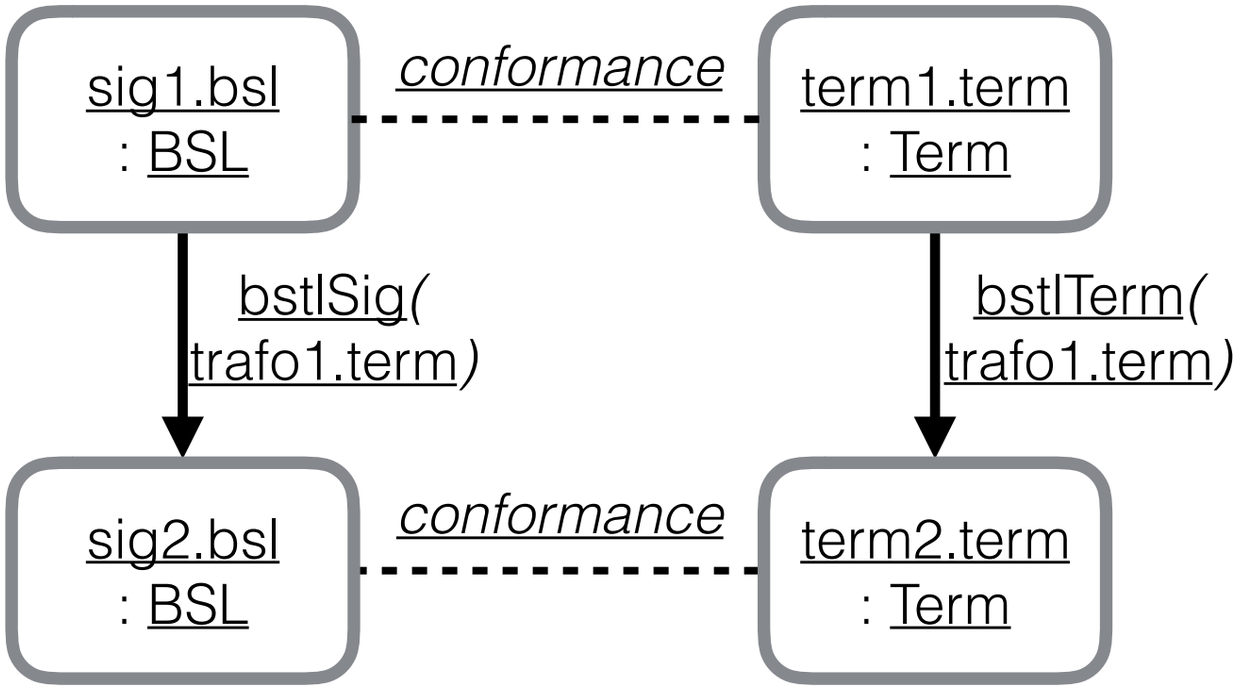
Higher level megamodel for CX by *co-transformation*



LAL megamodel <u>cx.cotransformation</u>

```
reuse coupling
reuse interpretation [ L2 ↦ L1, Any2 ↦ Any1 ]
reuse interpretation [ L1 ↦ L2, Any1 ↦ Any2 ]
axiom consistency { ∀ t ∈ XL. ∀ a, c ∈ L1. ∀ b, d ∈ L2.
  consistent(a, b)
  ∧ interpret(t, a) = c
  ∧ interpret(t, b) = d ⇒ 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']) ].
```

Megamodel compilation for CX by *co-transformation*

LAL megamodel

```
reuse coupling
reuse interpretation [ L2 ↦ L1, Any2 ↦ Any1 ]
reuse interpretation [ L1 ↦ L2, Any1 ↦ Any2 ]
axiom consistency { ∀t ∈ XL. ∀a, c ∈ L1. ∀b, d ∈ L2.
  consistent(a, b)
  ∧ interpret(t, a) = c
  ∧ interpret(t, b) = d ⇒ consistent(c, d) }
```

Ueber megamodel

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

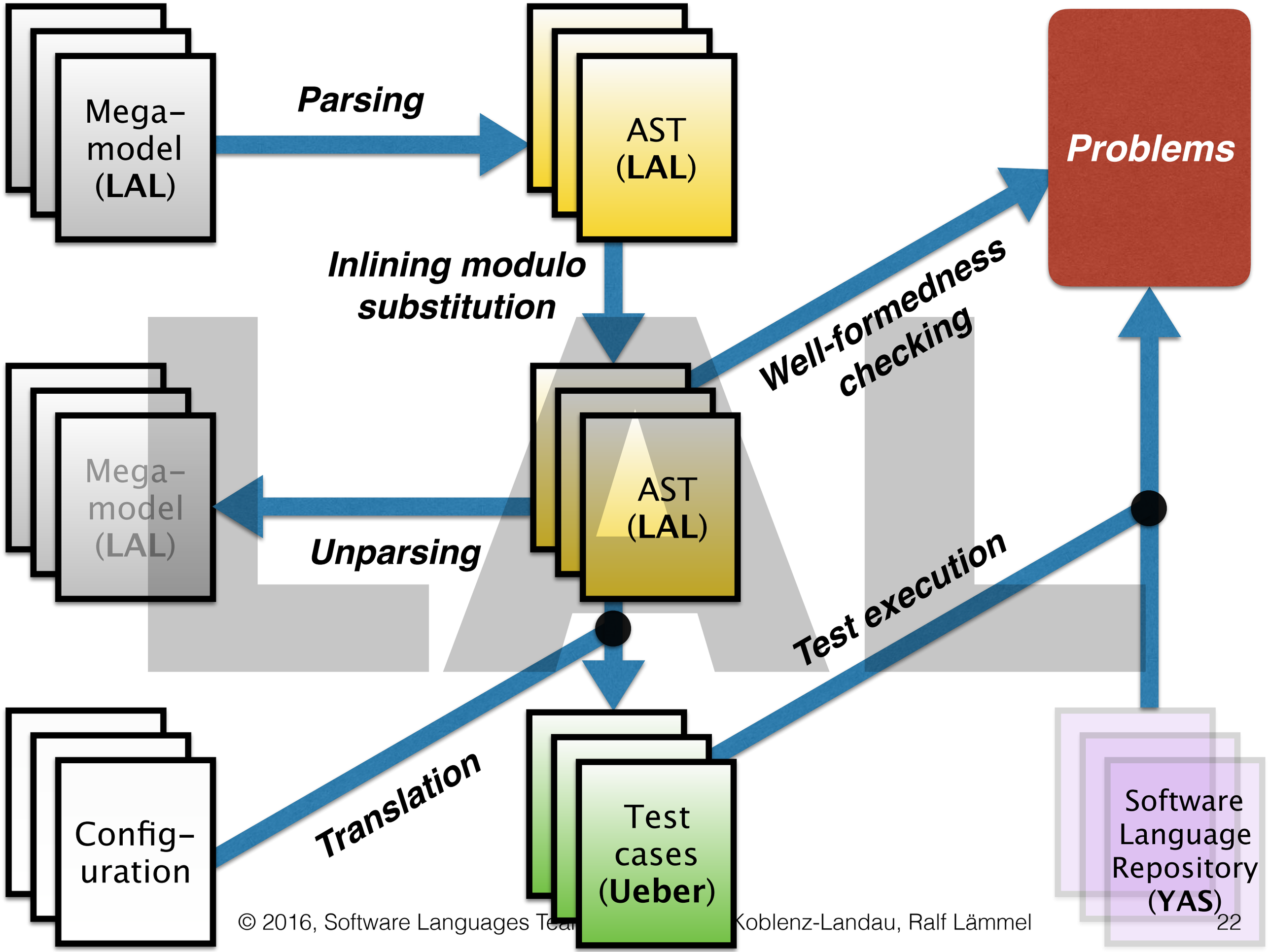
Configuration of compilation from higher to lower level megamodel

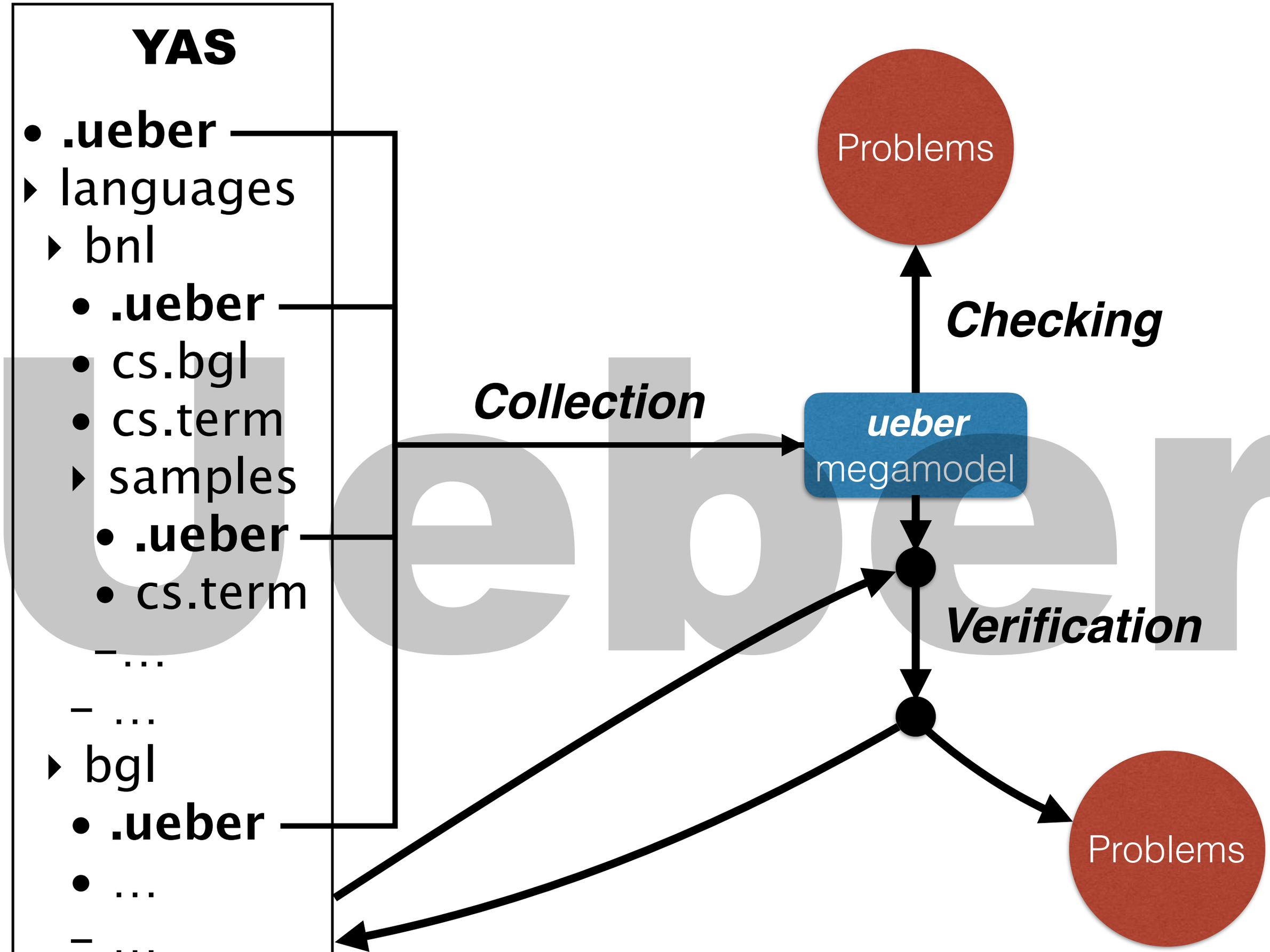
LAL configuration cx.cotransformation

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

Summary of megamodel compilation

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

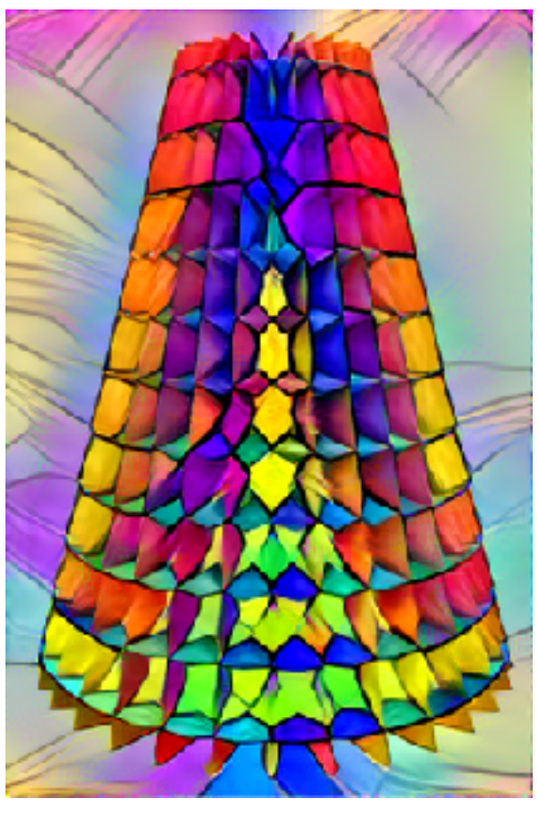

Mega-
model
(LAL)
Parsing
AST
(LAL)
Problems
Inlining modulo
substitution
Well-formedness
checking
Mega-
model
(LAL)
Unparsing
AST
(LAL)
Test execution
Translation
Config-
uration
Test
cases
(Ueber)
Software
Language
Repository
(YAS)


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

The basic skill set of software language engineering

Ralf Lämmel
Software Languages Team
University of Koblenz-Landau
Germany

See you tomorrow at SPLASH-I?



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

Ralf Lämmel: "***Software Languages. A slightly Haskell-biased introduction***".
Under consideration by Springer.
To appear in 2017.

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