SLEIR

Software Language Engineering by Intentional Rewriting

Vadim Zaytsev
Universiteit van Amsterdam
SQM 2014 @ CSMR-WCRE
3 February 2014

Who am I

- 2013–2014: Universiteit van Amsterdam
- 2010–2013: Centrum Wiskunde & Informatica
- 2008–2010: Universität Koblenz-Landau
- 2004–2008: Vrije Universiteit Amsterdam
- 2002–2004: Universiteit Twente
- 1998–2003: Rostov State University

http://grammarware.net

Who am I

- 2013–2014: ???
- 2010–2013: grammar manipulation
- 2008–2010: grammar transformation
- 2004–2008: grammar engineering
- 2002–2004: domain-specific languages
- 1998–2003: programming languages

http://grammarware.net

What is my dream

- Verify claims about software language engineering
- Automate what can be (semi)automated
- e.g.:
    - N. Wirth. On the Design of Programming Languages. In IFIP Congress. Pp. 386–393. 1974.

What is my story now

- Grammars = rewriting systems
    - (kind of) “in a broad sense”
- Grammar transformations = rewriting grammars
- Making grammar mutation suite
    - = rewriting grammar transformation operators

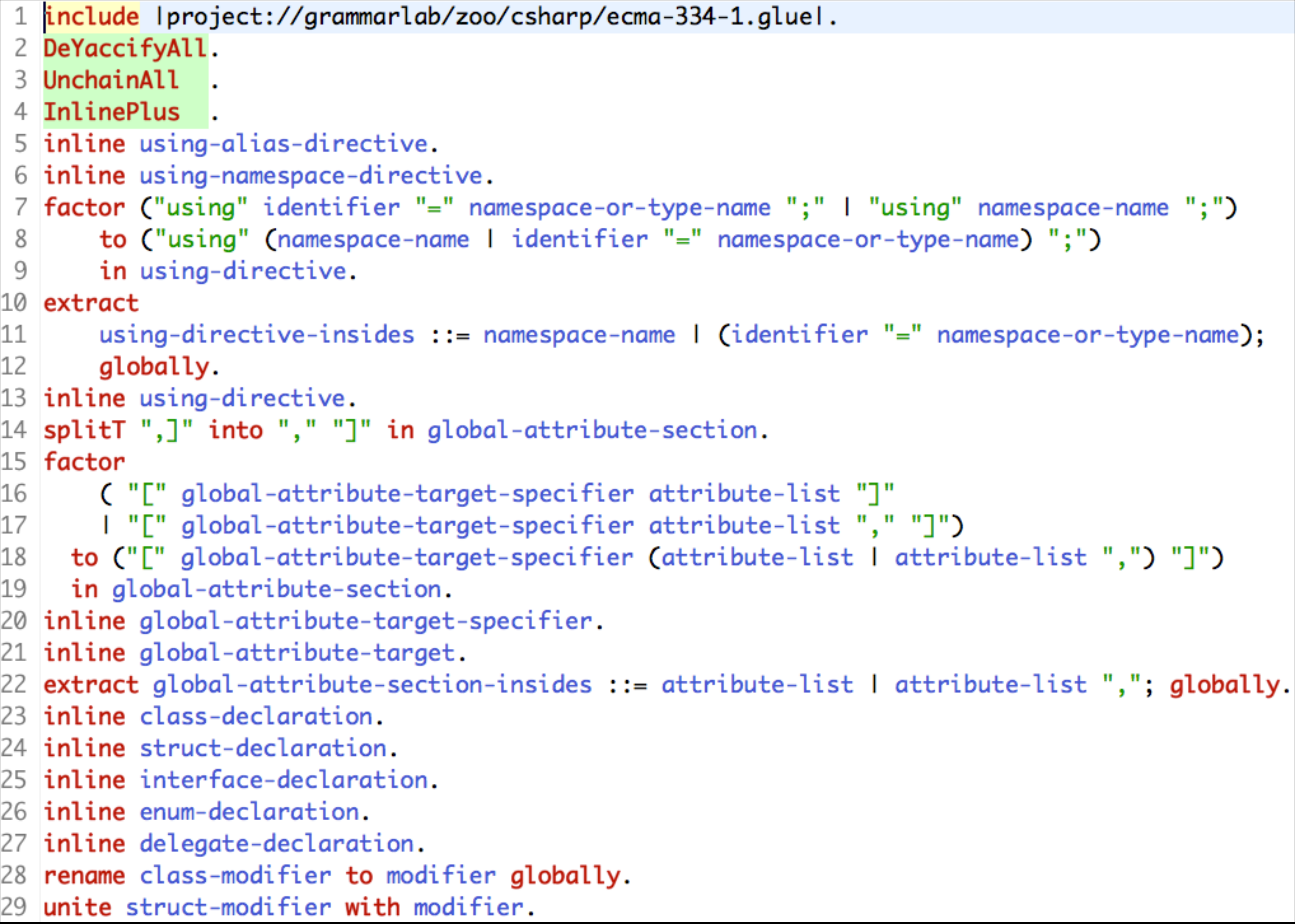
Automated SLE

- We have a software language X
- We want another software language Y
- We know how they relate to each other
- We wish to infer Y from X
  - automate as much as we can

GRAMMARLAB

- Library for Rascal language workbench
- Based on several years of published research and several years of hacking in SLPS (Rascal, Prolog, Python, Haskell, XSLT, ...)
- Made mostly at CWI (Centrum Wiskunde & Informatica)
- Also presented as a tutorial at MoDELS 2013

http://grammarware.github.com/lab



```
include |project://grammarlab/zoo/csharp/ecma-334-1.glue|.
DeYaccifyAll.
UnchainAll  .
InlinePlus  .
inline using-alias-directive.
inline using-namespace-directive.
factor ("using" identifier "=" namespace-or-type-name ";" | "using" namespace-name ";")
    to ("using" (namespace-name | identifier "=" namespace-or-type-name) ";")
    in using-directive.
extract
    using-directive-insides ::= namespace-name | (identifier "=" namespace-or-type-name);
    globally.
inline using-directive.
splitT ",]" into "," "]" in global-attribute-section.
factor
    ( "[" global-attribute-target-specifier attribute-list "]"
    | "[" global-attribute-target-specifier attribute-list "," "]")
  to ("[" global-attribute-target-specifier (attribute-list | attribute-list ",") "]")
  in global-attribute-section.
inline global-attribute-target-specifier.
inline global-attribute-target.
extract global-attribute-section-insides ::= attribute-list | attribute-list ","; globally.
inline class-declaration.
inline struct-declaration.
inline interface-declaration.
inline enum-declaration.
inline delegate-declaration.
rename class-modifier to modifier globally.
unite struct-modifier with modifier.
```

Grammar in a broad sense

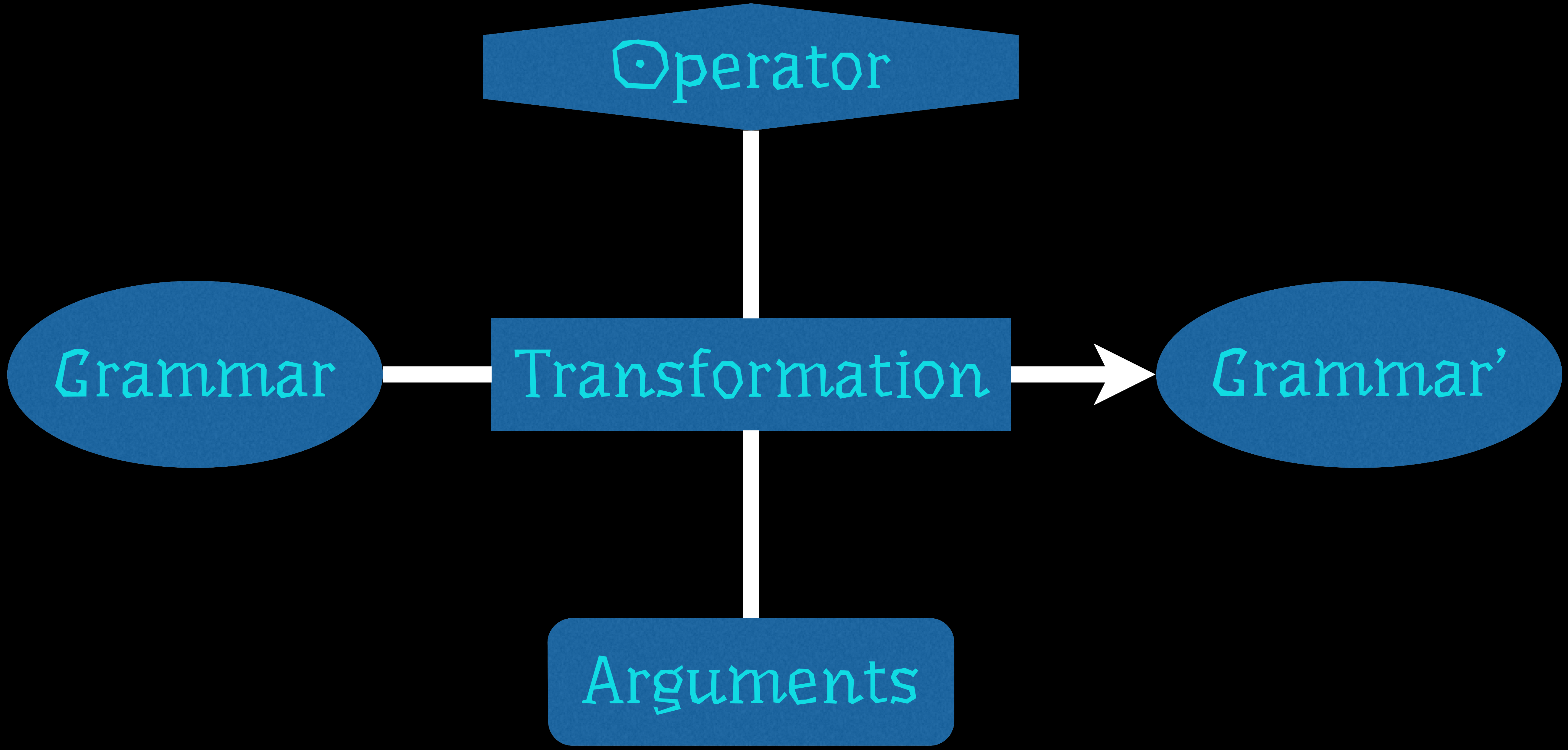
- Nonterminal
  - syntactic category
  - class
  - entity
  - type
  - …

- Terminal
  - atomic symbol
- Repetition
  - "one or more"
  - "zero or more"
  - "zero or one"

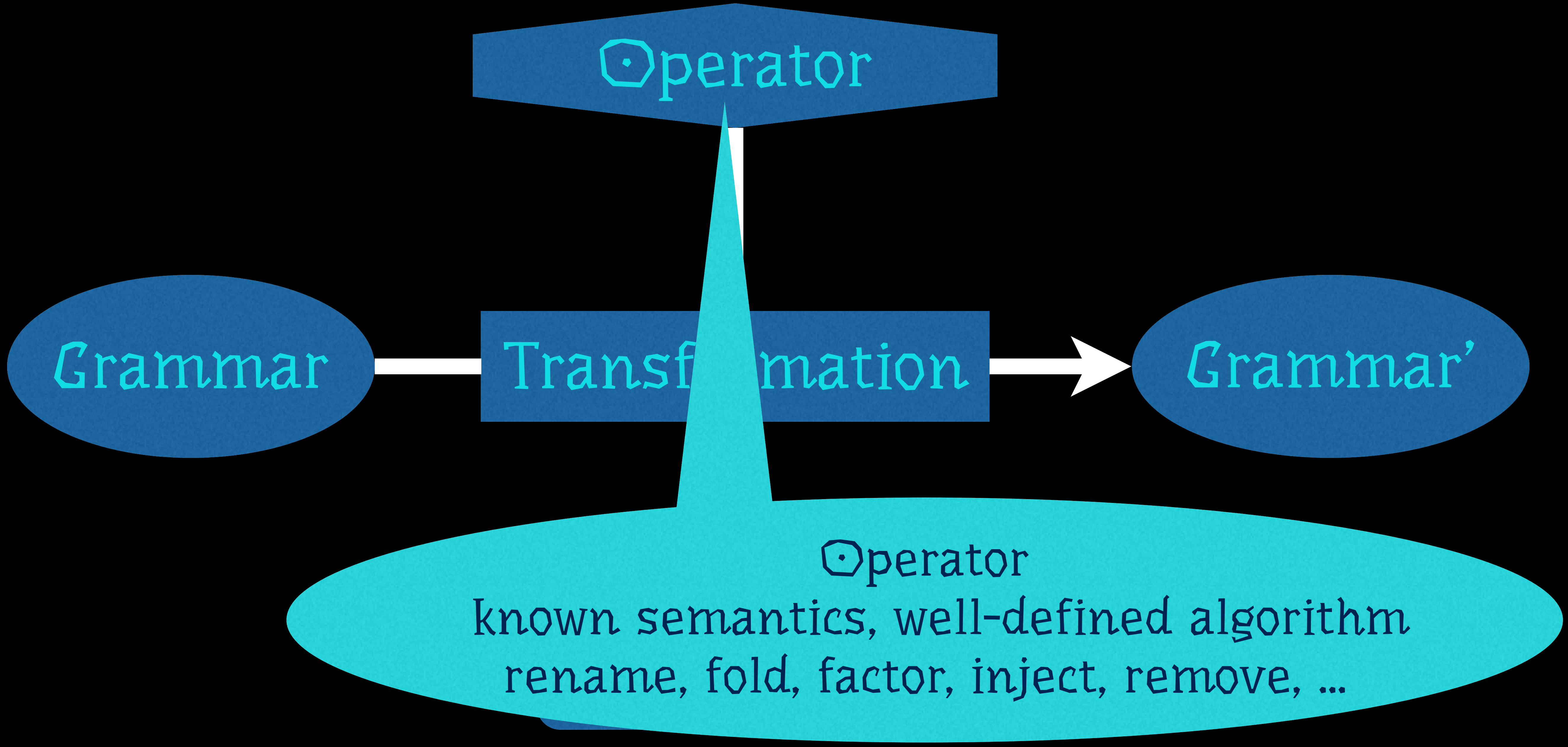
Grammar in a broad sense

- Label
  - named reference
  - node name
  - XML element
  - production label
- Mark
  - possibly named subexpr
  - purely decorative
  - line number
  - lightweight annotation

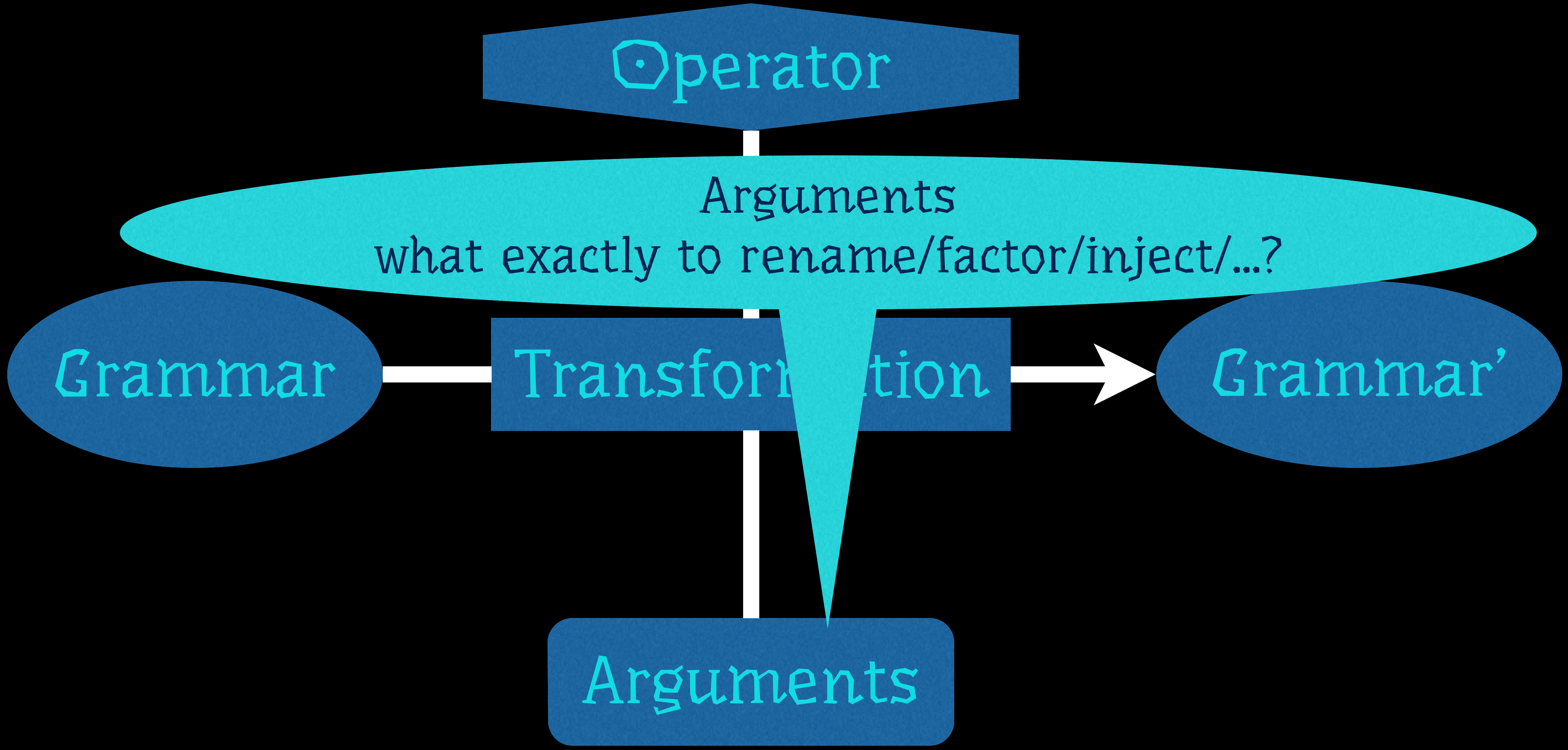
Grammar transformation



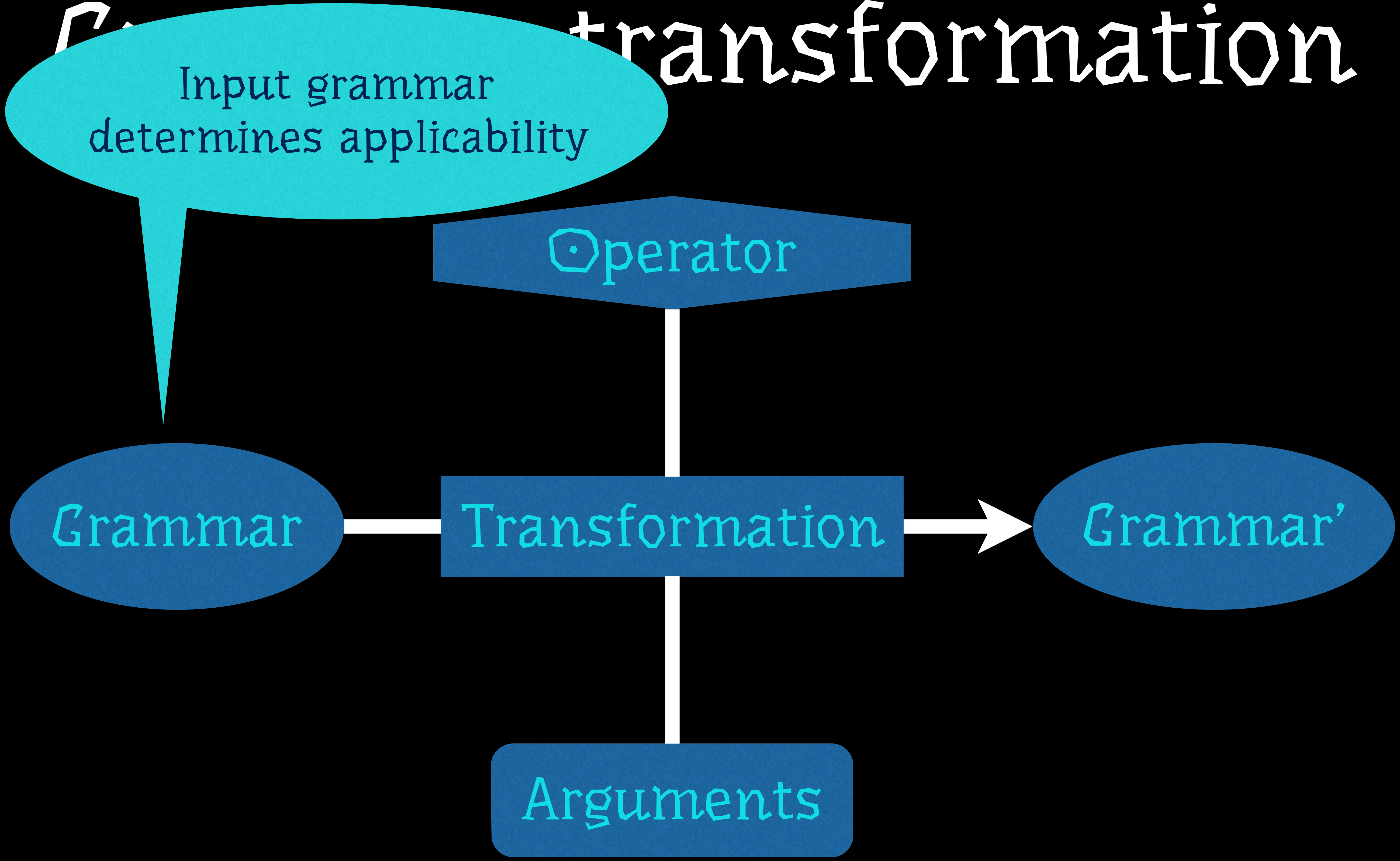
Grammar transformation


Operator
Grammar
Transformation
Grammar'
Operator
known semantics, well-defined algorithm
rename, fold, factor, inject, remove, ...

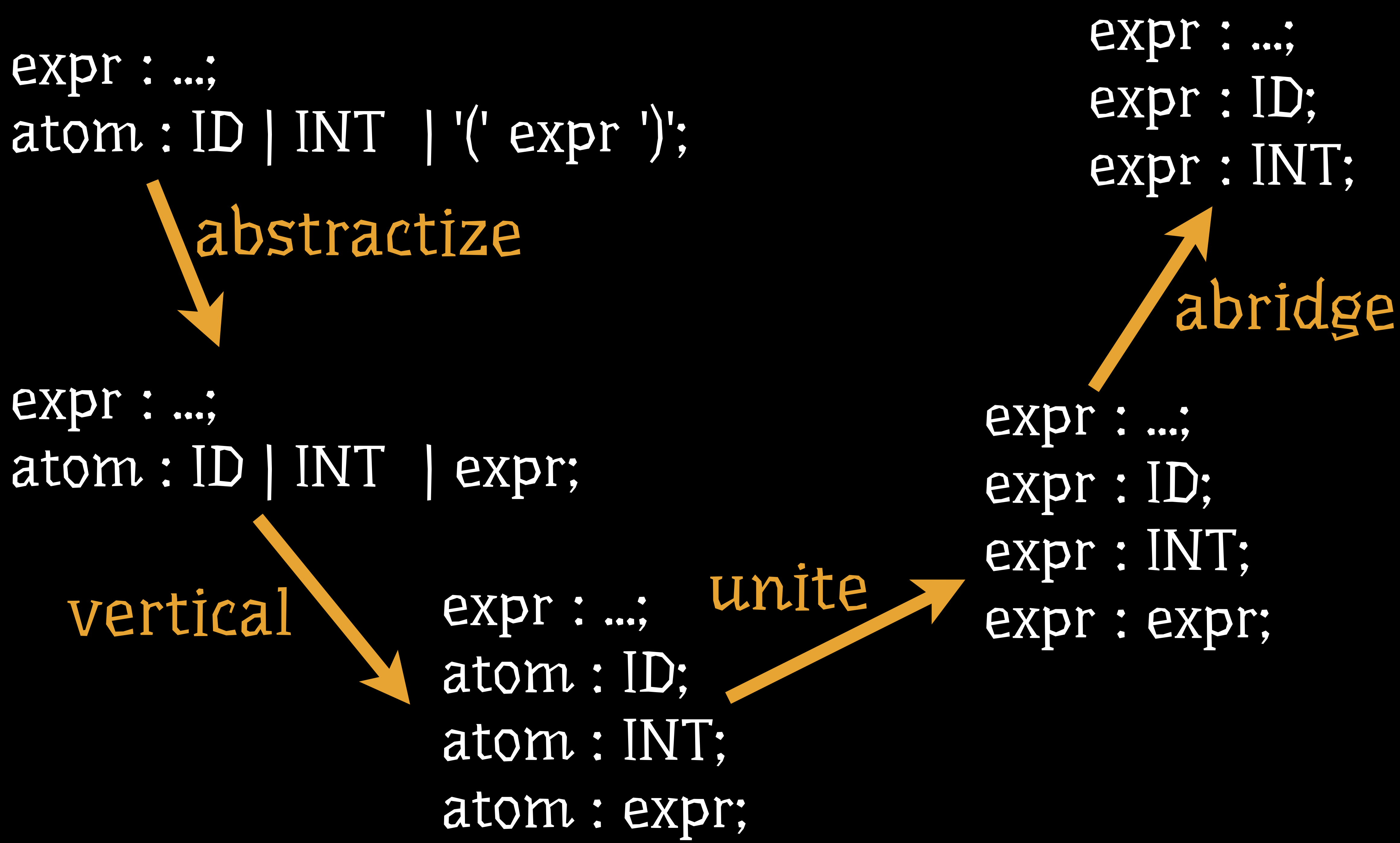
Grammar transformation



G transformation



Grammar transformation



```
expr : ...;
atom : ID | INT  | '(' expr ')';
```

```
expr : ...;
atom : ID | INT  | expr;
```

```
expr : ...;
atom : ID;
atom : INT;
atom : expr;
```

```
expr : ...;
expr : ID;
expr : INT;
expr : expr;
```

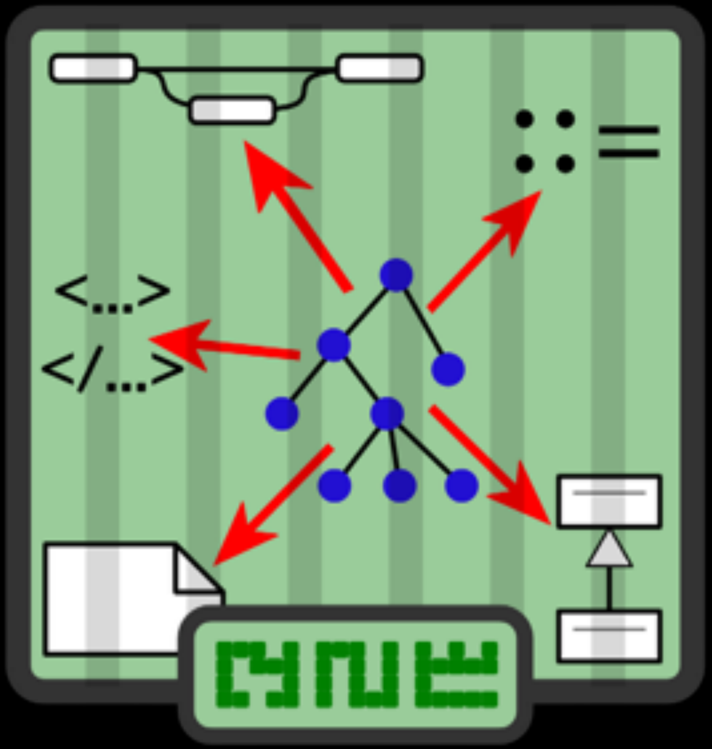
```
expr : ...;
expr : ID;
expr : INT;
```

R. Lämmel, V. Zaytsev, An Introduction to Grammar Convergence. IFM 2009, LNCS 5423.

Grammar programming

- As opposed to "grammar hacking"
- Grammar maintenance
  - corrective (fix "bugs" & problems)
  - adaptive (convergence & comparison)
  - perfective (new versions & dialects)
- Documents exact steps and their intent

Grammar Zoo



http://slps.github.io/zoo

- Language documentation
  - ISO, ECMA, W3C, OMG
- Document schemata
  - XSD, RELAX NG, Ecore
- Concrete syntax specs
  - Rascal library
  - SDF library
  - TXL library
  - ANTLR library
- Coursework
  - TESCOL, FL
- Versioning system
  - BGF, XBGF, EDD, LCF, LDF, XLDF
- Metamodels
  - entire AtlantEcore Zoo
- Other collections
  - books; test suites
  - mining
  - hunting
  - crawling
- ... [open] ...

Typical maintenance tasks

- Correct an error
- Collect metrics
- Claim equivalence
- Convert to a normal form / metalanguage
- Clean up technological idiosyncrasies
- Change a naming convention

Typical maintenance tasks

- Correct an error

  Lämmel, Zaytsev. Recovering Grammar Relationships for the Java Language Specification, SQJ, 2011.

- Collect metrics

  Power, Malloy. A Metrics Suite for Grammar-based Software. JSME, 2004.

- Claim equivalence

  R. Lämmel, V. Zaytsev, An Introduction to Grammar Convergence. IFM 2009.

- Convert to a normal form / metalanguage

  Zaytsev. BNF WAS HERE: What Have We Done About the Unnecessary Diversity of Notation ..., SAC, 2012.

- Clean up technological idiosyncrasies

  Lämmel, Verhoef, Cracking the 500 Language Problem, IEEE Software, 2001.
  Lämmel, Verhoef, Semi-automatic Grammar Recovery, SP&E, 2001.

- Change a naming convention

Typical maintenance tasks

- Correct an error

  Lämmel, Zaytsev. Recovering Grammar Relationships for the Java Language Specification, SQJ, 2011.

- Collect metrics

  Power, Malloy. A Metrics Suite for Grammar-based Software. JSME, 2004.

- Claim equivalence

  R. Lämmel, V. Zaytsev, An Introduction to Grammar Convergence. IFM 2009.

- Convert to a normal form / metalanguage

  Zaytsev. BNF WAS HERE: What Have We Done About the Unnecessary Diversity of Notation ..., SAC, 2012.

- Clean up technological idiosyncrasies

  Lämmel, Verhoef, Cracking the 500 Language Problem, IEEE Software, 2001.
  Lämmel, Verhoef, Semi-automatic Grammar Recovery, SP&E, 2001.

- Change a naming convention

Grammar Mutations

- Uniform intentional transformations in a large scope
- Bidirectional mappings between grammars
- “Rename all … to …” instead of “rename X to Y”
- Can generate transformation steps
- Transformation operator: precondition + rewriting
- Mutation: trigger + rewriting

Zaytsev. Language Evolution, Metasyntactically, BX / EC-EASST, 2012.

Type I mutations

- Trivial generalisation
- Precondition holds? Fire a transformation!
- Examples
  - distribute $\vdash$ DistributeAll
  - eliminate $\vdash$ EliminateTop

Type II mutations

- Automated generalisation
- Find where precondition holds & transform!
- Examples
  - concatT ⊢ ConcatAllT
  - reroot ⊢ Reroot2top

Type III mutations

- Narrowed generalisation
- Find subcases of Type I or II
- Examples
    - factor ⊢ Distribute; Undistribute
    - permute ⊢ PermutePostfix2Infix (& 5 others)

Type IV mutations

- Parametric generalisation
- Focus transformation according to parameters
- Examples
  - eliminate $\vdash$ SubGrammar
  - unite $\vdash$ UniteBySuffix

Back to maintenance

- Grammar has no starting symbol?
  - Reroot2top (Type II)
- Need abstract syntax from concrete syntax?
  - RetireTs (Type II)
- Grammar slicing?
  - SubGrammar (Type IV)

Better Call Saul!

Back to maintenance

- Grammar productions written in old BNF style?
  - DeyaccifyAll (Type I)
- Change naming convention?
  - RenameAllNLower2Camel (Type III)
- Grammar in a “readable” style with lots of chains?
  - UnchainAll (Type I)
  - InlineLazy (Type II)
  - MassageOptPlus2Star (Type III)

Better Call Saul!

Conclusion

- A case study in automated software language engineering
- Grammar mutations
  - Type I: trivially generalisable
  - Type II: automatically generalisable
  - Type III: generalisable to narrow subcases
  - Type IV: parametrically generalisable
- Code currently being migrated to the GrammarLab repo on GitHub
- Underdog font by Sergey Steblina & Jovanny Lemonad
- Questions?