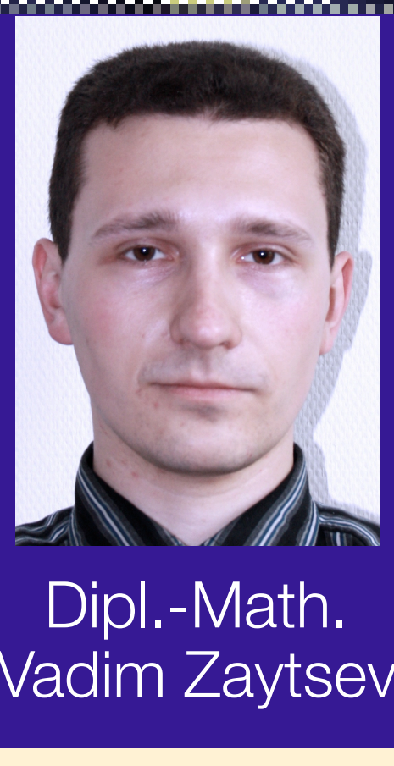
UNIVERSITÄT
KOBLENZ • LANDAU

Institut für Informatik
Software Languages Team

FB 4

Prof.Dr.
Ralf Lämmel



Dipl.-Math.
Vadim Zaytsev

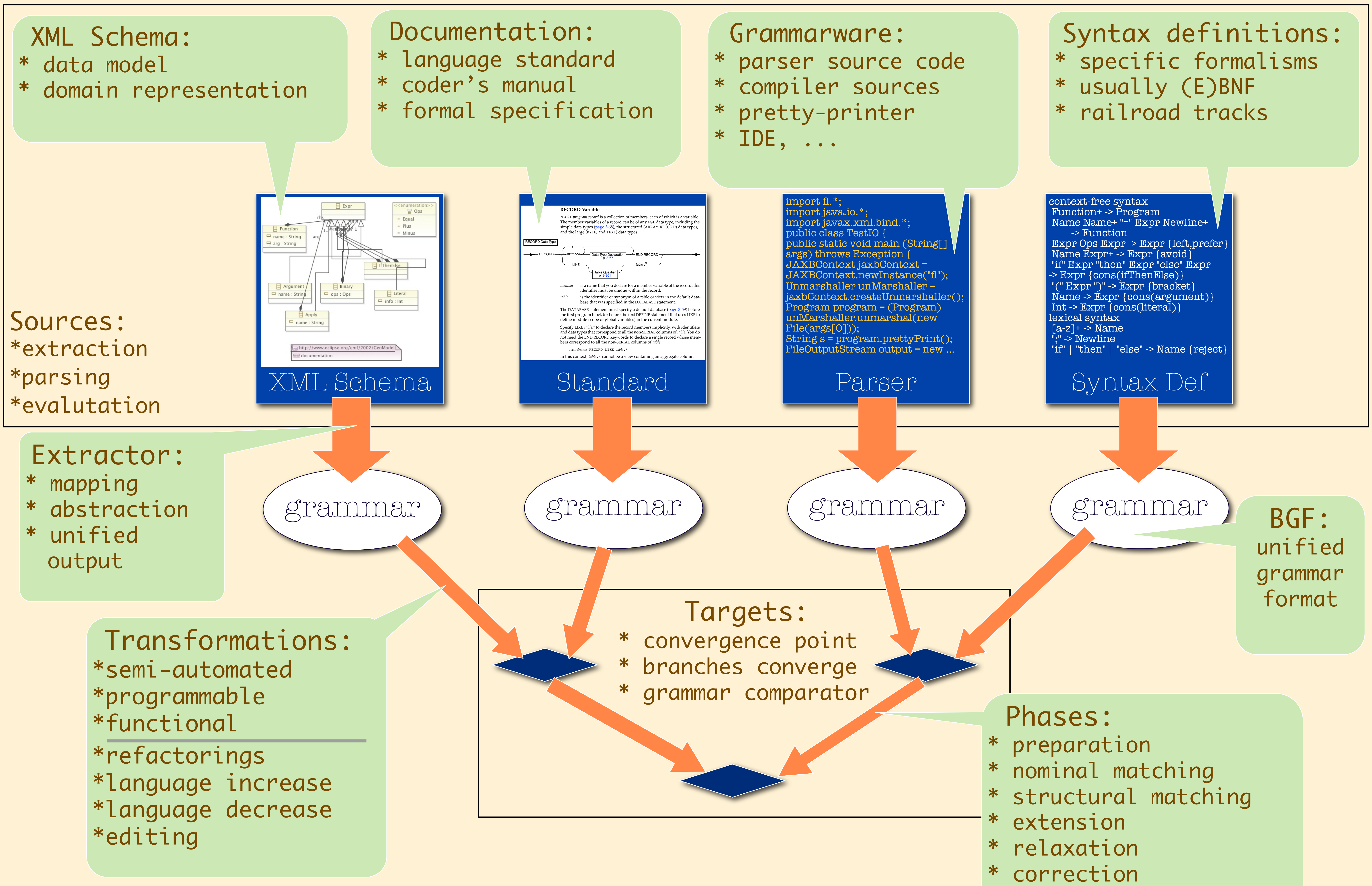
Semi-Automated Language Convergence

Grammar knowledge is ingrained in different kinds of software artifacts.

We want to establish and maintain correspondence among them.

Universal, not language-specific methodology is needed.

Grammar convergence is lightweight verification for mapping, binding, implementations, dialects, etc.



XML Schema:
- data model
- domain representation

Documentation:
- language standard
- coder's manual
- formal specification

Grammarware:
- parser source code
- compiler sources
- pretty-printer
- IDE, ...

Syntax definitions:
- specific formalisms
- usually (E)BNF
- railroad tracks

Sources:
*extraction
*parsing
*evalutation

```
import fl.*;
import java.io.*;
import javax.xml.bind.*;
public class TestIO {
public static void main (String[]
args) throws Exception {
JAXBContext jaxbContext =
JAXBContext.newInstance("fl");
Unmarshaller unMarshaller =
jaxbContext.createUnmarshaller();
Program program = (Program)
unMarshaller.unmarshal(new
File(args[0]));
String s = program.prettyPrint();
FileOutputStream output = new ...
```

```
context-free syntax
 Function+ -> Program
 Name Name+ "=" Expr Newline+
       -> Function
 Expr Ops Expr -> Expr {left,prefer}
 Name Expr+ -> Expr {avoid}
 "if" Expr "then" Expr "else" Expr
-> Expr {cons(ifThenElse)}
 "(" Expr ")" -> Expr {bracket}
 Name -> Expr {cons(argument)}
 Int -> Expr {cons(literal)}
lexical syntax
 [a-z]+ -> Name
 ";" -> Newline
 "if" | "then" | "else" -> Name {reject}
```

Extractor:
- mapping
- abstraction
- unified output

BGF:
unified grammar format

Transformations:
*semi-automated
*programmable
*functional

*refactorings
*language increase
*language decrease
*editing

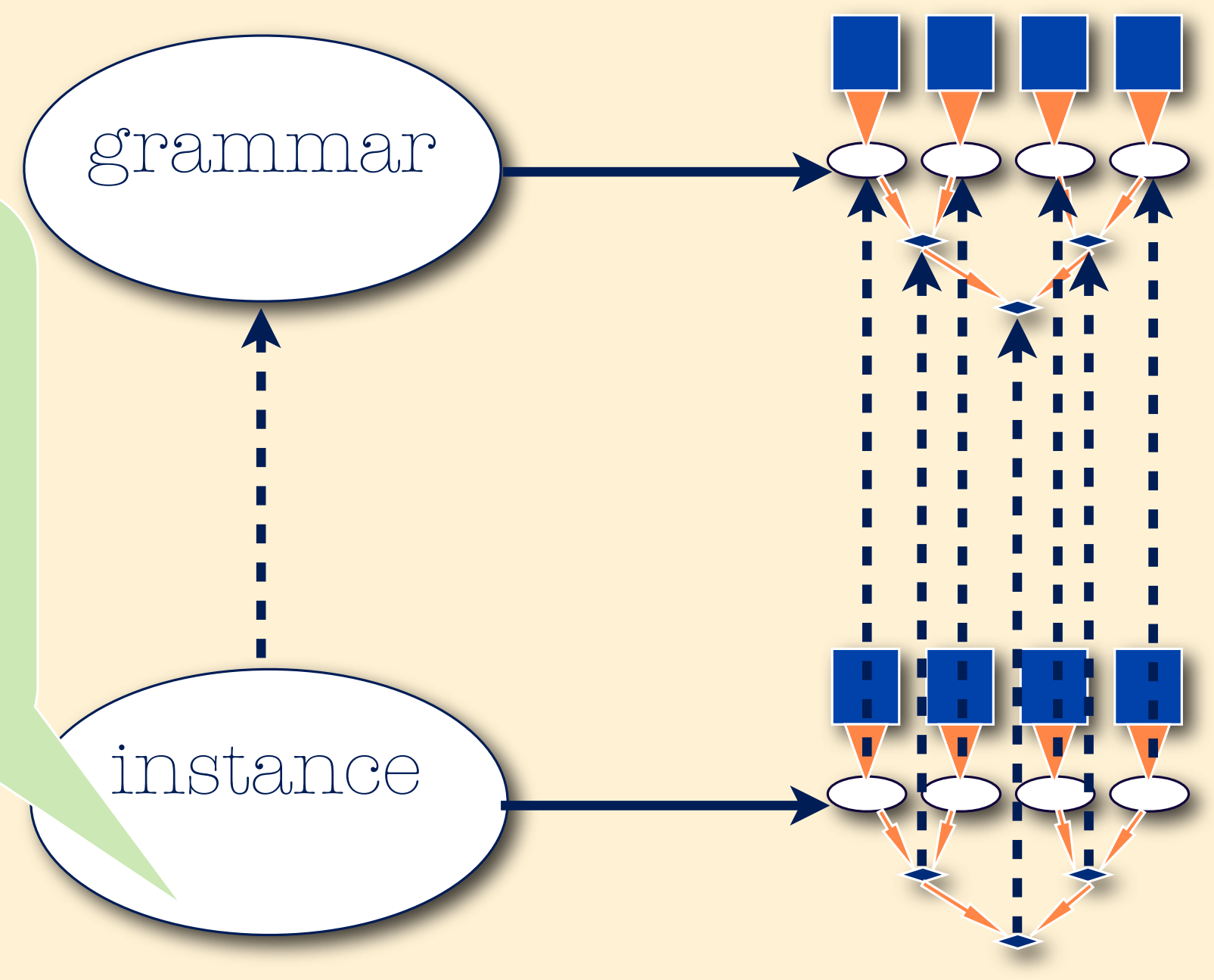
Targets:
- convergence point
- branches converge
- grammar comparator

Phases:
- preparation
- nominal matching
- structural matching
- extension
- relaxation
- correction

Related research topics for student projects:

- IDE support for interactive grammar transformation
- Full XML Schema support for grammar extraction
- Optimization of transformations by deforestation
- Proof of correctness for coupled transformations
- Model-based grammar comparison

Coupled transformations:
- repeat with instances
- use as test cases:
- parse & evaluate



grammar
instance