Grammar Convergence

reverse engineering relationships among grammar artifacts

XML Schema:
* data model
* domain representation

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

Grammarware:
* parser source code
* compiler sources
* IDE, ...

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

Sources:
*extraction
*parsing
*evaluation

XML Schema

Standard

Parser

Syntax Def

Extractor:
* mapping
* abstraction
* unified output

grammar **grammar** **grammar** **grammar**

unified grammar format

Transformations:
* semi-automated
* programmable
* functional
* refactorings
* language increase
* language decrease
* revise

Targets:
* convergence point
* branches converge
* grammar comparator

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

```
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}
```

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