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.

Sources:
- extraction
- parsing
- evaluation

Extractors:
- mapping
- abstraction
- unified output

Transformations:
- semi-automated
- programmable
- functional
- refactorings
- language increase
- language decrease
- editing

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

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

XML Schema:
- data model
- domain representation

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

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

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

BGF: unified grammar format