Tracing Abstractions through Generation

Software Transformation Systems Workshop,
GPCE 2004

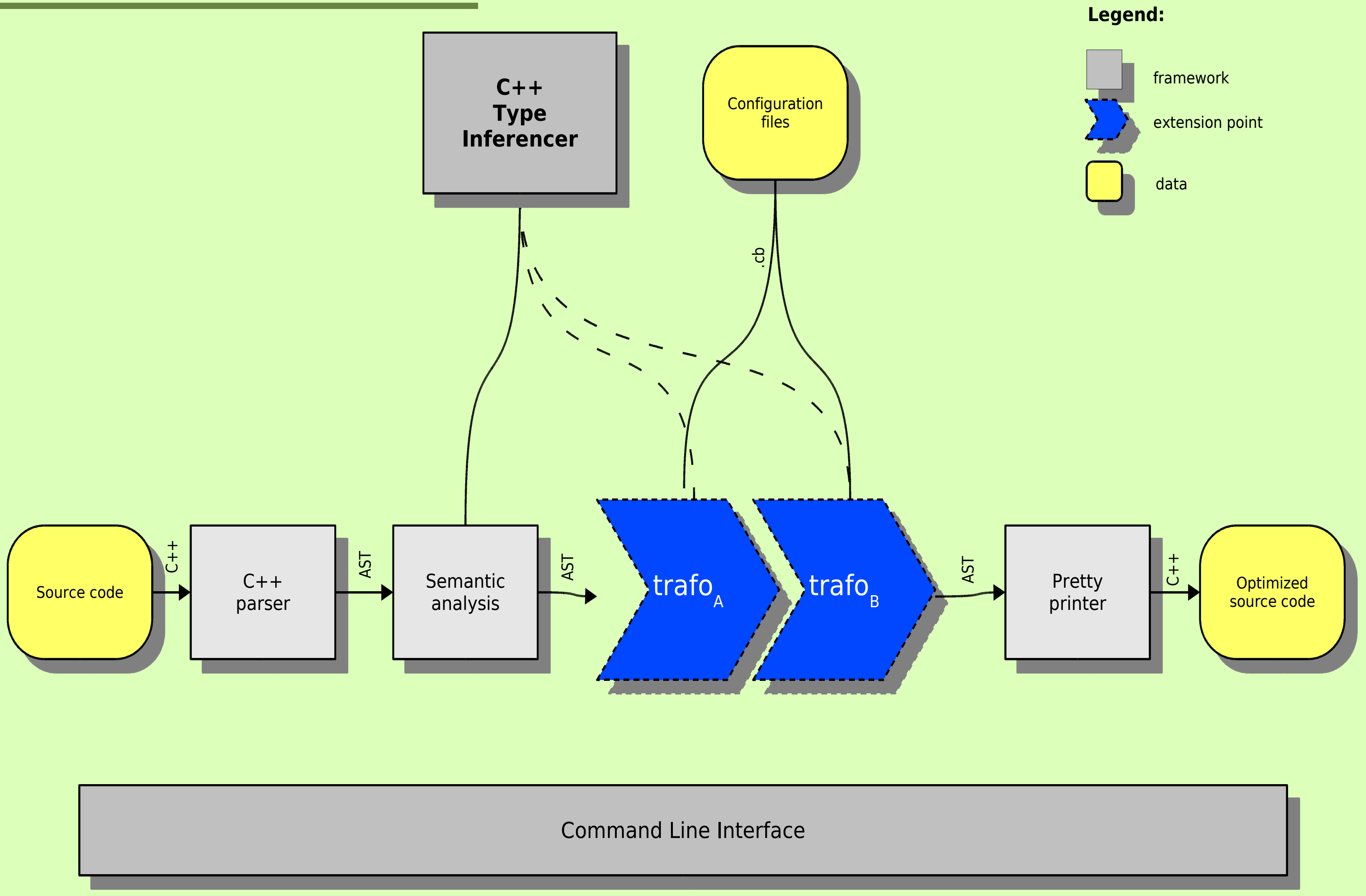
Karl Trygve Kalleberg[*]

University of Bergen
<karltk@ii.uib.no>

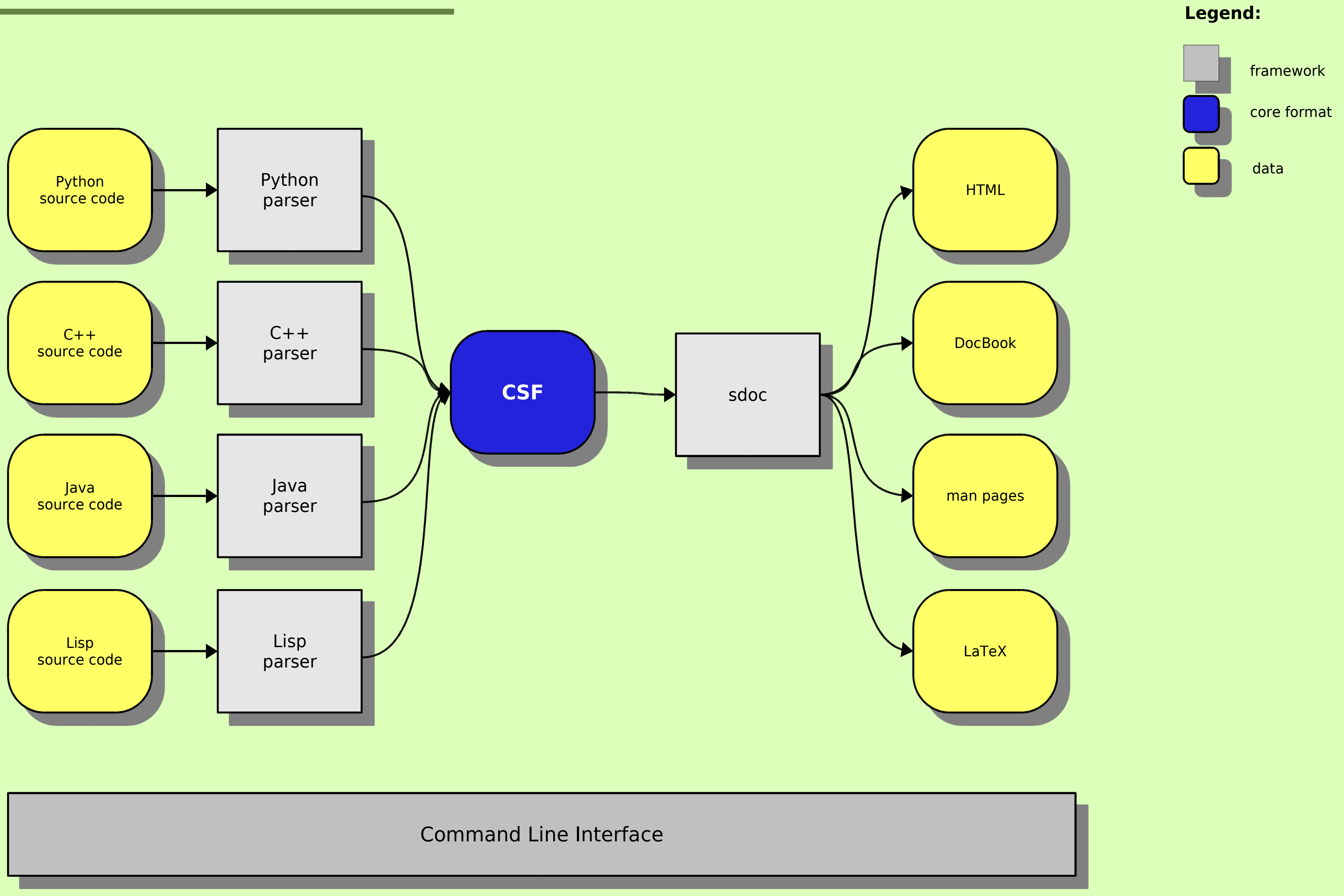
Utrecht University
<karltk@cs.uu.nl>

*) *Work funded by the Norwegian Research Council*

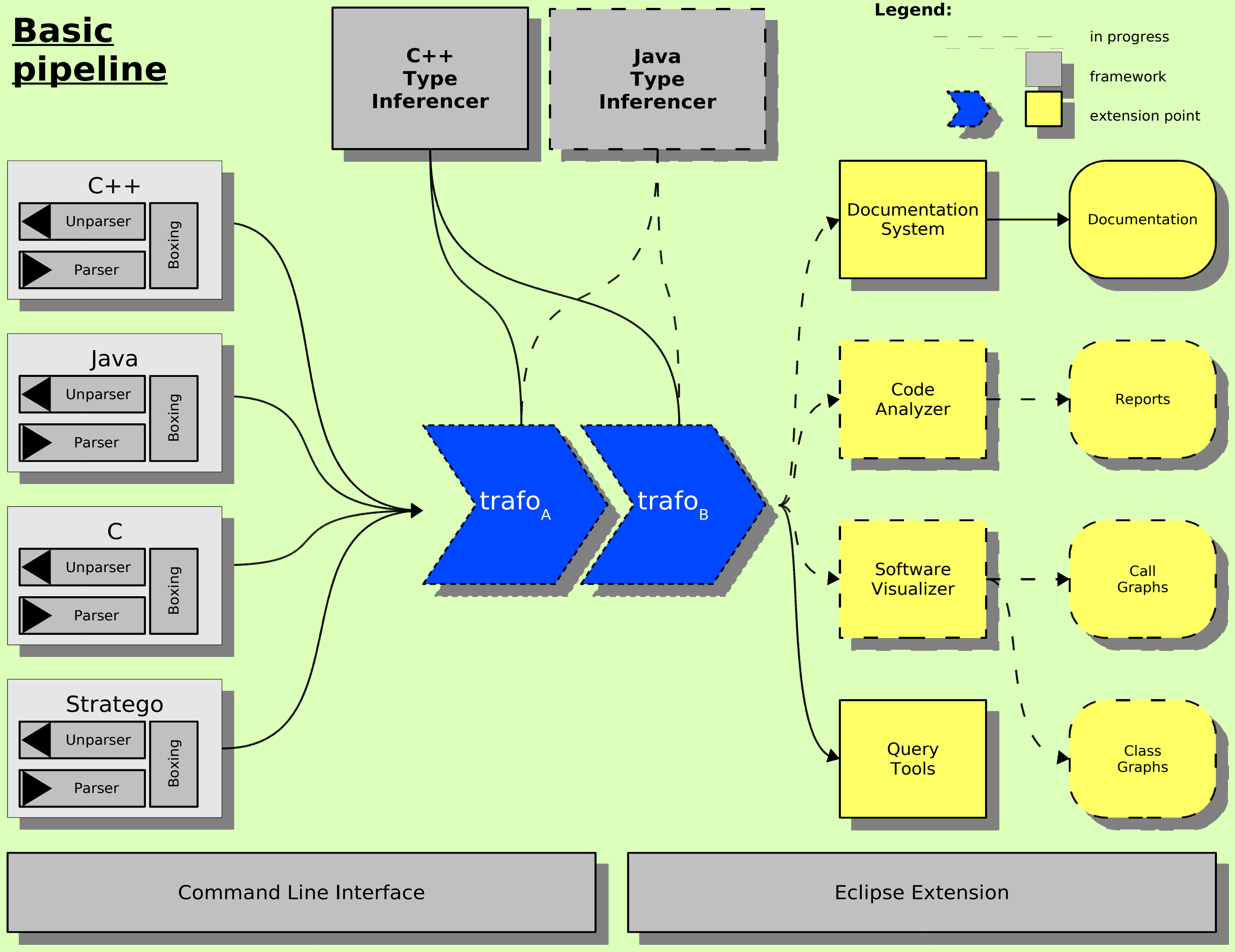
CodeBoost


Legend:
framework
extension point
data
C++ Type Inferencer
Configuration files
.cb
Source code
C++
C++ parser
AST
Semantic analysis
AST
trafo_A
trafo_B
AST
Pretty printer
C++
Optimized source code
Command Line Interface

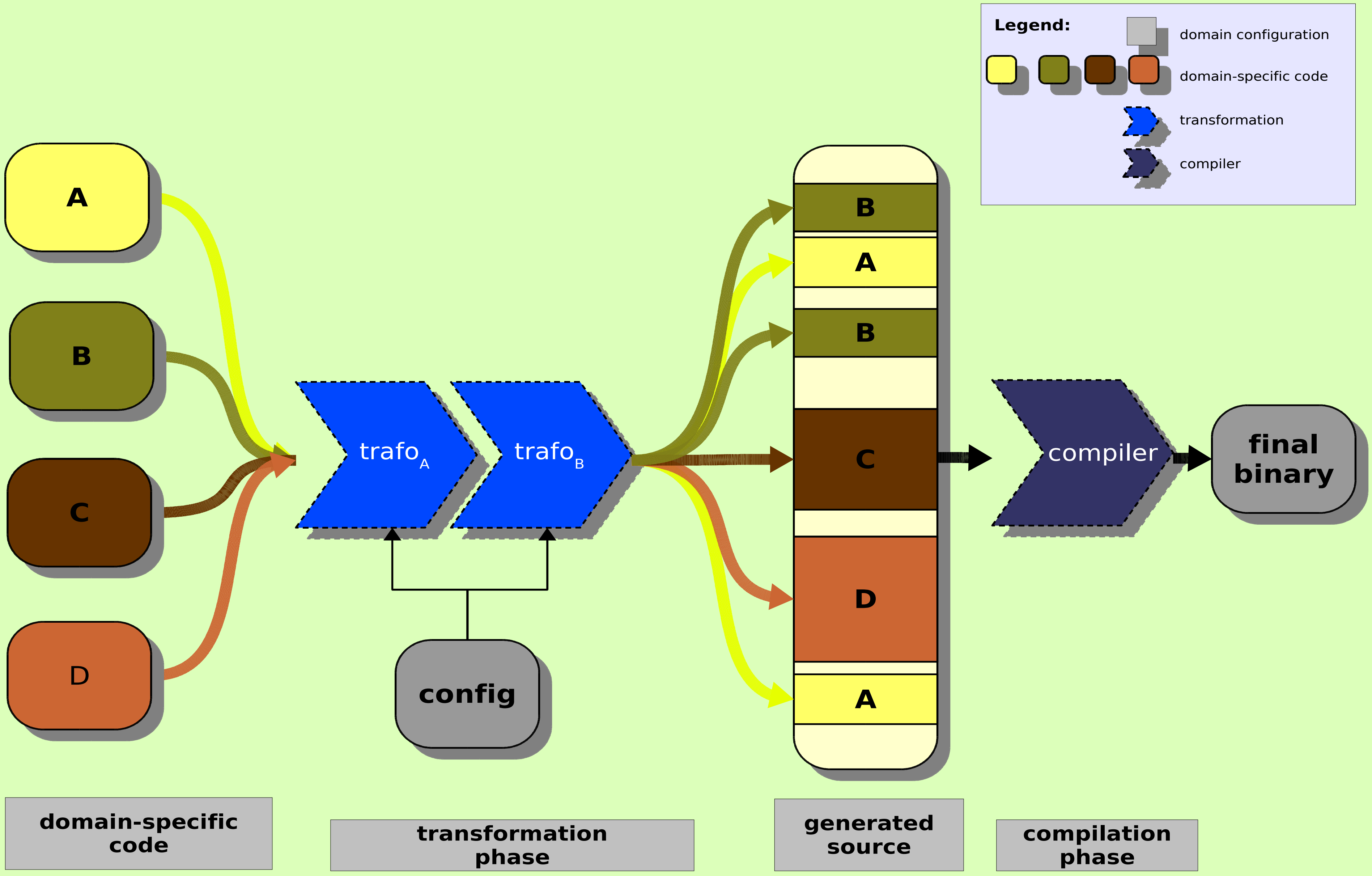
SDS: Software Development Foundation



Spoofax in-a-sketch



Problem description



Some sketched solutions

- Transformation phase
  - Use syntactically correct, semantically aware transformations
  - Support interactive replay
- Maintain trace of abstraction throughout complete pipeline
  - I.e. reverse arrows
- Compilation phase
  - Relate line numbers in generated source to domain abstractions
  - Problematic for glue code
- Deployment phase
  - Relate target abstractions to domain abstractions
- Runtime phase
  - Relate runtime exceptions to domain abstractions
- Debugging phase
  - Support language embedding

Concluding remarks

- Not fundamental research, so why bother?
  - Code generator pipelines seldom transparent
  - Another reason for people to write their own, custom transformation systems
  - Hampers productivity and happens often

- No obvious, established and employed techniques
- Can a “best-practice” be suggested?