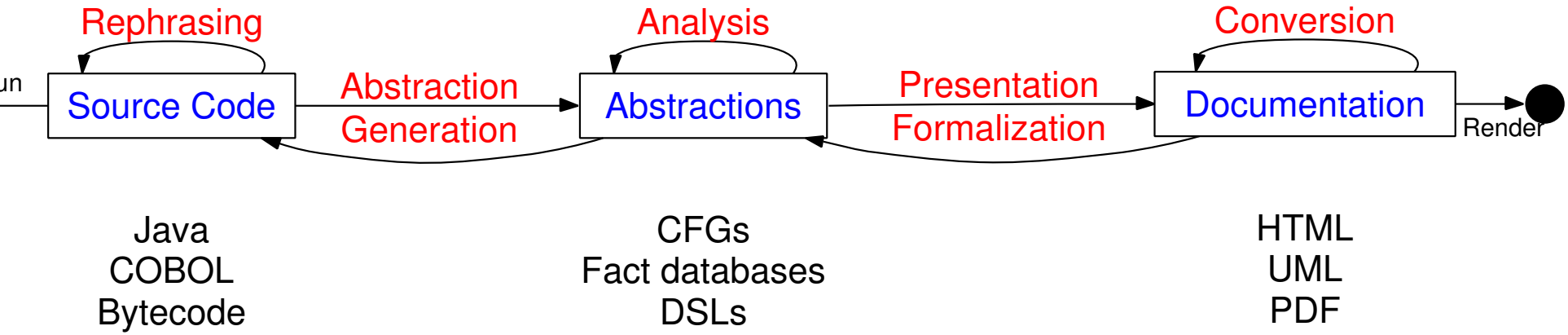
Combining Formalisms for Software Transformation

ASF+SDF Meta-Environment

Jurgen Vinju

STS — October 24, 2004

A sketch of the STS domain



- STSs support different engineering domains:

  Reverse engineering, MDA, Generative Programming, ...

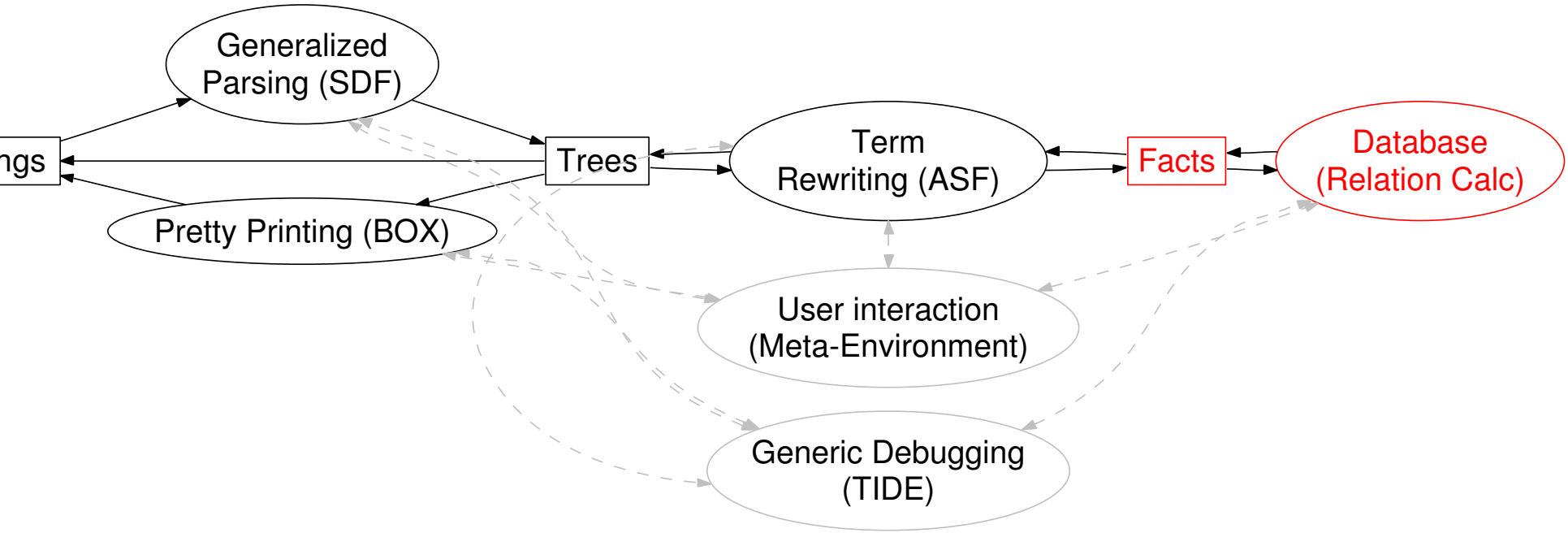
  All transitions between representations of source code

- Tools/libraries/DSLs to implement parts of such transitions:

  Parsing, Transformation, Relation Calculus, ...

- Open set of DSLs instead of a general purpose STS language.

Formalisms in the Meta-Environment



Nice features emerge by connecting formalisms, e.g:

ASF+SDF	syntax-safety, concrete syntax, layout+comment conservation
ASF+BOX	flexible (semantics directed) pretty printing
SDF+Meta	generic structure editing and syntax highlighting
ASF+TIDE	debugger-for-free, multilevel debugging
ASF+RelCalc	untangling complex analyses from tree structure

Question

- What about the connection between trafo and analysis formalisms?

  Either completely integrate (e.g. embracing Prolog), or ...

  Borrow features (e.g. add multiset datatype), or ...

  Keep a *loose coupling*, and formalize the interaction.

- How to connect rewriting tools to analysis tools:

  System: Type system integration & data marshalling

  User: Reusable Fact Extractions, and Design patterns for FEs.