# The basic skill set of software language engineering

Ralf Lämmel
Software Languages Team
University of Koblenz-Landau
Germany

This is sort of an advertisement for the upcoming textbook on software languages:

Ralf Lämmel: "***Software Languages. A slightly Haskell-biased introduction***".
Under consideration by Springer.
To appear in 2017.

http://www.softlang.org/book

# What’s to come today?

- An index-based story
- A profile based on languages
- A walk through the TOC
- The underlying repository

**Let’s run this a bit more like a discussion, not as a pure tutorial.**

# An index-based story

- abstract domain
- abstract interpretation
- **abstract syntax**
- abstraction
- **acceptance**
- **acceptor**
- Acme
- **algebraic signature**
- alpha conversion
- assignment
- axiomatic semantics
- **big-step …**
- **BNF**
- bottom
- compiler
- compiler optimization
- **compositionality**
- conclusion
- **concrete syntax**
- **context-free grammar**
- continuation
- CPS
- control flow
- data flow analysis
- **declarative language**
- denotational semantics
- derivation sequence
- derivation tree
- directional type
- domain model
- domain theory
- **DSL**
- **EBNF**
- embedded system
- **expression evaluation**
- external DSL
- **finite state machine**
- fixed point
- function
- function application
- function symbol
- **functional program**
- Galois connection
- goto
- **grammar**
- **Haskell**
- homomorphism
- imperative programming

- **inference rule**
- inlining
- internal DSL
- **interpreter**
- Java
- JavaScript
- judgment
- **language definition**
- lattice
- logic
- loop unrolling
- **many-sorted term**
- MDE
- meta-variable
- **metalevel**
- metametalevel
- method chaining
- model
- **modeling language**
- **nonterminal**
- normal form
- object model
- operational semantics
- **parse tree**
- **parser**
- **parsing**
- partial evaluation
- partial order
- Peano
- polymorphic function
- polymorphism
- pragmatics
- pre-term
- premise
- preservation
- **production**
- **profile**
- program analysis
- program specialization
- progress
- Prolog
- recursive function
- residual program
- **rule**
- semantic algebra

- semantic action
- **semantics**
- **signature**
- slicing
- **splice**
- **small-step …**
- sort
- start symbol
- statement execution
- substitution
- **syntax**
- **syntax checker**
- **syntax tree**
- System F
- term
- terminal
- text-to-model
- **text-to-object**
- **textual syntax**
- top
- transition system
- **type**
- type safety
- **type checking**
- type derivation
- type inference
- **type system**
- UML
- variable
- visual syntax
- **well-formedness**

# A profile based on languages

# Sample languages (syntax definition)

- BGL — Basic Grammar Language ('BNF')
- EGL — Extended Grammar Language ('EBNF')
- BSL — Basic Signature Language
- ESL — Extended Signature Language
- MML — Ecore-like Metamodeling Language
- DDL — SQL-like Data Definition Language

# Sample languages (GPLs, DSLs)

- BNL — Binary Number Language
- BTL — Basic TAPL Language
- FSML — FSM Language
- TBL — Tree-based Buddy Language
- GBL — Graph-based Buddy Language
- BFPL — Basic Functional Programming Language
- EFPL — Extended Functional Programming Language
- BIPL — Basic Imperative Programming Language
- EIPL — Extended Imperative Programming Language

# Implementation languages

- Haskell — for most (‘all’) illustrations
- ANTLR — whenever it works
- Python — for a few illustrations
- Java — for a few illustrations
- XML, XSD, JSON, JSON Schema — a bit
- Prolog — last chapter; do ‘everything’ again

# A walk through the TOC

# Software Languages

## A slightly Haskell-biased introduction

- Part I — Introduction
- Part II — Syntax and metaprogramming
- Part III — Semantics and types
- Part IV — Consolidation

# Software Languages

## A slightly Haskell-biased introduction

- **Part I — Introduction**
  - Preface
  - The notion of software language
  - Story of a language
- Part II — Syntax and metaprogramming
- Part III — Semantics and types
- Part IV — Consolidation

# Software Languages
## A slightly Haskell-biased introduction

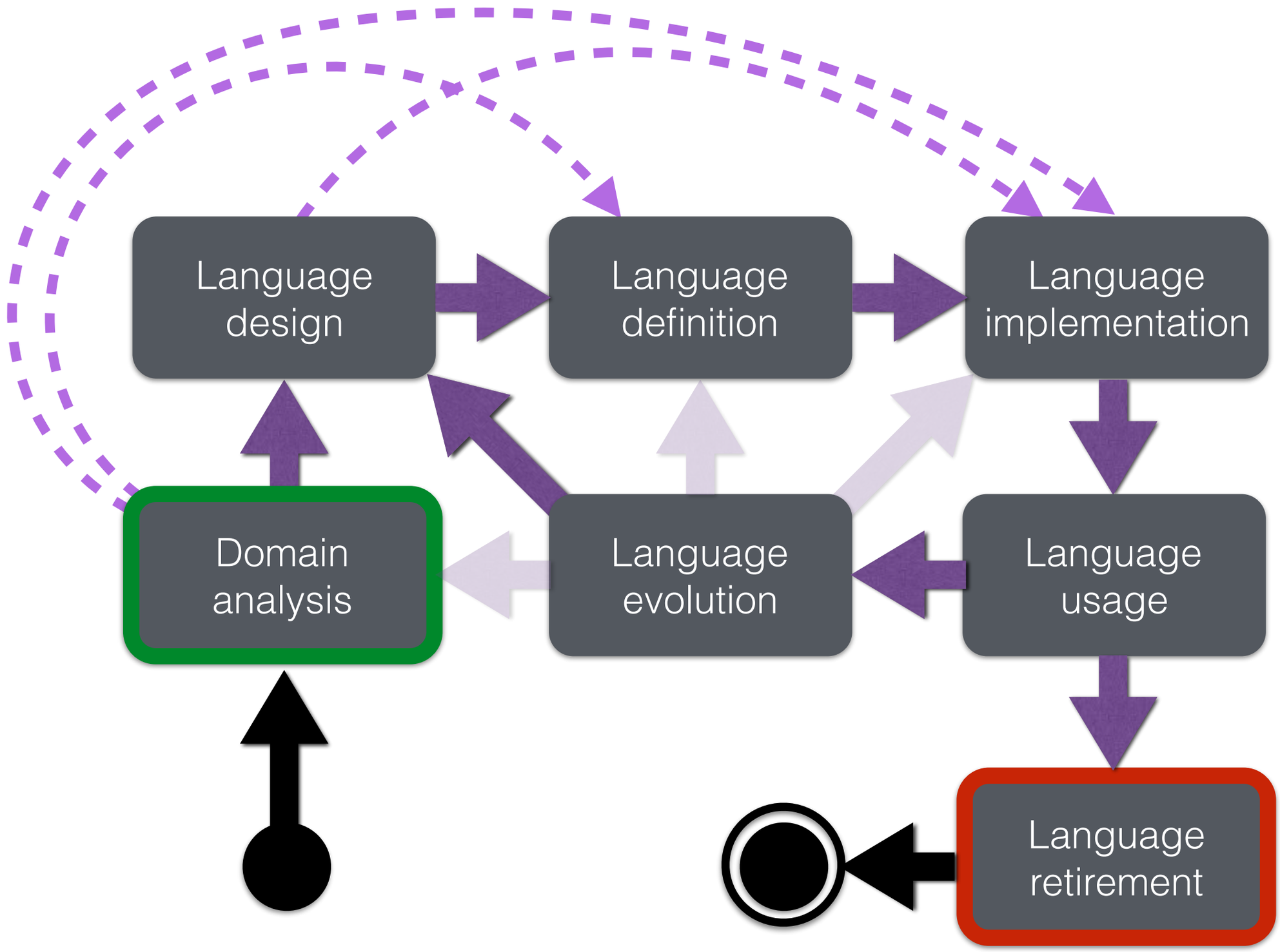
- Part I — Introduction
  - Preface
  - **The notion of software language**
    - Software language examples
    - Classification of software languages
    - The lifecycle of software languages
    - Software languages in software engineering
    - Software languages in academia
  - Story of a language

- Part II — Syntax and metaprogramming

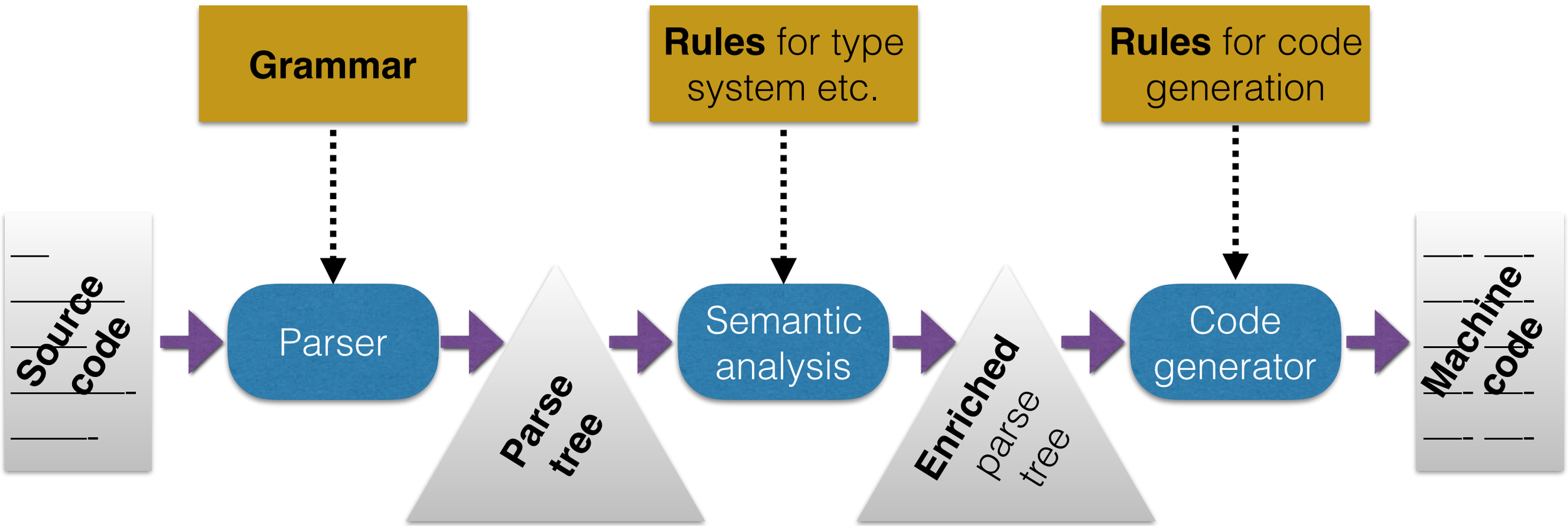
- Part III — Semantics and types

- Part IV — Consolidation

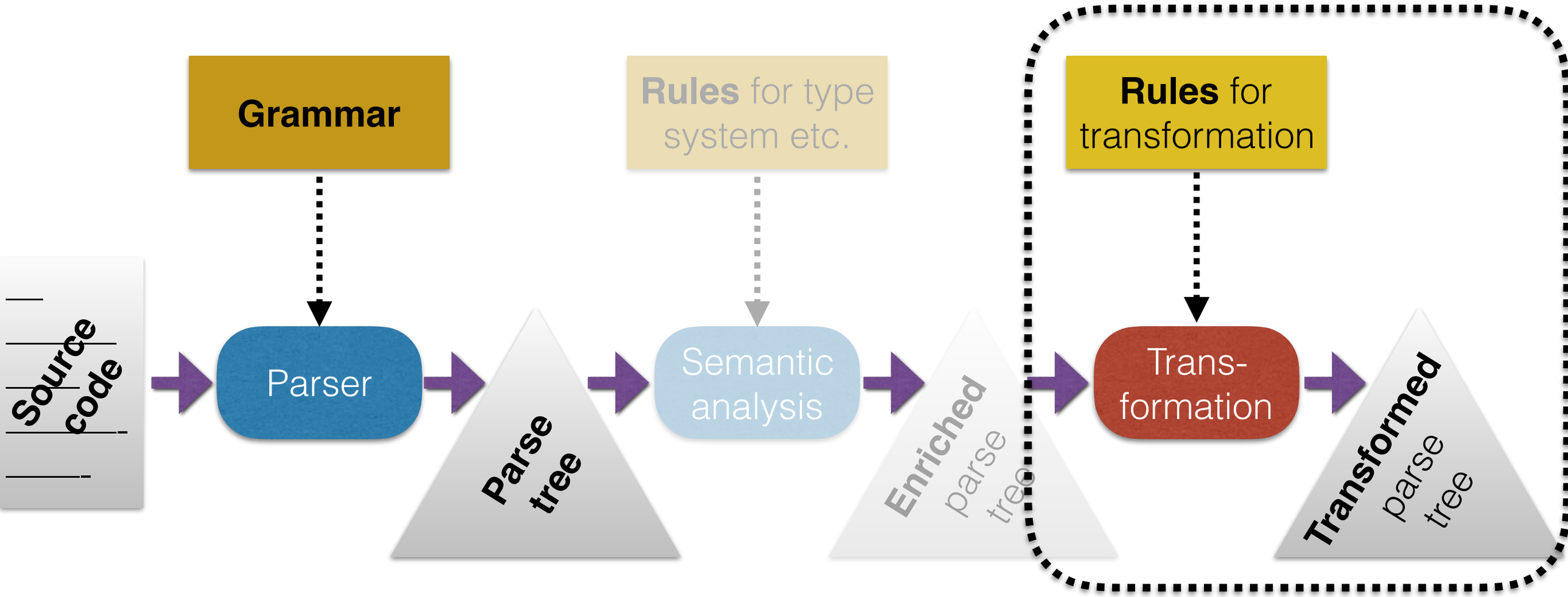
# The software language lifecycle



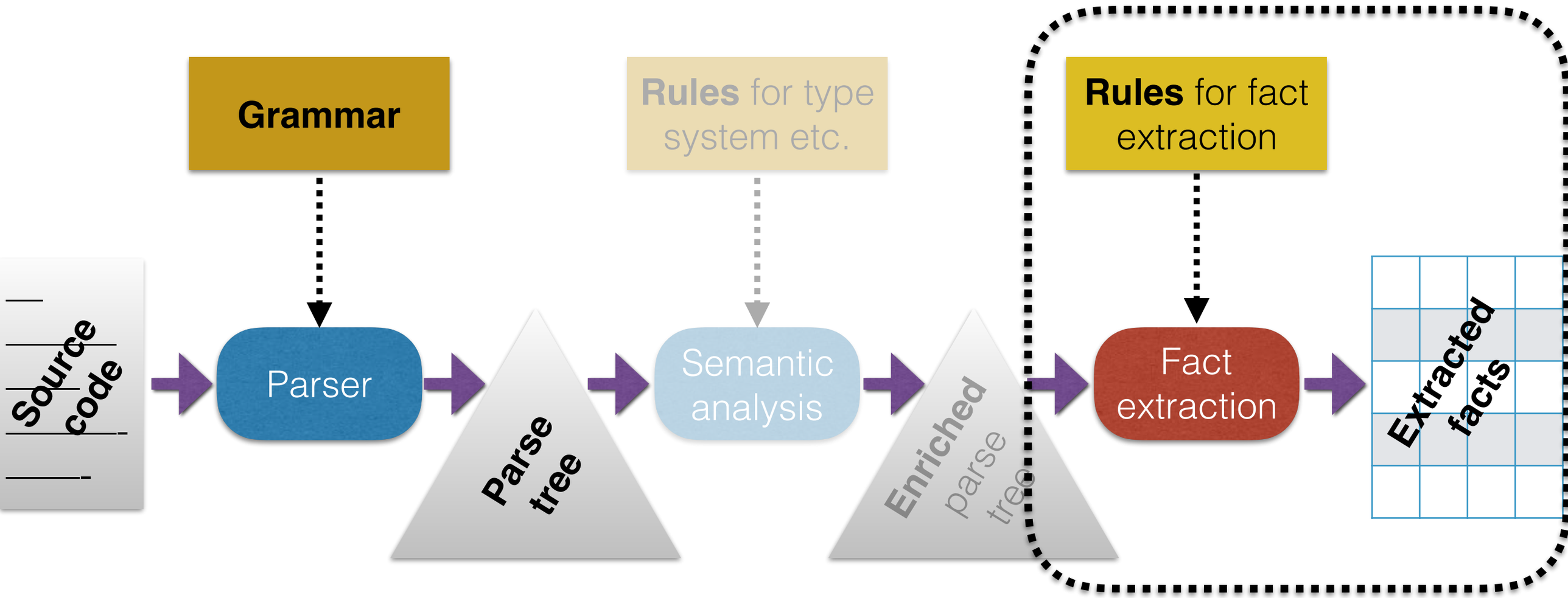
# Compilation



# Re-engineering



# Fact extraction in reverse engineering

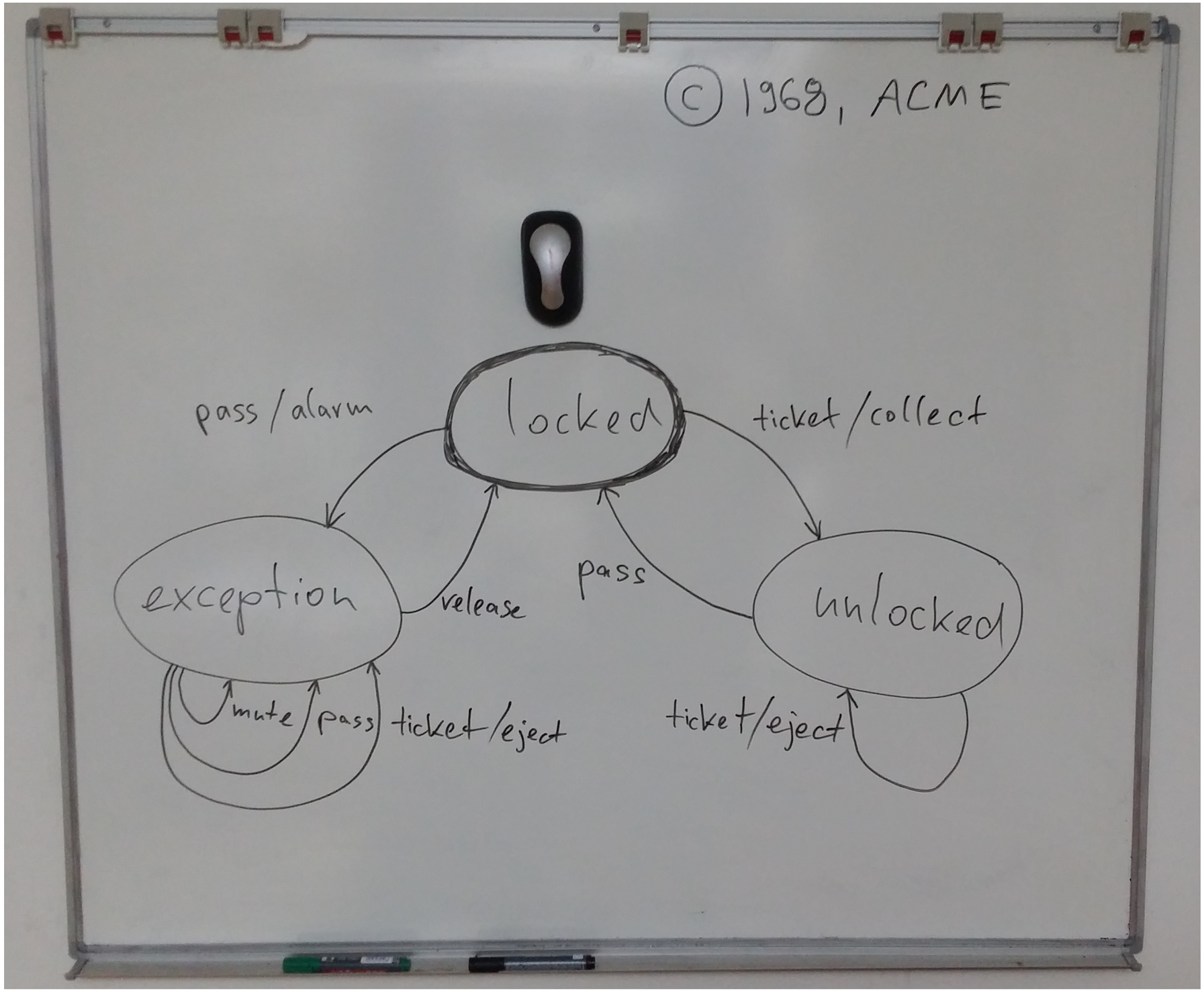


# Software Languages
## A slightly Haskell-biased introduction

- Part I — Introduction
  - Preface
  - The notion of software language
  - **Story of a language**
    - Language concepts
    - Internal DSL style
    - Textual syntax
    - Parsing text to objects
    - Language-agnostic models
    - Language constraints
    - Interpretation
    - Code generation
    - Visual syntax

- Part II — Syntax and metaprogramming
- Part III — Semantics and types
- Part IV — Consolidation

# The DSML FSML



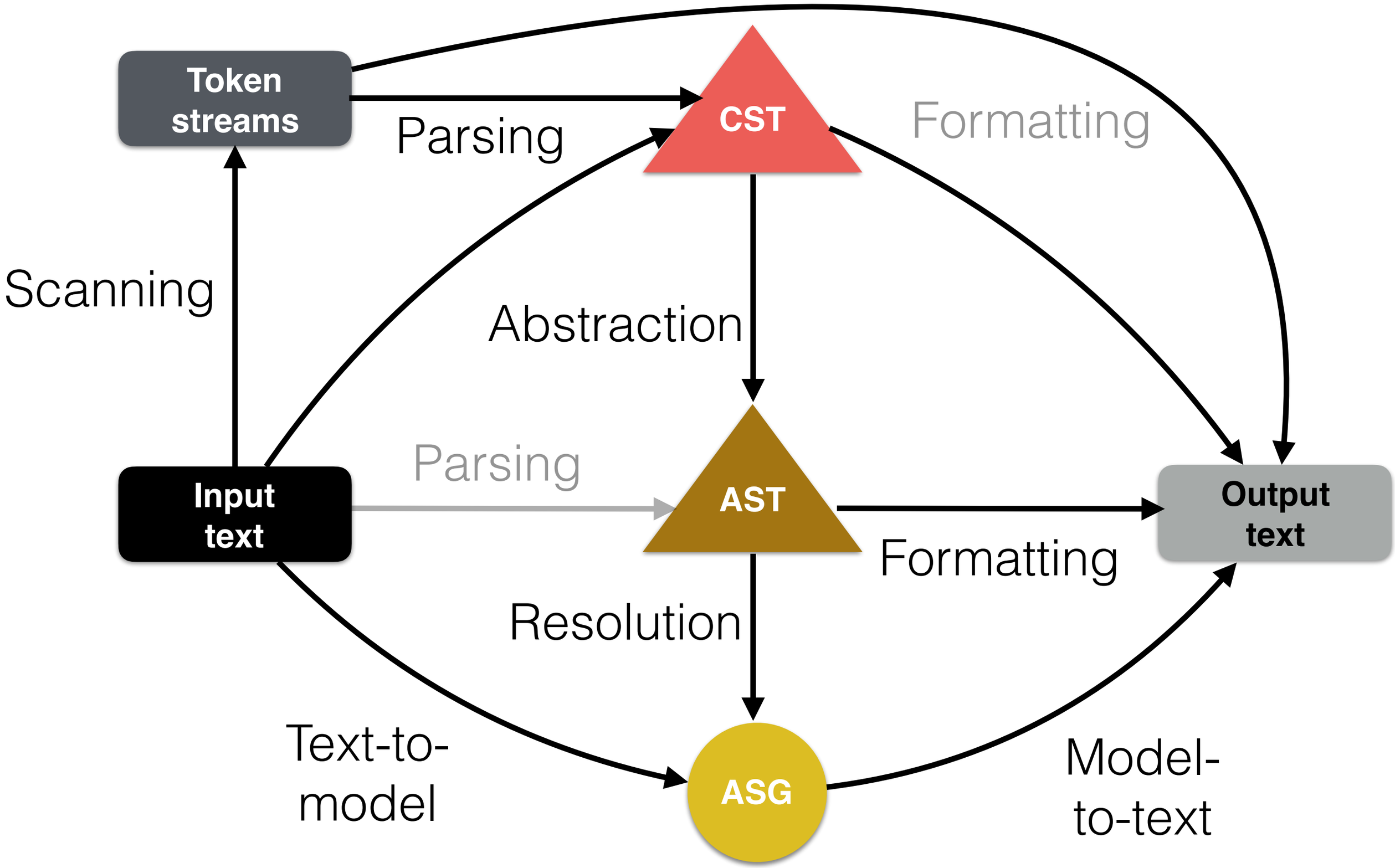
# Parsing (text-to-object) for FSML

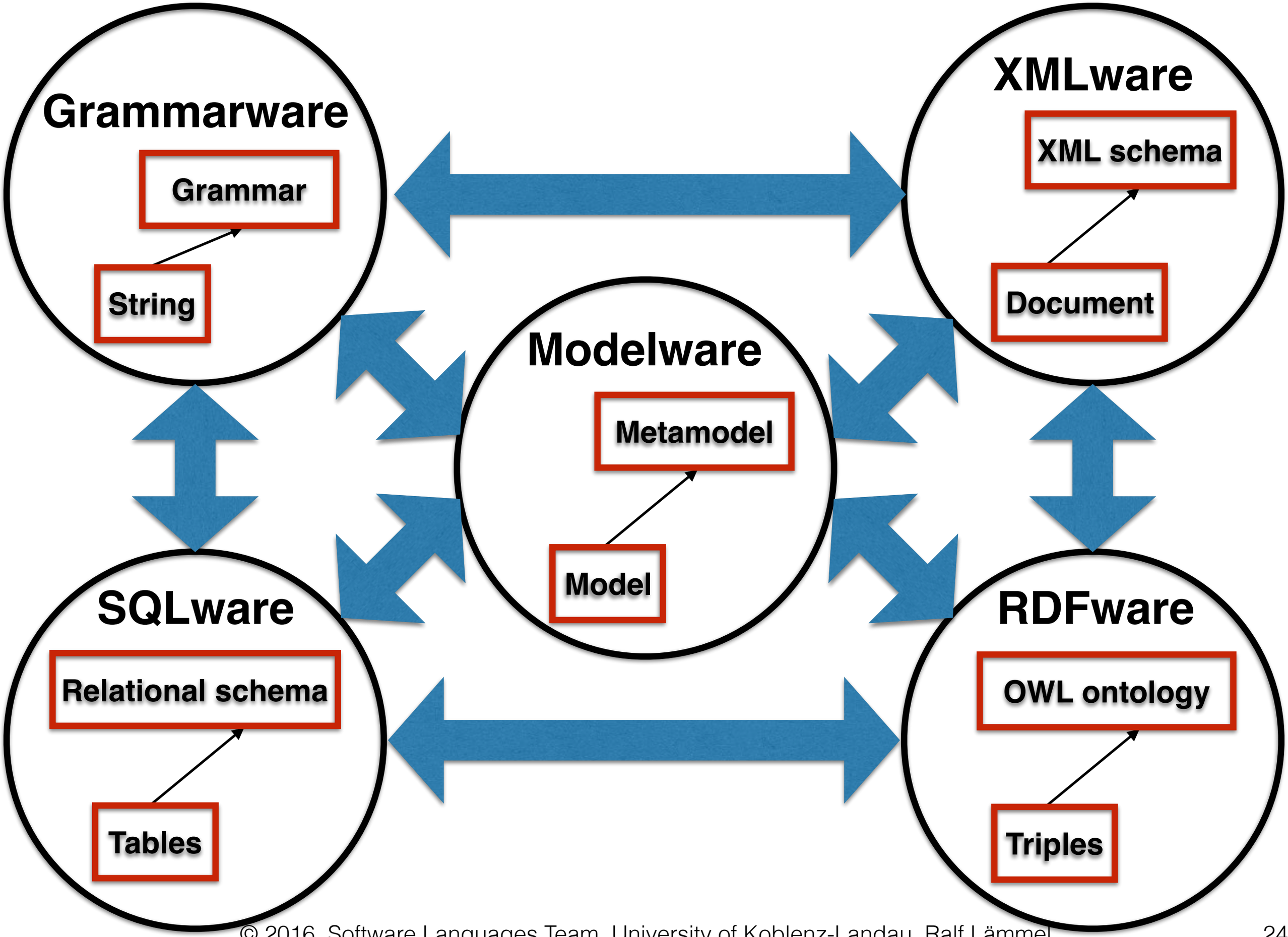
```
fsm : statedecl* EOF ;
statedecl :
   {boolean initial = false;}
   ('initial' {initial = true;})?
   'state' stateid
   {fsm.getStates().add(new State($stateid.text, initial));}
   '{' transition[$stateid.text]* '}'
   ;
transition[String source] :
   event
   {String action = null;}
   ('/' action {action = $action.text;})?
   {String target = source;}
   ('->' stateid {target = $stateid.text;})?
   {fsm.getTranss().add(new Trans(source, $event.text, action, target));}
   ';'
   ;
```

# Software Languages
## A slightly Haskell-biased introduction

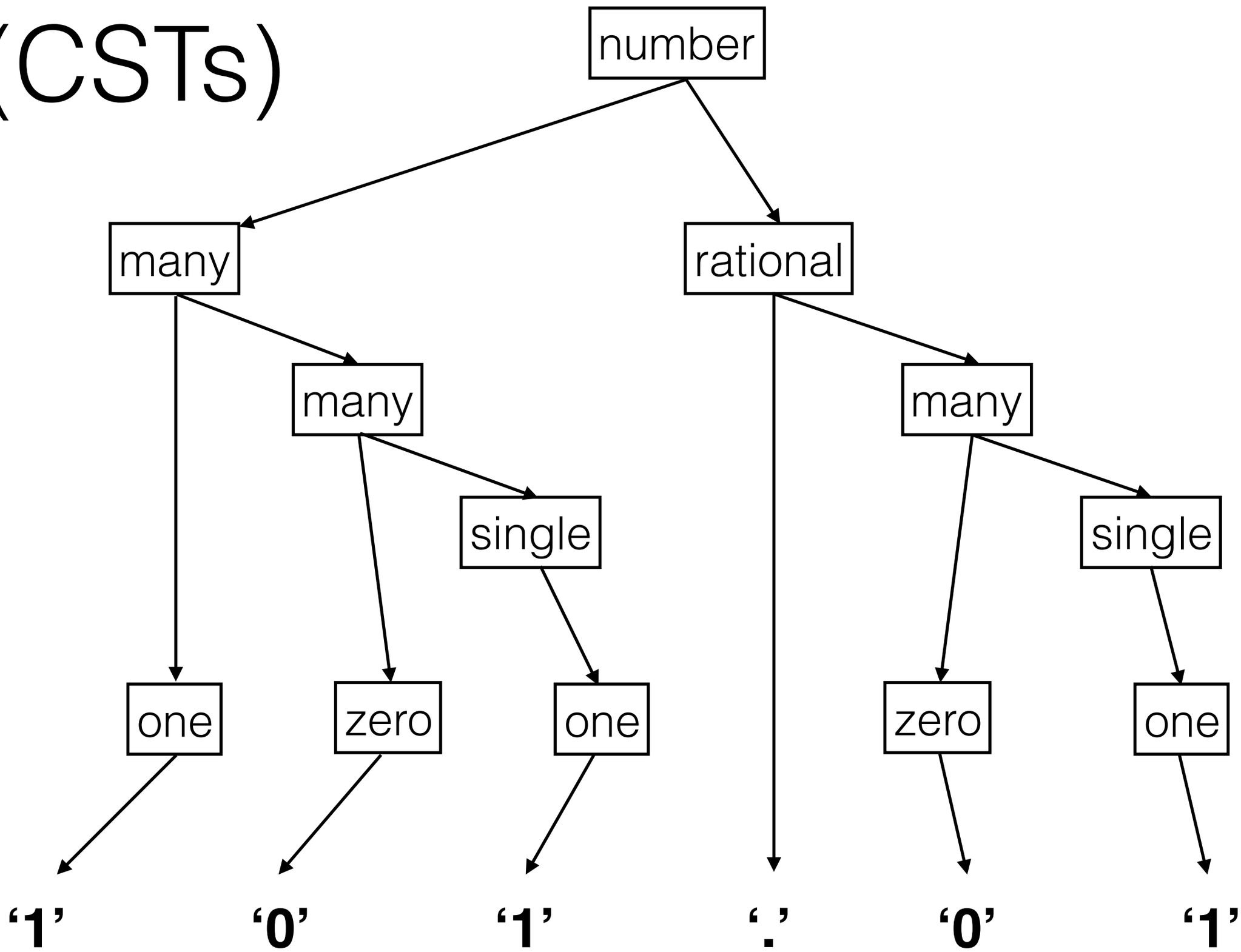
- Part I — Introduction
- **Part II — Syntax** and metaprogramming
  - **Abstract syntax—foundations**
  - **Abstract syntax—implementations**
  - Basics of metaprogramming
  - **Concrete syntax—foundations**
  - **Concrete syntax—implementations**
- Part III — Semantics and types
- Part IV — Consolidation

# Syntax mapping




Grammarware
Grammar
String
XMLware
XML schema
Document
Modelware
Metamodel
Model
SQLware
Relational schema
Tables
RDFware
OWL ontology
Triples

# Parse trees (CSTs)



# Mapping CSTs to ASTs

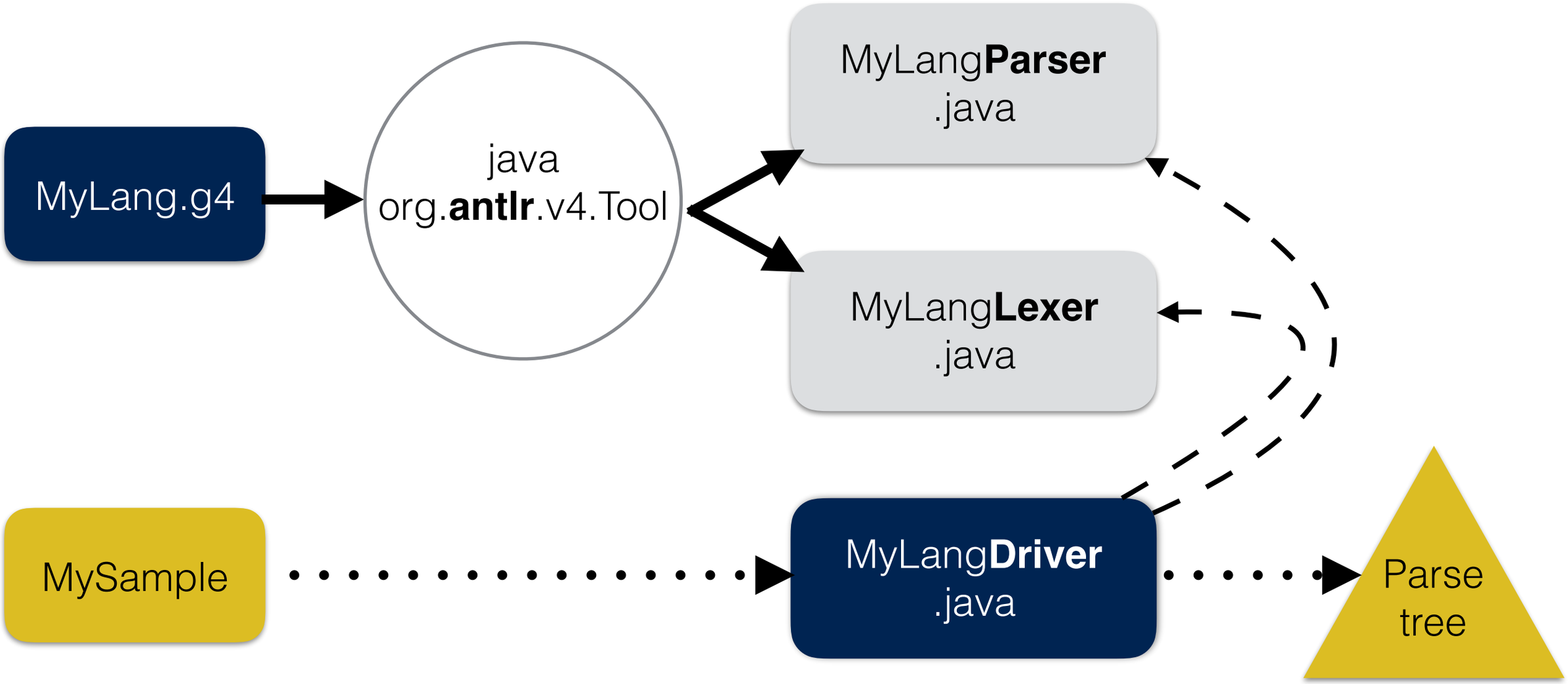
**Concrete syntax tree**

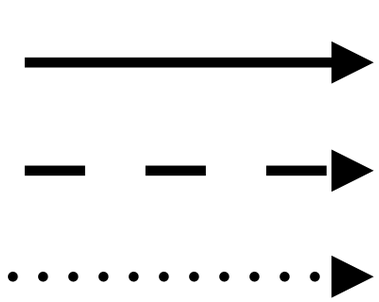
```
fork((number,number,[n(bits),n(rest)]),
  [ fork((many,bits,[n(bit),n(bits)]),
     [ fork((one,bit,[t('1')]),
      [leaf('1')]),
    fork((single,bits,[n(bit)]),
      [ fork((zero,bit,[t('0')]),
          [leaf('0')])])]),
   fork((integer,rest,[]),
     [])]).
```

**Abstract syntax tree**

```
number(
  many(
    one
    single(
      zero)),
  integer).
```

# Parser generation




# Mapping grammars to signatures

**concrete** `[rational] rest : '.' bits ;`

versus

**abstract** `symbol rational : bits -> rest ;`

syntax

# Resolution (AST-to-ASG)

**TBL**

```
newtype World = World {
  getPersons :: [Person]
}

data Person = Person {
  getName :: String,
  getBuddy :: Maybe String
}
```

**GBL**

```
newtype World = World {
  getPersons :: [IORef Person]
}

data Person = Person {
  getName :: String,
  getBuddy :: Maybe (IORef Person)
}
```

# Software Languages
## A slightly Haskell-biased introduction

- Part I — Introduction
- **Part II —** Syntax and **metaprogramming**
  - Abstract syntax—foundations
  - Abstract syntax—implementations
  - **Basics of metaprogramming**
  - Concrete syntax—foundations
  - Concrete syntax—implementations
- Part III — Semantics and types
- Part IV — Consolidation

# Pretty printing for FSML

```
initial state locked {
    ticket/collect -> unlocked;
    pass/alarm -> exception;
}
state unlocked {
    ticket/eject;
    pass -> locked;
}
...
```

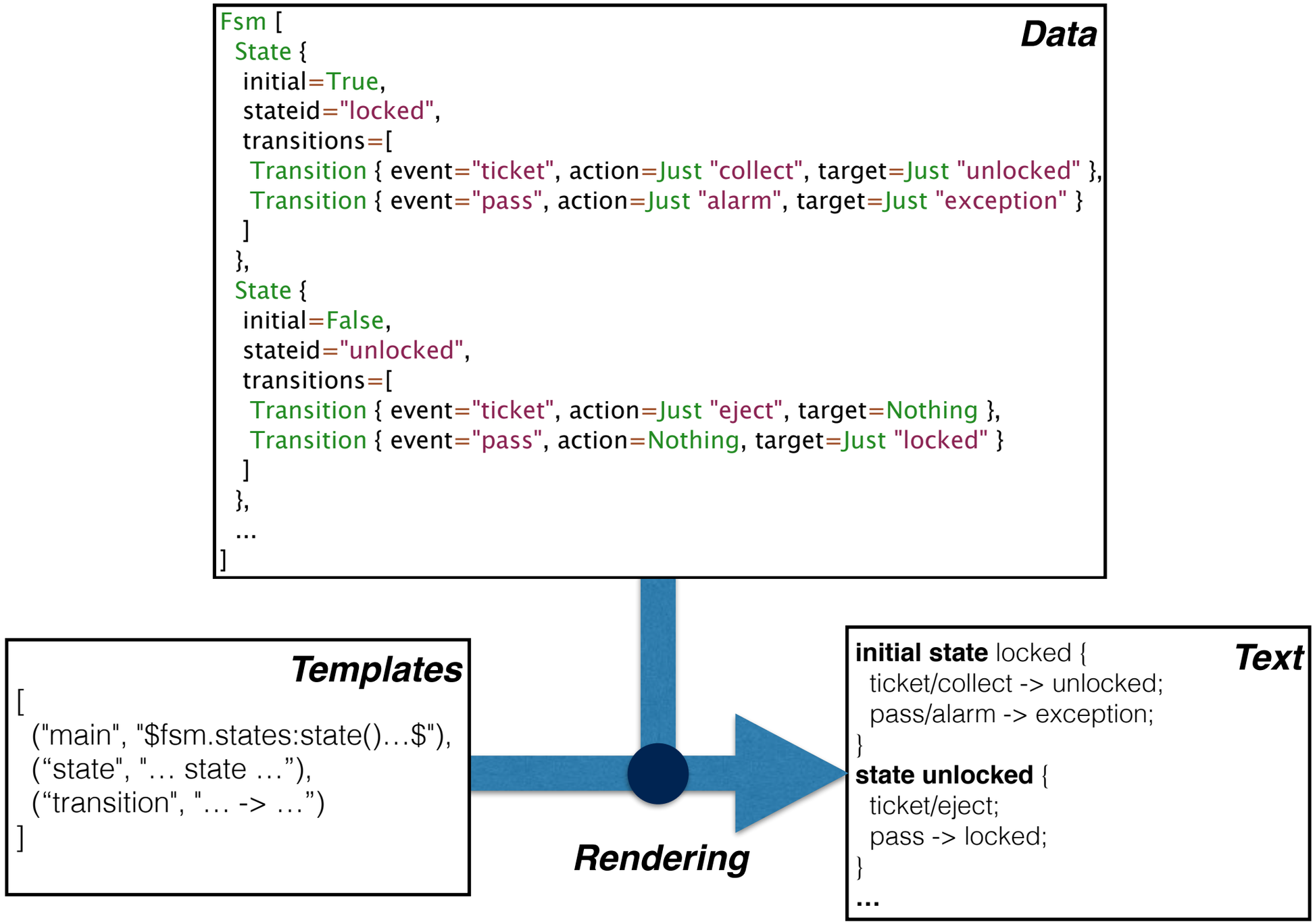
**Legend**

*horizontal composition*

*vertical composition*

*indentation*

# Template processing for FSML



# State id renaming for FSML

***Input***

```
initial state locked {
    ticket / collect -> unlocked;
    pass / alarm -> exception;
}
state unlocked {
    ticket / eject;
    pass -> locked;
}
...
```

***Output***

```
initial state closed {
    ticket / collect -> open;
    pass / alarm -> exception;
}
state open {
    ticket / eject;
    pass -> closed;
}
...
```

# Composition for FSML

```
initial state locked {
    ticket / collect -> unlocked;
}
state unlocked {
    ticket / eject;
    pass -> locked;
}
```

+

```
state locked {
   pass / alarm -> exception;
}
state exception {
   ticket / eject;
   pass;
   mute;
   release -> locked;
}
```

```
initial state locked {
    ticket / collect -> unlocked;
    pass / alarm -> exception;
}
state unlocked {
    ticket / eject;
    pass -> locked;
}
state exception {
   ticket / eject;
   pass;
   mute;
   release -> locked;
}
```

# Software Languages
## A slightly Haskell-biased introduction

- Part I — Introduction
- Part II — Syntax and metaprogramming
- **Part III — Semantics and types**
  - Operational semantics
  - Type systems
  - Denotational semantics
- Part IV — Consolidation

# Operational semantics (big step)

$$\text{zero} \longrightarrow \text{zero} \qquad [\text{ZERO}]$$

$$\frac{e \longrightarrow n}{\text{succ}(e) \longrightarrow \text{succ}(n)} \qquad [\text{SUCC}]$$

```
evaluate Zero = Just Zero
evaluate (Succ e) =
  case evaluate e of
    (Just n) →
      if isNat n
        then Just (Succ n)
        else Nothing
    Nothing → Nothing
```

# Type systems

$$\text{zero} : \text{nat} \qquad [\text{t}-\text{zero}]$$

$$\frac{e : \text{nat}}{\text{succ}(e) : \text{nat}} \qquad [\text{t}-\text{succ}]$$

```
typeOf Zero = Just Nat
typeOf (Succ e) =
  case typeOf e of
    (Just Nat) → Just Nat
    _ → Nothing
```

# Denotational semantics

$$
\begin{array}{ll}
\mathcal{S}\,[\![\mathsf{skip}]\!] & = \underline{skip} \\
\mathcal{S}\,[\![\mathsf{assign}(x,e)]\!] & = \underline{assign}\ x\ (\mathcal{E}\,[\![e]\!]) \\
\mathcal{S}\,[\![\mathsf{seq}(s_1,s_2)]\!] & = \underline{seq}\ (\mathcal{S}\,[\![s_1]\!])\ (\mathcal{S}\,[\![s_2]\!]) \\
\mathcal{S}\,[\![\mathsf{if}(e,s_1,s_2)]\!] & = \underline{if}\ (\mathcal{E}\,[\![e]\!])\ (\mathcal{S}\,[\![s_1]\!])\ (\mathcal{S}\,[\![s_2]\!]) \\
\mathcal{S}\,[\![\mathsf{while}(e,s)]\!] & = \underline{while}\ (\mathcal{E}\,[\![e]\!])\ (\mathcal{S}\,[\![s]\!])
\end{array}
$$

```
execute Skip = skip'
execute (Assign x e) = assign' x (evaluate e)
execute (Seq s1 s2) = seq' (execute s1) (execute s2)
execute (If e s1 s2) = if' (evaluate e) (execute s1) (execute s2)
execute (While e s) = while' (evaluate e) (execute s)
```

# Software Languages
## A slightly Haskell-biased introduction

- Part I — Introduction
- Part II — Syntax and metaprogramming
- Part III — Semantics and types
- **Part IV — Consolidation**
  - Metaprogramming techniques
  - Logic programming to the rescue
  - Postface

# Software Languages
## A slightly Haskell-biased introduction

- Part I — Introduction
- Part II — Syntax and metaprogramming
- Part III — Semantics and types
- Part IV — Consolidation
  - **Metaprogramming techniques**
    - Term rewriting
    - Attribute grammars
    - Multi-stage programming
    - Partial evaluation
  - Logic programming to the rescue
  - Postface

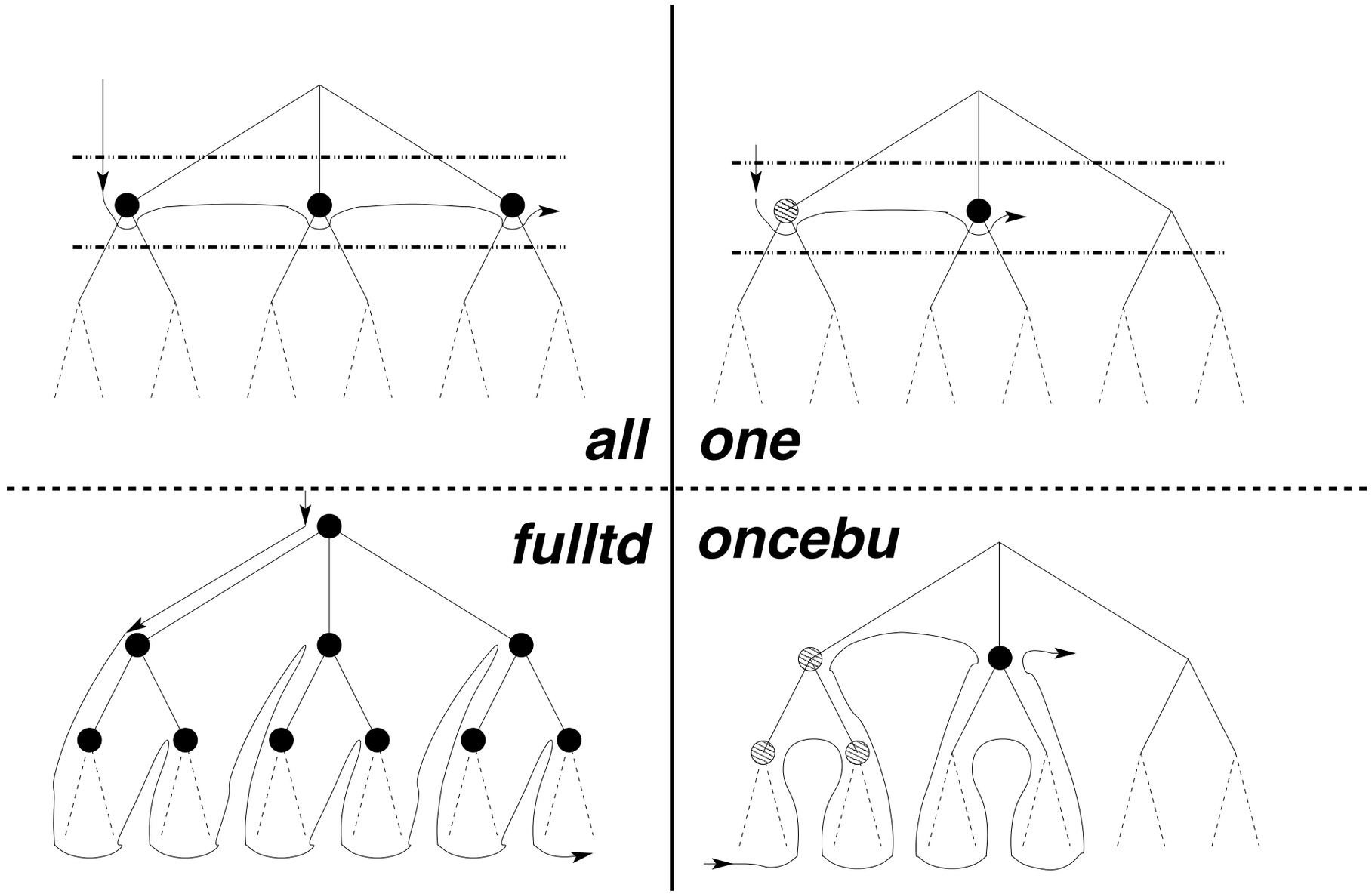
# Term rewriting incl. concrete object syntax

```
-- Laws on expressions
x + y = y + x
x * y = y * x
```

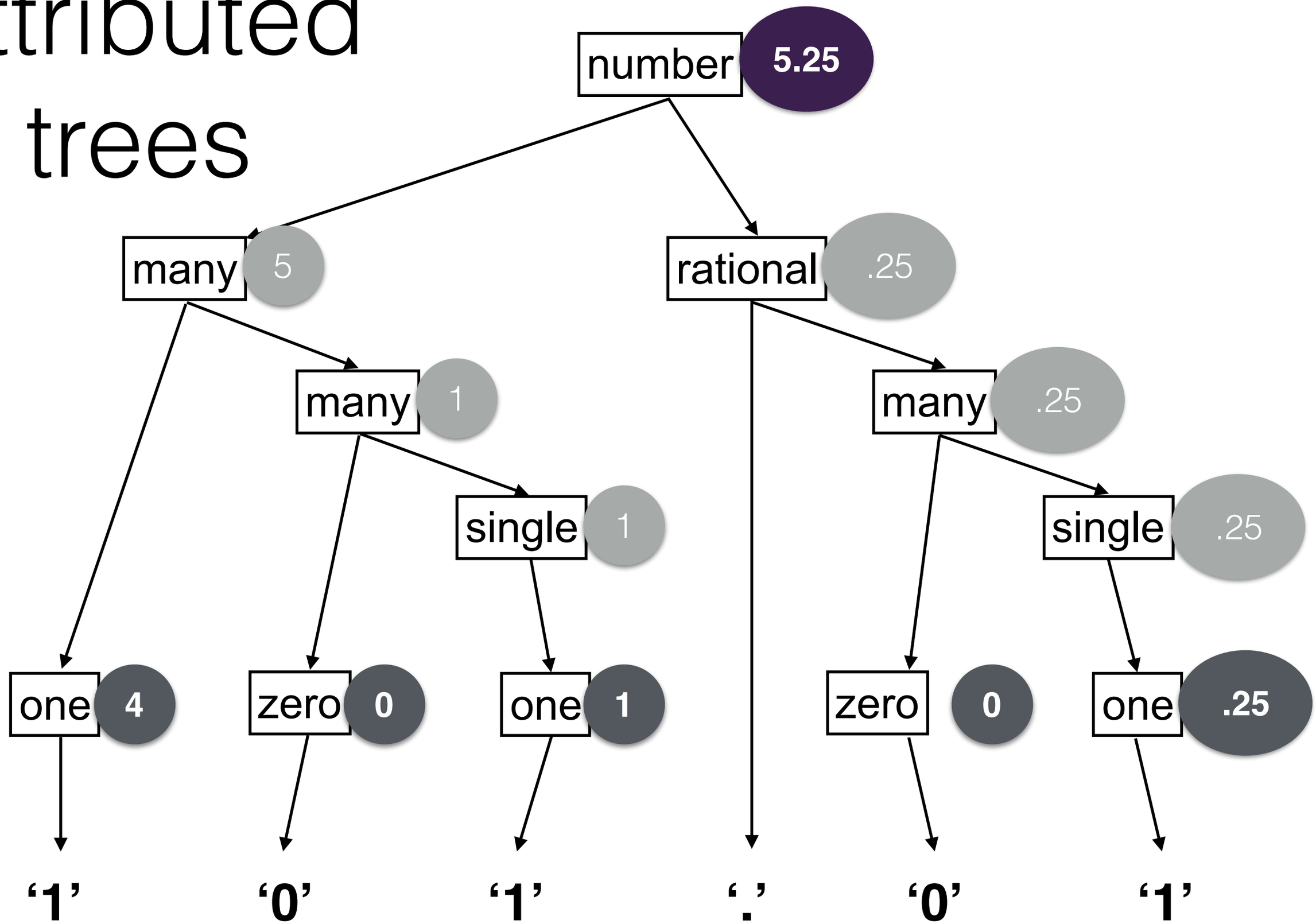
```
-- Implementation based on abstract object syntax
commute :: Expr -> Maybe Expr
commute (Binary Add x y) = Just $ Binary Add y x
commute (Binary Mul x y) = Just $ Binary Mul y x
commute _ = Nothing
```

```
-- Implementation based on concrete object syntax
commute :: Expr -> Maybe Expr
commute [el| $x + $y |] = Just [el| $y + $x |]
commute [el| $x * $y |] = Just [el| $y * $x |]
commute _ = Nothing
```

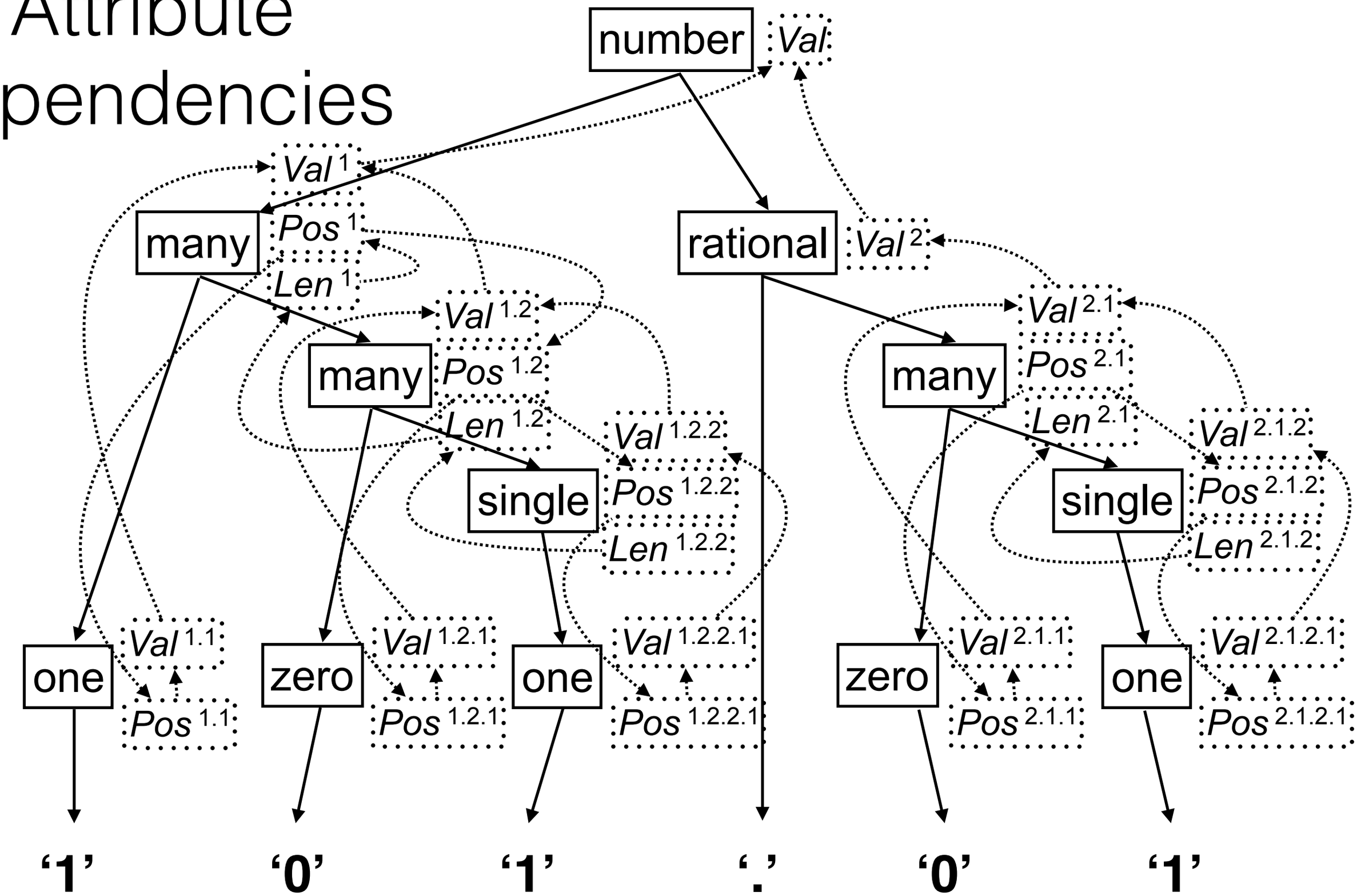
# Term rewriting incl. strategies



# Attributed trees



# Attribute dependencies

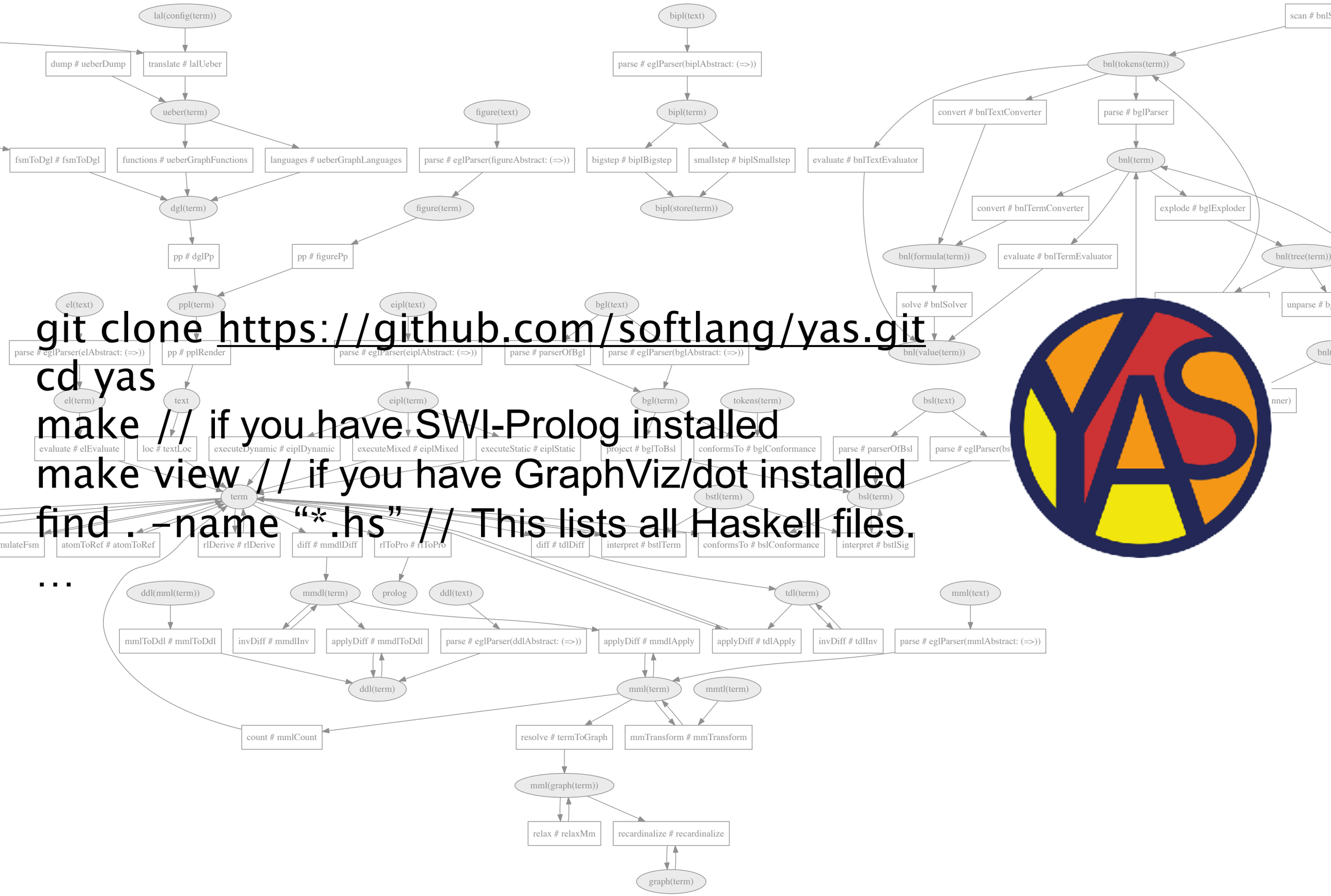


# Yet Another SLR
## (Software Language Repository)

The underlying repository




https://github.com/softlang/yas/




```
git clone https://github.com/softlang/yas.git
cd yas
make // if you have SWI-Prolog installed
make view // if you have GraphViz/dot installed
find . -name "*.hs" // This lists all Haskell files.
...
```

# Haskell

```
./languages/BFPL/Haskell/Language/BFPL/Domains.hs
./languages/BFPL/Haskell/Language/BFPL/Inliner.hs
./languages/BFPL/Haskell/Language/BFPL/Interpreter.hs
./languages/BFPL/Haskell/Language/BFPL/Samples/Exp.hs
./languages/BFPL/Haskell/Language/BFPL/Samples/Factorial.hs
./languages/BFPL/Haskell/Language/BFPL/Specializer.hs
./languages/BFPL/Haskell/Language/BFPL/Syntax.hs
./languages/BFPL/Haskell/Language/BFPL/Typing.hs
./languages/BFPL/Haskell/Main.hs
./languages/BGL/Haskell/Language/BGL/BottomUpAcceptor.hs
./languages/BGL/Haskell/Language/BGL/CST.hs
./languages/BGL/Haskell/Language/BGL/Sample.hs
./languages/BGL/Haskell/Language/BGL/SampleWithGreediness.hs
./languages/BGL/Haskell/Language/BGL/SampleWithoutEpsilon.hs
./languages/BGL/Haskell/Language/BGL/Syntax.hs
./languages/BGL/Haskell/Language/BGL/TopDownAcceptor.hs
./languages/BGL/Haskell/Language/BGL/TopDownParser.hs
./languages/BGL/Haskell/Main.hs
./languages/BIPL/Haskell/Data/Perhaps/Bool.hs
./languages/BIPL/Haskell/Data/Perhaps/Bottomness.hs
./languages/BIPL/Haskell/Data/Perhaps/Either.hs
./languages/BIPL/Haskell/Data/Perhaps/Finiteness.hs
./languages/BIPL/Haskell/Data/Perhaps/Map.hs
./languages/BIPL/Haskell/Data/Perhaps/Sign.hs
./languages/BIPL/Haskell/Data/Perhaps/Utilities.hs
./languages/BIPL/Haskell/Data/Perhaps.hs
./languages/BIPL/Haskell/Language/AL/Assembler.hs
```

```
./languages/BIPL/Haskell/Language/AL/Assembler.hs
./languages/BIPL/Haskell/Language/AL/Sample.hs
./languages/BIPL/Haskell/Language/AL/Syntax.hs
./languages/BIPL/Haskell/Language/BIPL/Algebra/Interpretation.hs
./languages/BIPL/Haskell/Language/BIPL/Algebra/Scheme.hs
./languages/BIPL/Haskell/Language/BIPL/Algebra/Signature.hs
./languages/BIPL/Haskell/Language/BIPL/Checker.hs
./languages/BIPL/Haskell/Language/BIPL/Compiler.hs
./languages/BIPL/Haskell/Language/BIPL/CS/Combinators.hs
./languages/BIPL/Haskell/Language/BIPL/CS/Domains.hs
./languages/BIPL/Haskell/Language/BIPL/CS/Interpreter.hs
./languages/BIPL/Haskell/Language/BIPL/Domains.hs
./languages/BIPL/Haskell/Language/BIPL/DS/Combinators.hs
./languages/BIPL/Haskell/Language/BIPL/DS/Domains.hs
./languages/BIPL/Haskell/Language/BIPL/DS/Interpreter.hs
./languages/BIPL/Haskell/Language/BIPL/Extraction.hs
./languages/BIPL/Haskell/Language/BIPL/Goto/Combinators.hs
./languages/BIPL/Haskell/Language/BIPL/Goto/Domains.hs
./languages/BIPL/Haskell/Language/BIPL/Goto/Interpreter.hs
./languages/BIPL/Haskell/Language/BIPL/Goto/Sample.hs
./languages/BIPL/Haskell/Language/BIPL/Goto/Syntax.hs
./languages/BIPL/Haskell/Language/BIPL/Interpreter.hs
./languages/BIPL/Haskell/Language/BIPL/Rename/Condition.hs
./languages/BIPL/Haskell/Language/BIPL/Rename/Transformation.hs
./languages/BIPL/Haskell/Language/BIPL/Sample.hs
./languages/BIPL/Haskell/Language/BIPL/Sign/Domains.hs
./languages/BIPL/Haskell/Language/BIPL/Sign/Fix.hs
./languages/BIPL/Haskell/Language/BIPL/Sign/Sample.hs
```

```
./languages/BIPL/Haskell/Language/BIPL/Sign/Domains.hs
./languages/BIPL/Haskell/Language/BIPL/Sign/Fix.hs
./languages/BIPL/Haskell/Language/BIPL/Sign/Sample.hs
./languages/BIPL/Haskell/Language/BIPL/Sign/V1/Analysis.hs
./languages/BIPL/Haskell/Language/BIPL/Sign/V2/Analysis.hs
./languages/BIPL/Haskell/Language/BIPL/Syntax.hs
./languages/BIPL/Haskell/Language/ML/Machine.hs
./languages/BIPL/Haskell/Language/ML/Sample.hs
./languages/BIPL/Haskell/Language/ML/Syntax.hs
./languages/BIPL/Haskell/Main.hs
./languages/BNL/ESL/haskell/as.hs
./languages/BNL/Haskell/Language/BNL/Acceptor.hs
./languages/BNL/Haskell/Language/BNL/Conversion.hs
./languages/BNL/Haskell/Language/BNL/Parser.hs
./languages/BNL/Haskell/Language/BNL/Syntax.hs
./languages/BNL/Haskell/Main.hs
./languages/BSL/Haskell/Language/BSL/Conformance.hs
./languages/BSL/Haskell/Language/BSL/Sample.hs
./languages/BSL/Haskell/Language/BSL/Syntax.hs
./languages/BSL/Haskell/Main.hs
./languages/BTL/Haskell/Language/BTL/BigStep.hs
./languages/BTL/Haskell/Language/BTL/BigStepWithGuards.hs
./languages/BTL/Haskell/Language/BTL/Closure.hs
./languages/BTL/Haskell/Language/BTL/Domains.hs
./languages/BTL/Haskell/Language/BTL/Dynamics.hs
./languages/BTL/Haskell/Language/BTL/Interpreter.hs
./languages/BTL/Haskell/Language/BTL/Nf.hs
```

```
./languages/BTL/Haskell/Language/BTL/Interpreter.hs
./languages/BTL/Haskell/Language/BTL/Nf.hs
./languages/BTL/Haskell/Language/BTL/Sample.hs
./languages/BTL/Haskell/Language/BTL/SmallStep.hs
./languages/BTL/Haskell/Language/BTL/Syntax.hs
./languages/BTL/Haskell/Language/BTL/Typing.hs
./languages/BTL/Haskell/Language/BTL/Universal/Sample.hs
./languages/BTL/Haskell/Language/BTL/Universal/Term.hs
./languages/BTL/Haskell/Main.hs
./languages/EGL/Haskell/Language/EGL/Interpreter.hs
./languages/EGL/Haskell/Language/EGL/Lexer.hs
./languages/EGL/Haskell/Language/EGL/Parser.hs
./languages/EGL/Haskell/Language/EGL/Sample.hs
./languages/EGL/Haskell/Language/EGL/Syntax.hs
./languages/EGL/Haskell/Main.hs
./languages/EL/Haskell/Language/EL/Lexer.hs
./languages/EL/Haskell/Language/EL/MoreRules.hs
./languages/EL/Haskell/Language/EL/Normalizer.hs
./languages/EL/Haskell/Language/EL/Parser.hs
./languages/EL/Haskell/Language/EL/QQ/Lexer.hs
./languages/EL/Haskell/Language/EL/QQ/MoreRules.hs
./languages/EL/Haskell/Language/EL/QQ/Parser.hs
./languages/EL/Haskell/Language/EL/QQ/Rules/Associate.hs
./languages/EL/Haskell/Language/EL/QQ/Rules/Commute.hs
./languages/EL/Haskell/Language/EL/QQ/Rules/Simplify.hs
./languages/EL/Haskell/Language/EL/QQ/Rules.hs
./languages/EL/Haskell/Language/EL/QQ/Syntax.hs
./languages/EL/Haskell/Language/EL/QuasiQuoter.hs
```

```
./languages/EL/Haskell/Language/EL/Rules.hs
./languages/EL/Haskell/Language/EL/Syntax.hs
./languages/EL/Haskell/Main.hs
./languages/FSML/Haskell/Language/FSML/Acceptor.hs
./languages/FSML/Haskell/Language/FSML/ApplicativeParser.hs
./languages/FSML/Haskell/Language/FSML/BoolChecker.hs
./languages/FSML/Haskell/Language/FSML/CombinatorFormatter.hs
./languages/FSML/Haskell/Language/FSML/Domains.hs
./languages/FSML/Haskell/Language/FSML/Eq.hs
./languages/FSML/Haskell/Language/FSML/Extraction.hs
./languages/FSML/Haskell/Language/FSML/Interpreter.hs
./languages/FSML/Haskell/Language/FSML/Lexer.hs
./languages/FSML/Haskell/Language/FSML/Merge/Condition.hs
./languages/FSML/Haskell/Language/FSML/Merge/Sample.hs
./languages/FSML/Haskell/Language/FSML/Merge/Transformation.hs
./languages/FSML/Haskell/Language/FSML/MonadicParser.hs
./languages/FSML/Haskell/Language/FSML/Normalization.hs
./languages/FSML/Haskell/Language/FSML/QQ/Determinism.hs
./languages/FSML/Haskell/Language/FSML/QQ/DistinctIds.hs
./languages/FSML/Haskell/Language/FSML/QQ/Reachability.hs
./languages/FSML/Haskell/Language/FSML/QQ/ResolvableTargets.hs
./languages/FSML/Haskell/Language/FSML/QQ/Sample.hs
./languages/FSML/Haskell/Language/FSML/QQ/SingleInitial.hs
./languages/FSML/Haskell/Language/FSML/QuasiQuoter.hs
./languages/FSML/Haskell/Language/FSML/Rename/Condition.hs
./languages/FSML/Haskell/Language/FSML/Rename/Sample.hs
./languages/FSML/Haskell/Language/FSML/Rename/Transformation.hs
./languages/FSML/Haskell/Language/FSML/Sample.hs
```

```
./languages/FSML/Haskell/Language/FSML/Rename/Sample.hs
./languages/FSML/Haskell/Language/FSML/Rename/Transformation.hs
./languages/FSML/Haskell/Language/FSML/Sample.hs
./languages/FSML/Haskell/Language/FSML/SampleIO.hs
./languages/FSML/Haskell/Language/FSML/StringChecker.hs
./languages/FSML/Haskell/Language/FSML/StringTemplate/Sample.hs
./languages/FSML/Haskell/Language/FSML/StringTemplate/Syntax.hs
./languages/FSML/Haskell/Language/FSML/Syntax.hs
./languages/FSML/Haskell/Language/FSML/TemplateFormatter.hs
./languages/FSML/Haskell/Main.hs
./languages/GBL/Haskell/Language/GBL/Resolver.hs
./languages/GBL/Haskell/Language/GBL/Sample.hs
./languages/GBL/Haskell/Language/GBL/Syntax.hs
./languages/GBL/Haskell/Language/GBL/V2/Sample.hs
./languages/GBL/Haskell/Language/GBL/V2/Syntax.hs
./languages/GBL/Haskell/Main.hs
./languages/TBL/Haskell/Language/TBL/Sample.hs
./languages/TBL/Haskell/Language/TBL/Syntax.hs
./languages/TBL/Haskell/Main.hs
./lib/Haskell/Data/Generics/SimpleStrategySchemes.hs
./lib/Haskell/Data/Generics/StrategyBasics.hs
./lib/Haskell/Data/Generics/StrategyPrimitives.hs
./lib/Haskell/Data/Generics/StrategySchemes.hs
./lib/Haskell/Data/Graph.hs
./lib/Haskell/Data/Term.hs
./lib/Haskell/Data/TermRep.hs
./lib/Haskell/Text/Parsec/Utilities.hs
./samples/Haskell/Power.hs
```

```
./lib/Haskell/Data/Generics/SimpleStrategySchemes.hs
./lib/Haskell/Data/Generics/StrategyBasics.hs
./lib/Haskell/Data/Generics/StrategyPrimitives.hs
./lib/Haskell/Data/Generics/StrategySchemes.hs
./lib/Haskell/Data/Graph.hs
./lib/Haskell/Data/Term.hs
./lib/Haskell/Data/TermRep.hs
./lib/Haskell/Text/Parsec/Utilities.hs
./samples/Haskell/Power.hs
./samples/Haskell/TestTypedPower.hs
./samples/Haskell/TestUntypedPower.hs
./samples/Haskell/TypedPower.hs
./samples/Haskell/UntypedPower.hs
```

$$
\begin{aligned}
S &\rightarrow aS \mid bX \\
X &\rightarrow aX \mid bY \\
Y &\rightarrow aY \mid bZ \mid \Lambda \\
Z &\rightarrow aZ \mid \Lambda
\end{aligned}
$$

Source: http://stackoverflow.com/questions/29111649/language-generated-by-a-context-free-grammar

# Grammars

```
./languages/BNL/cs.bgl
./languages/BTL/cs.bgl
./languages/expr/cs.bgl

./languages/BFPL/cs.egl
./languages/BFPL/ls.egl
./languages/BGL/cs.egl
./languages/BGL/ls.egl
./languages/BIPL/cs.egl
./languages/BIPL/ls.egl
./languages/BMPL/cs.egl
./languages/BMPL/ls.egl
./languages/BNL/EGL/cs.egl
./languages/BNL/EGL/ls.egl
./languages/BSL/cs.egl
./languages/BSL/ls.egl
./languages/BTL/ls.egl
./languages/DDL/cs.egl
./languages/DDL/ls.egl
```

$$
\begin{array}{rcl}
S & \rightarrow & aS \mid bX \\
X & \rightarrow & aX \mid bY \\
Y & \rightarrow & aY \mid bZ \mid \Lambda \\
Z & \rightarrow & aZ \mid \Lambda
\end{array}
$$

```
./languages/DDL/cs.egl
./languages/DDL/ls.egl
./languages/EFPL/cs.egl
./languages/EFPL/ls.egl
./languages/EGL/cs.egl
./languages/EGL/ls.egl
./languages/EGL/samples/ambiguity.egl
./languages/EGTL/cs.egl
./languages/EGTL/ls.egl
./languages/EIPL/cs.egl
./languages/EIPL/ls.egl
./languages/EL/cs.egl
./languages/EL/ls.egl
./languages/EL/qq/cs.egl
./languages/EL/qq/ls.egl
./languages/ESL/cs.egl
./languages/ESL/ls.egl
./languages/figure/cs.egl
./languages/figure/ls.egl
```

$$
\begin{array}{rcl}
S & \rightarrow & aS \mid bX \\
X & \rightarrow & aX \mid bY \\
Y & \rightarrow & aY \mid bZ \mid \Lambda \\
Z & \rightarrow & aZ \mid \Lambda
\end{array}
$$

```
./languages/figure/cs.egl
./languages/figure/ls.egl
./languages/FSML/cs.egl
./languages/FSML/ls.egl
./languages/LAL/cs.egl
./languages/LAL/ls.egl
./languages/MML/cs.egl
./languages/MML/ls.egl
./languages/TBL/cs.egl
./languages/TBL/ls.egl
```

$$
\begin{array}{lcl}
S & \rightarrow & aS \mid bX \\
X & \rightarrow & aX \mid bY \\
Y & \rightarrow & aY \mid bZ \mid \Lambda \\
Z & \rightarrow & aZ \mid \Lambda
\end{array}
$$

```
symbol number: bits # rest -> number ;
symbol single: bit -> bits ;
symbol many: bit # bits -> bits ;
symbol zero: -> bit ;
symbol one: -> bit ;
symbol integer: -> rest ;
symbol rational: bits -> rest ;
```

# Signatures

```
./languages/BNL/as.bsl
./languages/BSL/sample.bsl
./languages/BSTL/tests/sig1.bsl
./languages/BSTL/tests/sig2.bsl

./languages/BTL/as.bsl
./languages/expr/as.bsl
./languages/BCL/as.esl
./languages/BFPL/as.esl
./languages/BGL/as.esl
./languages/BIPL/as.esl
./languages/BMPL/as.esl
./languages/BNL/ESL/as.esl
./languages/BSL/as.esl
./languages/BSTL/as.esl
./languages/DDL/as.esl
./languages/DGL/as.esl
./languages/EFPL/as.esl
./languages/EGL/as.esl
```

```
symbol number: bits # rest -> number ;
symbol single: bit -> bits ;
symbol many: bit # bits -> bits ;
symbol zero: -> bit ;
symbol one: -> bit ;
symbol integer: -> rest ;
symbol rational: bits -> rest ;
```

```
symbol number: bits # rest -> number ;
symbol single: bit -> bits ;
symbol many: bit # bits -> bits ;
symbol zero: -> bit ;
symbol one: -> bit ;
symbol integer: -> rest ;
symbol rational: bits -> rest ;
```

```
./languages/EGL/as.esl
./languages/EGTL/as.esl
./languages/EIPL/as.esl
./languages/EL/as.esl
./languages/EL/qq/as.esl
./languages/ESL/as.esl
./languages/figure/as.esl
./languages/FSML/as.esl
./languages/FSML/Prolog/to-java/as.esl
./languages/LAL/as.esl
./languages/LAL/config/as.esl
./languages/MMDL/as.esl
./languages/MML/as.esl
./languages/MMTL/as.esl
./languages/PPL/as.esl
./languages/TBL/as.esl
./languages/TDL/as.esl
./languages/ueber/as.esl
```

SWI Prolog

```
./languages/BCL/bcl0k.pro
./languages/BFPL/as-to-cs.pro
./languages/BFPL/bfplAbstract.pro
./languages/BFPL/Prolog/big-step/evaluate.pro
./languages/BFPL/Prolog/big-step/op.pro
./languages/BFPL/Prolog/big-step/value.pro
./languages/BFPL/Prolog/denotational/combinators.pro
./languages/BFPL/Prolog/denotational/expr.pro
./languages/BFPL/Prolog/denotational/op.pro
./languages/BFPL/Prolog/denotational/prog.pro
./languages/BFPL/Prolog/small-step/closure.pro
./languages/BFPL/Prolog/small-step/normal.pro
./languages/BFPL/Prolog/small-step/op.pro
./languages/BFPL/Prolog/small-step/step.pro
./languages/BFPL/Prolog/small-step/substitution.pro
./languages/BFPL/Prolog/typing/expr.pro
./languages/BFPL/Prolog/typing/op.pro
./languages/BFPL/Prolog/typing/prog.pro
./languages/BGL/api.pro
./languages/BGL/bglAbstract.pro
./languages/BGL/bglAcceptBottomUp.pro
./languages/BGL/bglAcceptTopDown.pro
```

```
./languages/BGL/bglAcceptBottomUp.pro
./languages/BGL/bglAcceptTopDown.pro
./languages/BGL/bglParseTopDown.pro
./languages/BGL/egl-to-bgl.pro
./languages/BGL/explode.pro
./languages/BGL/implode.pro
./languages/BGL/unparse.pro
./languages/BIPL/biplAbstract.pro
./languages/BIPL/Prolog/big-step/evaluate.pro
./languages/BIPL/Prolog/big-step/execute.pro
./languages/BIPL/Prolog/small-step/closure.pro
./languages/BIPL/Prolog/small-step/normal.pro
./languages/BIPL/Prolog/small-step/step.pro
./languages/BIPL/Prolog/store.pro
./languages/BMPL/bmplAbstract.pro
./languages/BMPL/bmplToProlog.pro
./languages/BNL/bnlAcceptor.pro
./languages/BNL/bnlFomula.pro
./languages/BNL/bnlParser.pro
./languages/BNL/bnlSolver.pro
./languages/BNL/bnlTermConverter.pro
./languages/BNL/bnlTermEvaluator.pro
./languages/BNL/bnlTermSEvaluator.pro
```

```
./languages/BNL/bnlTermEvaluator.pro
./languages/BNL/bnlTermSEvaluator.pro
./languages/BNL/bnlTextConverter.pro
./languages/BNL/bnlTextEvaluator.pro
./languages/BNL/Prolog/as.pro
./languages/BNL/scanner.pro
./languages/BSL/api.pro
./languages/BSL/as.pro
./languages/BSL/bgl-to-bsl.pro
./languages/BSL/bslAbstract.pro
./languages/BSL/bslConformance.pro
./languages/BSL/esl-to-bsl.pro
./languages/BSL/prefix-term.pro
./languages/BSL/tuple-term.pro
./languages/BSTL/bstlSig.pro
./languages/BSTL/bstlTerm.pro
./languages/BTL/Prolog/bigstep.pro
./languages/BTL/Prolog/btlSyntax.pro
./languages/BTL/Prolog/closure.pro
./languages/BTL/Prolog/smallstep.pro
./languages/BTL/Prolog/typing.pro
./languages/BTL/Prolog/values.pro
./languages/BTL/scanner.pro
```

SWI Prolog

```
./languages/BTL/Prolog/typing.pro
./languages/BTL/Prolog/values.pro
./languages/BTL/scanner.pro
./languages/DDL/ddlAbstract.pro
./languages/DDL/ddlEq.pro
./languages/DGL/pp.pro
./languages/EFPL/as-to-cs.pro
./languages/EFPL/efplAbstract.pro
./languages/EFPL/Prolog/big-step/evaluate.pro
./languages/EFPL/Prolog/efplAlpha.pro
./languages/EFPL/Prolog/efplInference.pro
./languages/EFPL/Prolog/efplTyping.pro
./languages/EGL/api.pro
./languages/EGL/eglAbstract.pro
./languages/EGL/parse-debug.pro
./languages/EGL/parse.pro
./languages/EGL/scanner.pro
./languages/EGTL/egtlAbstract.pro
./languages/EGTL/egtlExtension.pro
./languages/EGTL/egtlGeq.pro
./languages/EGTL/egtlInterpret.pro
./languages/EIPL/eiplAbstract.pro
./languages/EIPL/Prolog/dynamic.pro
```

```
./languages/EIPL/eiplAbstract.pro
./languages/EIPL/Prolog/dynamic.pro
./languages/EIPL/Prolog/mixed.pro
./languages/EIPL/Prolog/static.pro
./languages/EIPL/Prolog/typing.pro
./languages/EIPL/Prolog/undeclare.pro
./languages/EL/elAbstract.pro
./languages/EL/elEvaluate.pro
./languages/EL/elOps.pro
./languages/EL/elRewrite.pro
./languages/EL/qq/elqqAbstract.pro
./languages/EL/qq/elqqQuasiQuoter.pro
./languages/EL/qq/rules/elqqAssociate.pro
./languages/EL/qq/rules/elqqCommute.pro
./languages/EL/qq/rules/elqqSimplify.pro
./languages/EL/rules/elAssociate.pro
./languages/EL/rules/elCommute.pro
./languages/EL/rules/elSimplify.pro
./languages/ESL/api.pro
./languages/ESL/applied-term.pro
./languages/ESL/as.pro
./languages/ESL/conformance.pro
./languages/ESL/eslAbstract.pro
```

SWI Prolog

```
./languages/ESL/conformance.pro
./languages/ESL/eslAbstract.pro
./languages/expr/scanner.pro
./languages/figure/figureAbstract.pro
./languages/figure/parsing/scanner.pro
./languages/figure/pp.pro
./languages/FSML/fsmlAbstract.pro
./languages/FSML/Prolog/accept.pro
./languages/FSML/Prolog/fsmlConstraints.pro
./languages/FSML/Prolog/simulation.pro
./languages/FSML/Prolog/to-dgl.pro
./languages/FSML/Prolog/to-java/action.pro
./languages/FSML/Prolog/to-java/as-to-cs.pro
./languages/FSML/Prolog/to-java/input.pro
./languages/FSML/Prolog/to-java/state.pro
./languages/FSML/Prolog/to-java/stepper.pro
./languages/GBL/gblAbstract.pro
./languages/GBL/tblToGbl.pro
./languages/LAL/lalAbstract.pro
./languages/LAL/lalDeps.pro
./languages/LAL/lalNorm.pro
./languages/LAL/lalOk.pro
./languages/LAL/lalPp.pro
```

SWI Prolog

```
./languages/LAL/lalOk.pro
./languages/LAL/lalPp.pro
./languages/LAL/lalReuse.pro
./languages/LAL/lalTermEq.pro
./languages/LAL/lalTextEq.pro
./languages/LAL/lalUeber.pro
./languages/MMDL/mmdlApply.pro
./languages/MMDL/mmdlDiff.pro
./languages/MMDL/mmdlInv.pro
./languages/MMDL/mmdlToDdl.pro
./languages/MML/api.pro
./languages/MML/as.pro
./languages/MML/atom-to-ref/coupling.pro
./languages/MML/atom-to-ref/mmt.pro
./languages/MML/atom-to-ref/mt.pro
./languages/MML/conformance/cardinality.pro
./languages/MML/conformance/main.pro
./languages/MML/conformance/member.pro
./languages/MML/conformance/mml2pro.pro
./languages/MML/crud.pro
./languages/MML/editing/add-object.pro
./languages/MML/editing/lists.pro
./languages/MML/editing/new-object.pro
```

```
./languages/MML/crud.pro
./languages/MML/editing/add-object.pro
./languages/MML/editing/lists.pro
./languages/MML/editing/new-object.pro
./languages/MML/editing/obj-add-key.pro
./languages/MML/editing/obj-remove-key.pro
./languages/MML/editing/remove-object.pro
./languages/MML/graph-chase.pro
./languages/MML/mmlAbstract.pro
./languages/MML/mmlCount.pro
./languages/MML/mmlDdl.pro
./languages/MML/mmlEq.pro
./languages/MML/mmlToDdl.pro
./languages/MML/mmlTree.pro
./languages/MML/ok.pro
./languages/MML/recardinalize.pro
./languages/MML/relax-cardinality.pro
./languages/MML/subclass.pro
./languages/MML/term-chase.pro
./languages/MML/term-to-graph.pro
./languages/MMTL/mmTransform.pro
./languages/MMTL/mmValueToReference.pro
./languages/PPL/pplRender.pro
```

```
./languages/MMTL/mmTransform.pro
./languages/MMTL/mmValueToReference.pro
./languages/PPL/pplRender.pro
./languages/Prolog/eval.pro
./languages/Prolog/expr.pro
./languages/TBL/tblAbstract.pro
./languages/TBL/tblGraph.pro
./languages/TBL/tblOk.pro
./languages/TBL/tblOne.pro
./languages/TDL/tdlApply.pro
./languages/TDL/tdlDiff.pro
./languages/TDL/tdlInv.pro
./languages/Term/dict/abssyn.pro
./languages/Term/dict/conversion.pro
./languages/Term/dict/crud.pro
./languages/Term/graph/abssyn.pro
./languages/Term/graph/as-to-cs.pro
./languages/Term/graph/conversion.pro
./languages/Term/graph/crud.pro
./languages/Term/graph/deref.pro
./languages/Term/graph/iterate.pro
./languages/Term/graph/normal.pro
./languages/Term/graph/search.pro
```

SWI Prolog

```
./languages/Term/graph/crud.pro
./languages/Term/graph/deref.pro
./languages/Term/graph/iterate.pro
./languages/Term/graph/normal.pro
./languages/Term/graph/search.pro
./languages/Text/textEq.pro
./languages/Text/textLoc.pro
./languages/ueber/macros/bgl-and-bsl.pro
./languages/ueber/macros/bmpl.pro
./languages/ueber/macros/egl.pro
./languages/ueber/macros/esl.pro
./languages/ueber/macros/forall.pro
./languages/ueber/macros/fxy.pro
./languages/ueber/macros/graph.pro
./languages/ueber/macros/include.pro
./languages/ueber/macros/lal.pro
./languages/ueber/macros/mml.pro
./languages/ueber/macros/parse.pro
./languages/ueber/macros/pickyParse.pro
./languages/ueber/macros/ppl.pro
./languages/ueber/macros/rules.pro
./languages/ueber/macros/test.pro
./languages/ueber/ueberApply.pro
```

```
./languages/ueber/macros/test.pro
./languages/ueber/ueberApply.pro
./languages/ueber/ueberDispatch.pro
./languages/ueber/ueberDump.pro
./languages/ueber/ueberEq.pro
./languages/ueber/ueberGraphFunctions.pro
./languages/ueber/ueberGraphLanguages.pro
./languages/ueber/ueberIO.pro
./languages/ueber/ueberMain.pro
./languages/ueber/ueberNorm.pro
./languages/ueber/ueberOk.pro
./languages/ueber/ueberReport.pro
./languages/ueber/ueberSub.pro
./languages/ueber/ueberVerify.pro
./lib/Prolog/assoc.pro
./lib/Prolog/data.pro
./lib/Prolog/dynamic.pro
./lib/Prolog/generics.pro
./lib/Prolog/higher-order.pro
./lib/Prolog/http.pro
./lib/Prolog/io.pro
./lib/Prolog/logvars.pro
./lib/Prolog/multifile.pro
```

```
./languages/ueber/ueberMain.pro
./languages/ueber/ueberNorm.pro
./languages/ueber/ueberOk.pro
./languages/ueber/ueberReport.pro
./languages/ueber/ueberSub.pro
./languages/ueber/ueberVerify.pro
./lib/Prolog/assoc.pro
./lib/Prolog/data.pro
./lib/Prolog/dynamic.pro
./lib/Prolog/generics.pro
./lib/Prolog/higher-order.pro
./lib/Prolog/http.pro
./lib/Prolog/io.pro
./lib/Prolog/logvars.pro
./lib/Prolog/multifile.pro
./lib/Prolog/scanning.pro
./lib/Prolog/strategyBasics.pro
./lib/Prolog/strategySchemes.pro
./lib/Prolog/syntax.pro
./lib/Prolog/test.pro
./lib/Prolog/ueber.pro
./main.pro
```

SWI Prolog

# Discussion

- Many roads to Rome — this is **one** SLE treatment
- Focus on late BSc, early MSc + further reading
- Bits of PLT, CC, MDE — where needed
- Too many technology options — we opt for Haskell
- Prolog would be easier — if readers were willing

***Thanks!***

**Please get in touch, if you like to be an early "adopter".**

Ralf Lämmel: "***Software Languages. A slightly Haskell-biased introduction***".
Under consideration by Springer.
To appear in 2017.

http://www.softlang.org/book