

# Introduction to ASF+SDF

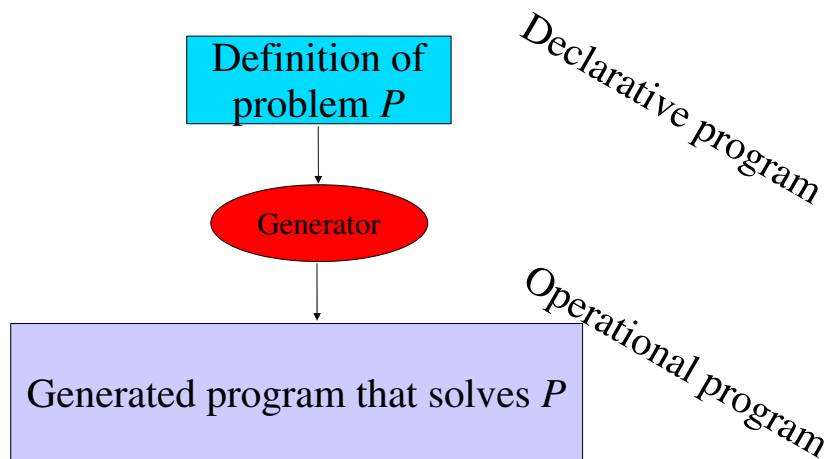
Mark van den Brand  
Paul Klint  
Jurgen Vinju



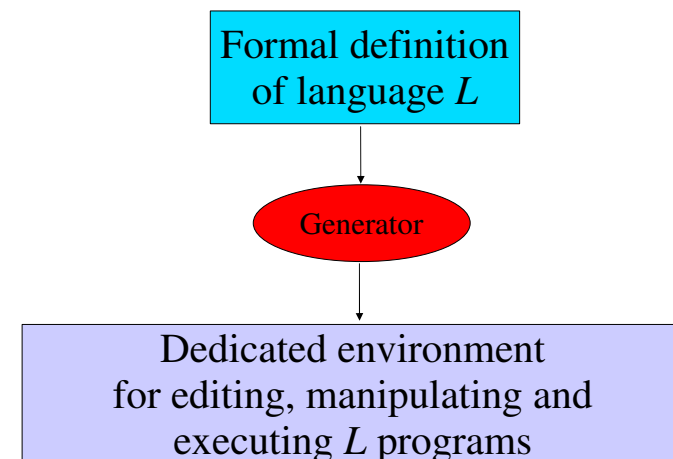
# ASF+SDF

- Goal: defining languages & manipulating programs
- SDF: Syntax definition Formalism
  - lexical & context-free syntax
- ASF: Algebraic Specification Formalism
  - static & dynamic semantics; fact extraction
- ASF+SDF Meta-Environment: IDE for ASF+SDF
- Manuals/documentation: [www.meta-environment.org](http://www.meta-environment.org)

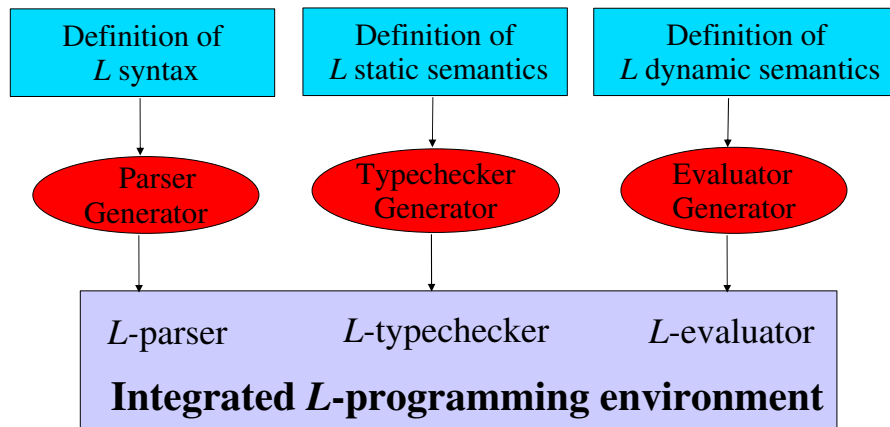
## What is a Program Generator?



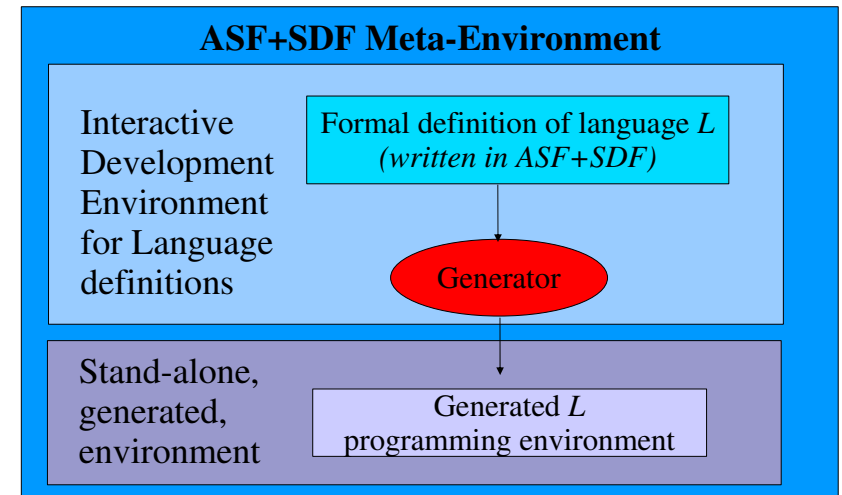
## Programming Environment Generator



# Programming Environment Generator = collection of program generators



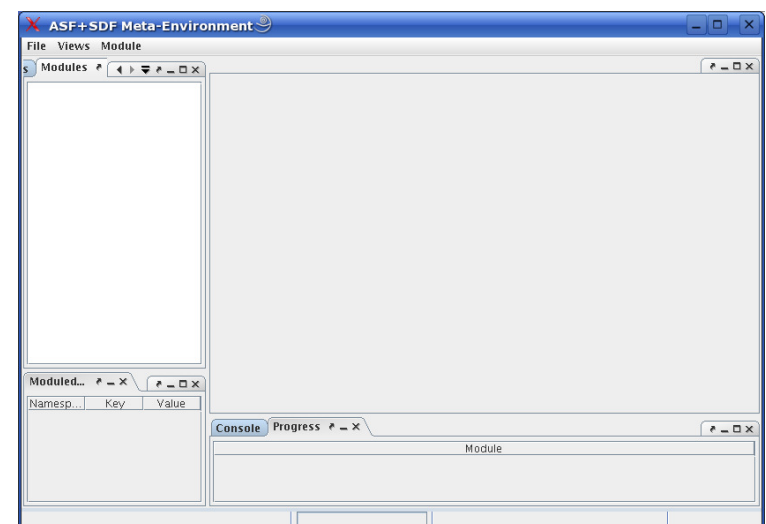
# ASF+SDF Meta-Environment



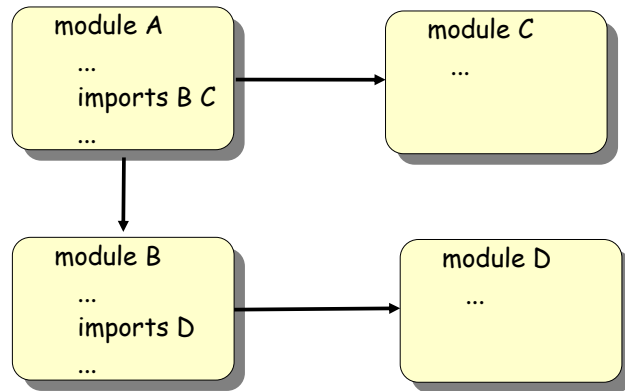
# ASF+SDF Meta-Environment

- An interactive development environment for generating tools from formal language definitions
- Based on:
  - Full context-free grammars
  - Conditional term rewriting
- Language definitions written in ASF+SDF
  - SDF: Syntax definition Formalism
  - ASF: Algebraic Specification Formalism

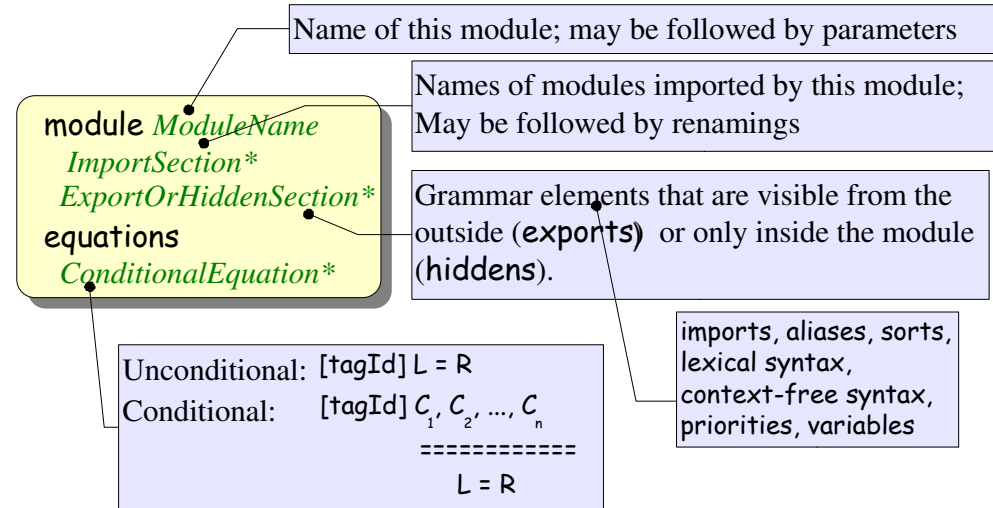
# Typing asfsdf-meta ...



# Anatomy of ASF+SDF specifications



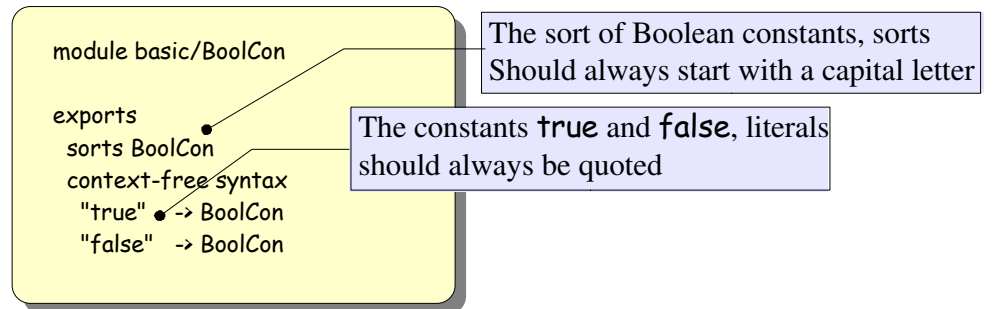
# Anatomy of an ASF+SDF Module



# Plan

- **Booleans**
- **Steps towards a Pico environment**
  - Step 1: define syntax
  - Step 2: define a typechecker
  - Step 3: define an evaluator
  - Step 4: define a compiler
- **Traversal functions**
  - Step 5: define a fact extractor

# BoolCon: Boolean Constants



## Booleans (1)

```
module basic/Booleans
imports basic/BoolCon
exports
  sorts Boolean
  context-free syntax
  BoolCon -> Boolean
```

Import Boolean constants

The sort of Boolean expressions

Each Boolean constant is a Boolean expression, also-called **injection rule** or **chain rule**

## Booleans (2)

```
Boolean "|" Boolean -> Boolean {left}
Boolean "&" Boolean -> Boolean {left}
"not" (Boolean) -> Boolean
 "(" Boolean ")" -> Boolean {bracket}

context-free priorities
Boolean "&" Boolean -> Boolean >
Boolean "|" Boolean -> Boolean
```

The infix operators and **&** and or **|**. Both are left-associative (**left**)

The prefix function **not**

( and ) may be used as brackets in Boolean expressions; they are ignored after parsing

**&** has higher priority than **|**

Example:

**Bool & Bool | Bool**  
is interpreted as:  
**(Bool & Bool) | Bool**

## Booleans (3)

```
context-free syntax
"not" (Boolean) -> Boolean
 "(" Boolean ")" -> Boolean {bracket}
```

```
context-free priorities
Boolean "&" Boolean -> Boolean {left} >
Boolean "|" Boolean -> Boolean {left}
```

**Shorthand** for defining the infix operators and (**&**) and (**|**). Both are left-associative (**left**). These rules are promoted to context-free syntax rules

## Booleans (4)

```
hiddens
context-free start-symbols
  Boolean
```

```
imports
  basic/Comments
```

```
variables
  "Bool"[0-9\']* -> Boolean
```

The start symbol of a grammar. Without a start symbol the parser does not know how to start parsing an input sentence

Import the standard comment conventions for equations

Declares the variables **Bool**, **Bool1**, **Bool2**, **Bool'**, **Bool''**, **Bool1'**, etc.