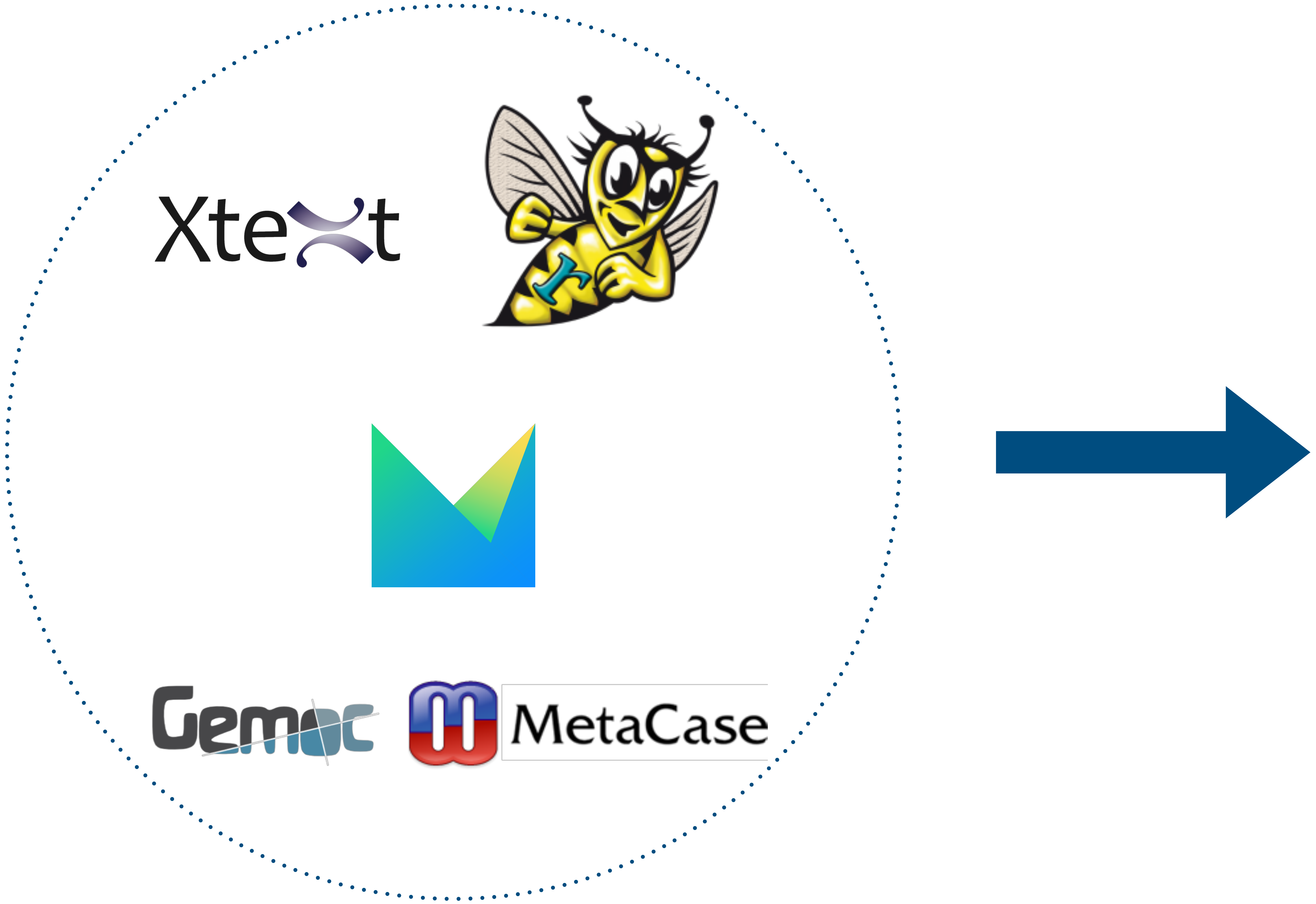
MSc assignment: Projectional Editing support for Textual Languages

Mauricio Verano Merino

14th of February 2019

Context

Language Workbenches



Language

User Interface
Debuggers
Processors
Type Checkers
Code Generators
...

Rascal

http://www.rascal-mpl.org

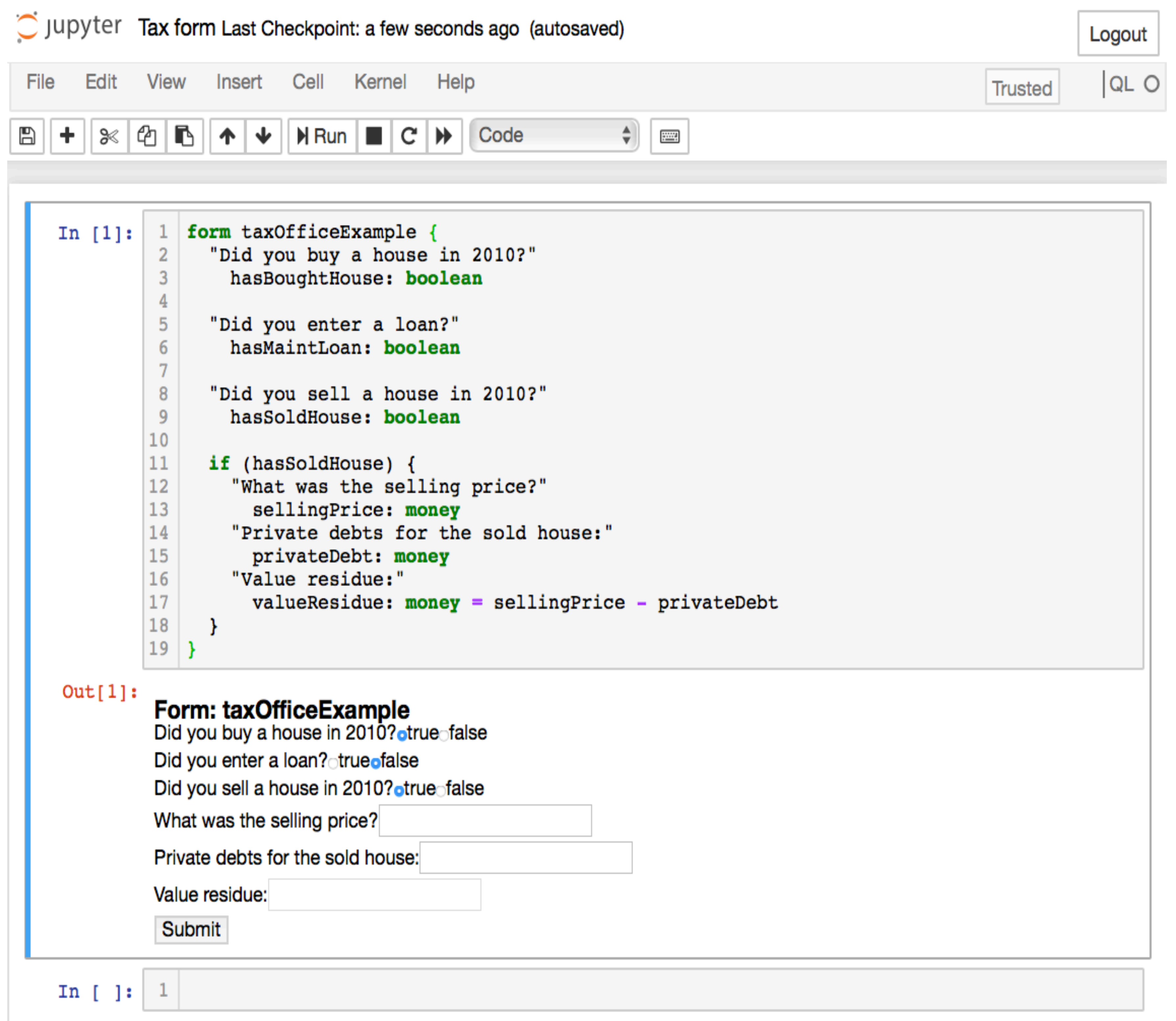
- Meta-programming language for source code analysis and transformation.
- Language workbench for DSL development.
- Successfully used in various domains (e.g., finance, digital forensics)
- Command line Read-Eval-Print Loop (REPL) + Eclipse IDE.

Bacatá

A language parametric Jupyter notebook generator for DSLs written using Rascal language workbench, which reuses language components.

Objectives:

- Open up the interactive notebook metaphor for DSLs
- Extend current set of generated IDE services of language workbenches
- Generate DSL notebooks with minimum effort



In [1]:
```
form taxOfficeExample {
  "Did you buy a house in 2010?"
    hasBoughtHouse: boolean

  "Did you enter a loan?"
    hasMaintLoan: boolean

  "Did you sell a house in 2010?"
    hasSoldHouse: boolean

  if (hasSoldHouse) {
    "What was the selling price?"
      sellingPrice: money
    "Private debts for the sold house:"
      privateDebt: money
    "Value residue:"
      valueResidue: money = sellingPrice - privateDebt
  }
}
```

Out[1]:

Form: taxOfficeExample

Did you buy a house in 2010? [✓ true] false

Did you enter a loan? true [✓ false]

Did you sell a house in 2010? [✓ true] false

What was the selling price?

Private debts for the sold house:

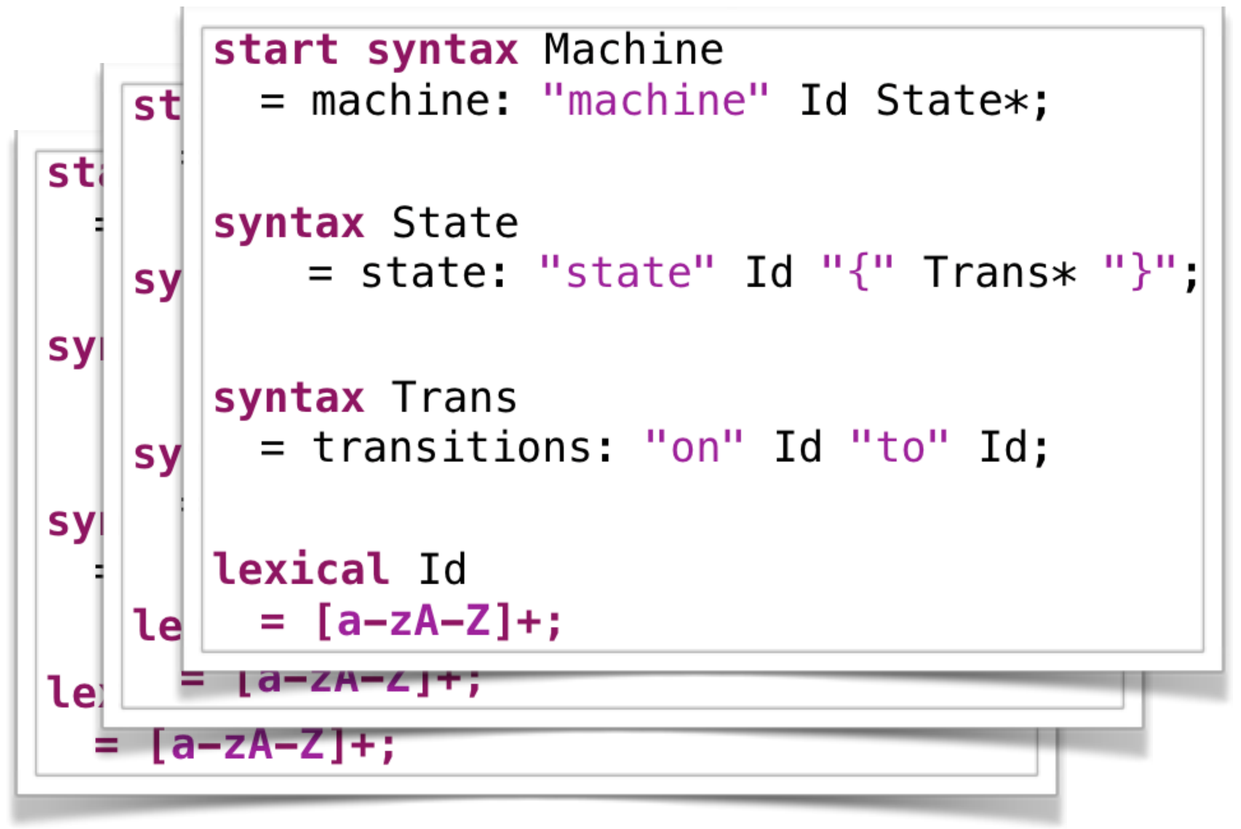
Value residue:

Submit

In [ ]:

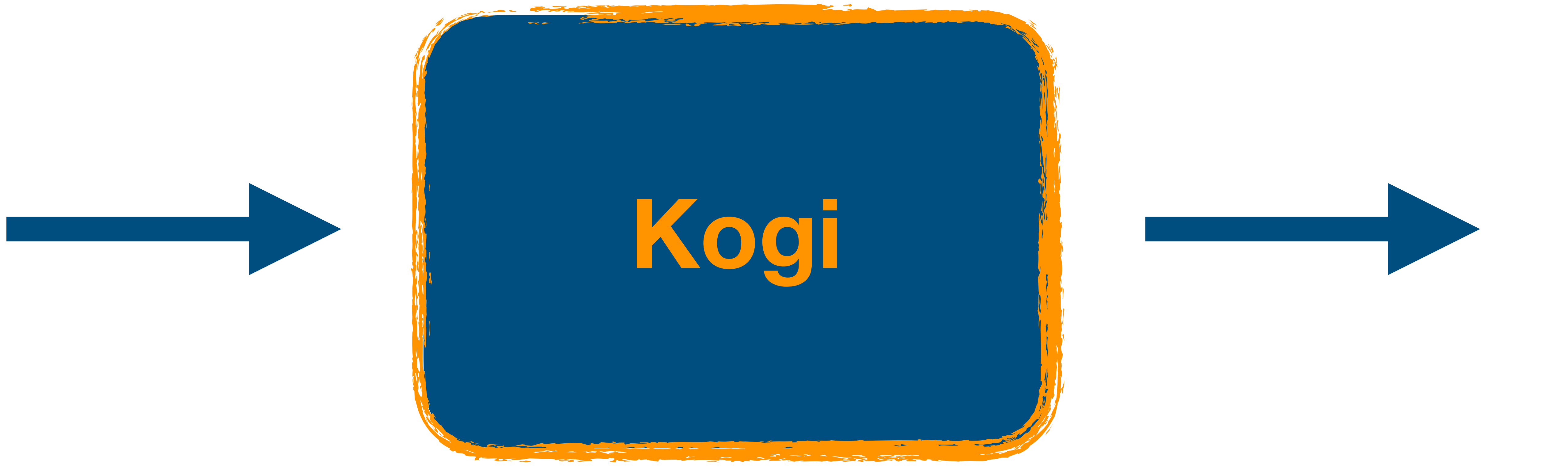
Language Workbench Support for Block-based Environments

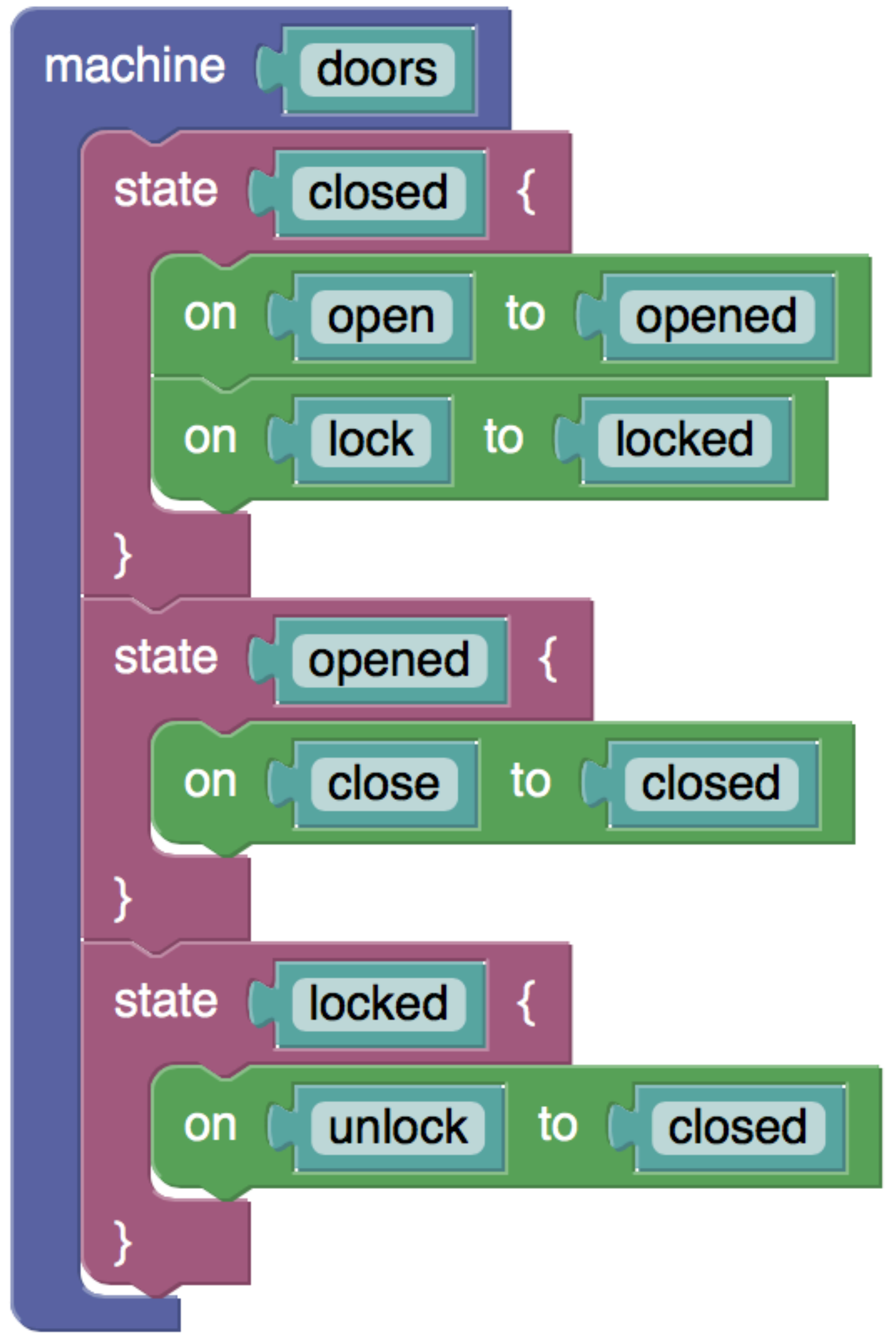
Tool for deriving block-based environments from context-free grammars.



```
start syntax Machine
  = machine: "machine" Id State*;

syntax State
    = state: "state" Id "{" Trans* "}";

syntax Trans
  = transitions: "on" Id "to" Id;

lexical Id
  = [a-zA-Z]+;
```

Context-free grammars





Projectional Editing support for Textual Languages

Textual Languages
(Rascal)
Grammars
Projectional Editors
(MPS)
Concepts
Aspects

Language A
Abstract Syntax
Parser
Type checker
Concrete Syntax
Interpreter
Program A
Textual
Projectional
Editor
Constraints
Transformations
Structure
Type system
Model A
Language A’

Projectional Editing support for Textual Languages

➡**Goal**

- Explore a way to enable interoperability between textual and projectional language workbenches.

➡**Exploratory research**

- Characterize constructs of projectional editors
- Define a possible mapping between textual and projectional language artefacts.
- Manipulate MPS models somehow API-wise outside MPS

➡**Case study** (Océ)

- Small configuration DSL