Введение в MPS

Универсальные и специализированные языки программирования

- Универсальные языки недостаточно выразительны

Универсальные и специализированные языки программирования

- Универсальные языки недостаточно выразительны

```
public void method() {
  apply(1, new _FunctionTypes.return_P1_E0<Integer, Integer>() {
    public Integer invoke(Integer a) {
      return a + 1;
    }
  });
}

public void apply(int data, _FunctionTypes.return_P1_E0<? extends Integer,
                                                        ? super Integer> func) {
  func.invoke(data);
}
```

Универсальные и специализированные языки программирования

- Универсальные языки недостаточно выразительны

```
public void method() {
  apply(1, {int a => a + 1;});
}
```

```
public void apply(int data, {int => int} func) {
  func(data);
}
```

Универсальные и специализированные языки программирования

- Универсальные языки недостаточно выразительны

```
if (repository != null) {
  List<DevKit> devkits = repository.getAllDevkits();
  if (devkits != null) {
    devkits.add(new DevKit());
  }
}
```

Универсальные и специализированные языки программирования

- Универсальные языки недостаточно выразительны

```
repository.?getAllDevkits().?add(new DevKit());
```

Универсальные и специализированные языки программирования

- Универсальные языки недостаточно выразительны
- Все новые конструкции ограничены синтаксисом универсального языка

Универсальные и специализированные языки программирования

- Универсальные языки недостаточно выразительны
- Все новые конструкции ограничены синтаксисом универсального языка
- Предметную область (domain) приходится транслировать на выразительные средства языка

Универсальные и специализированные языки программирования

- Универсальные языки недостаточно выразительны
- Все новые конструкции ограничены синтаксисом универсального языка
- Предметную область (domain) приходится транслировать на выразительные средства языка
- Специализированные языки (DSL) позволяют описывать предметную область в ее терминах

Специализированные языки (DSL)

- Хорошо описывают предметную область

Специализированные языки (DSL)

- Хорошо описывают предметную область
- Требуют интеграции с каким-либо универсальным языком

Специализированные языки (DSL)

- Хорошо описывают предметную область
- Требуют интеграции с каким-либо универсальным языком
- Сложно комбинировать разные DSL в одном приложении

LOP - "традиционный" подход

(Language Oriented Programming)

Исходный текст

Лексический анализ

Парсер (синтаксический анализ)

Компилятор (семантика)

LOP - "традиционный" подход

```
if (a == b) {
  a++;
}
```

Исходный текст

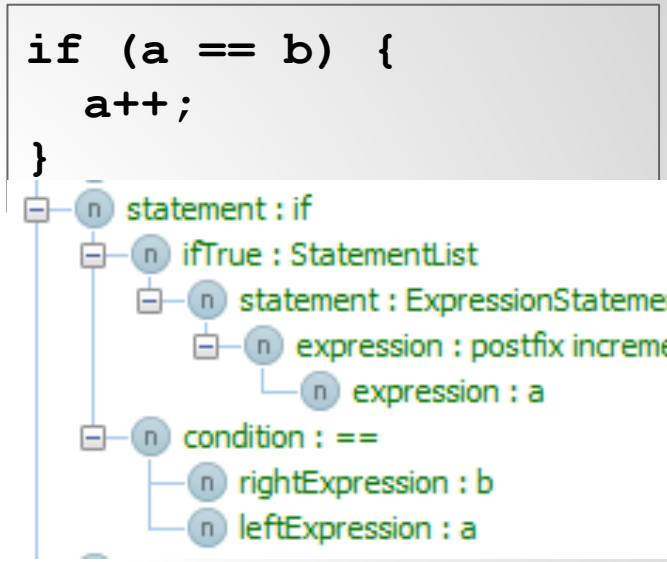
Лексический анализ

Парсер (синтаксический анализ)

Компилятор (семантика)

LOP - "традиционный" подход

Исходный текст



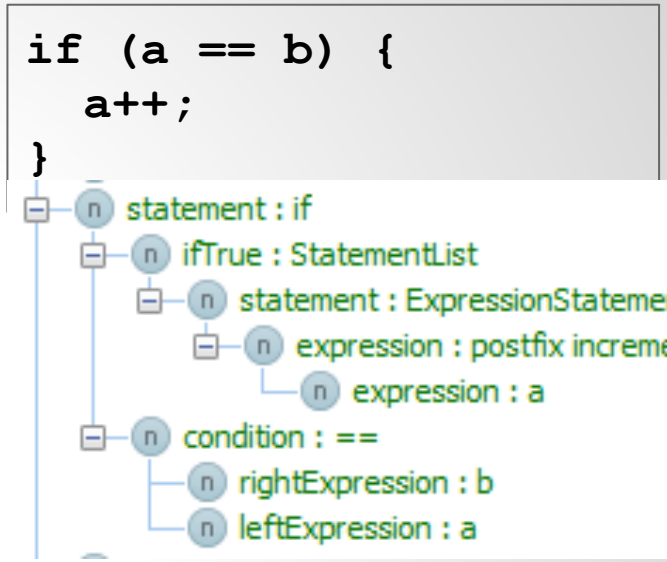
Лексический анализ

Парсер (синтаксический анализ)

Компилятор (семантика)

LOP - "традиционный" подход

Исходный текст



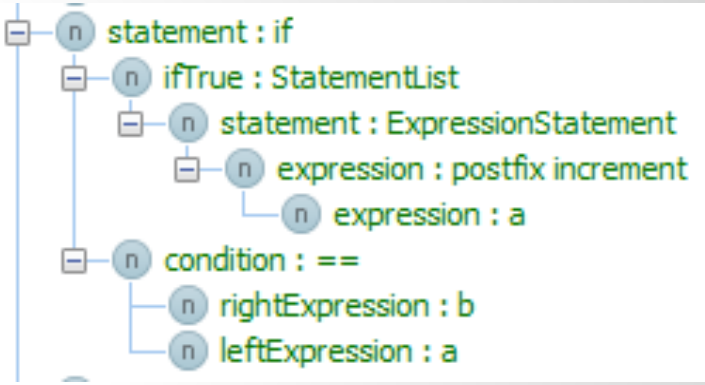
Лексический анализ

Парсер (синтаксический анализ)

Компилятор (семантика)

```
load a
load b
cmp
…
```

MPS



Модель (AST)

↓

Генератор

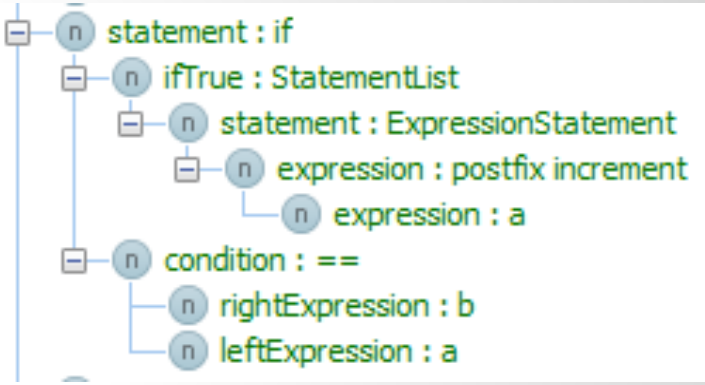
↓

Код

```
load a
load b
cmp
…
```

MPS

Проекционный редактор



Модель (AST)

↓

Генератор

↓

Код

```
load a
load b
cmp
…
```

Проблемы, решенные в MPS

- Смешивание языков

```
L1:     "1 + 2 = {1+2}"
L2:     "1 + 2 = ${1+2}"
L1+L2:
 "balance is ${acc.getBalance()}"
```

Проблемы, решенные в MPS

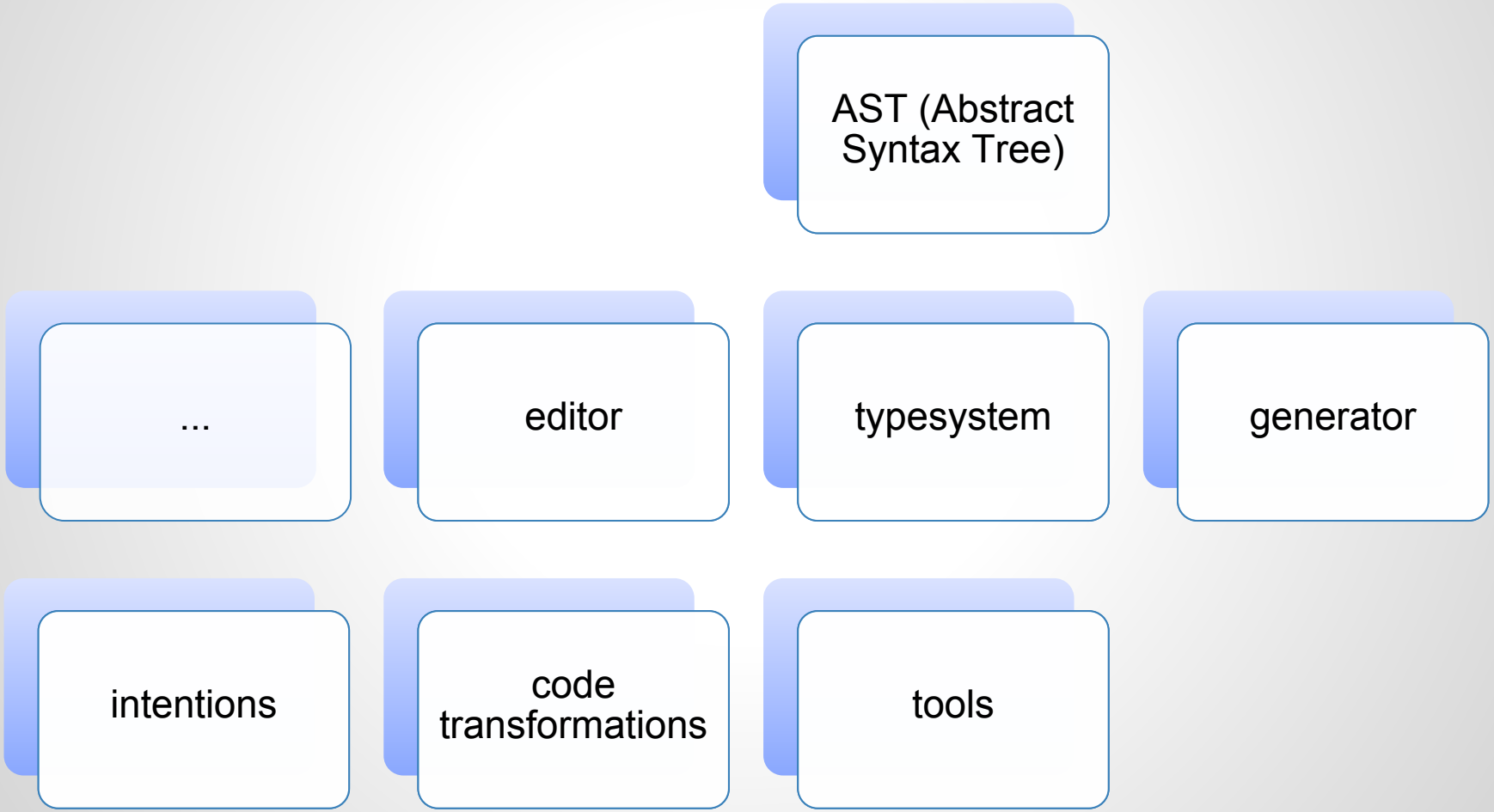
- Смешивание языков
- Набор инструментов
  - редактор
  - code completion
  - поиск
  - рефакторинги
  - intentions
  - ...

Проблемы, решенные в MPS

- Смешивание языков
- Набор инструментов
- Быстрая разработка нового языка вместе с инструментарием

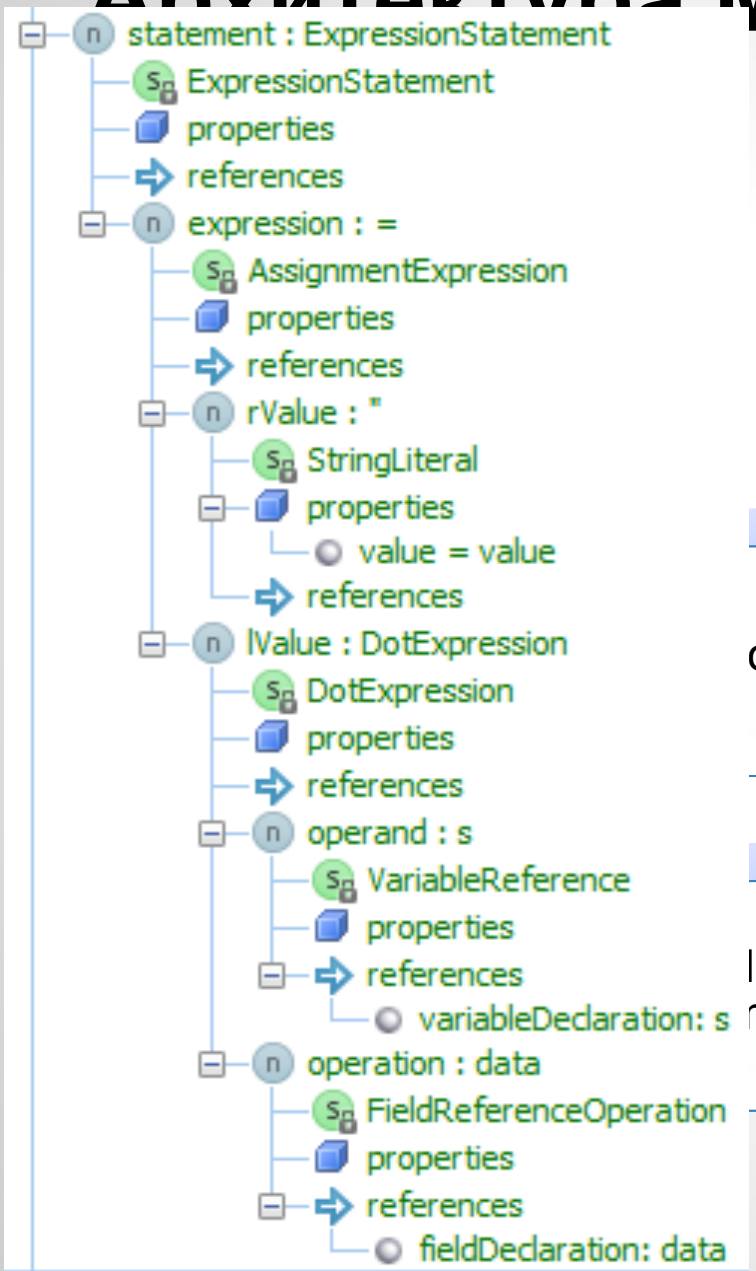
Архитектура MPS

```
s.data = "value"
```



Архитектура MPS

```
s.data = "value"
```



AST (Abstract Syntax Tree)

or

typesystem

generator

le
nations

tools

Архитектура MPS

```
s.data = "value"
```

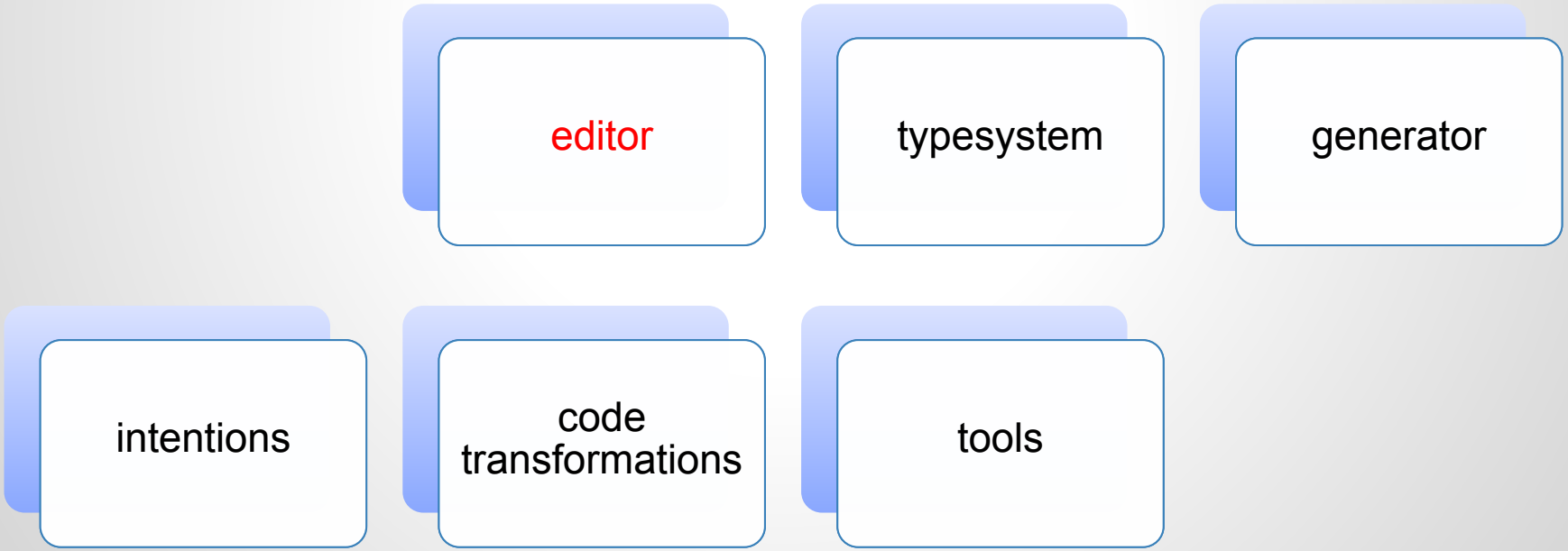
AST (Abstract Syntax Tree)

```
abstract concept BaseAssignmentExpression extends     Expression
                                          implements  <none>

  instance can be root: false
  alias: <no alias>
  short description: <no short description>

  properties:
  << ... >>

  children:
  lValue : Expression[1]
  rValue : Expression[1]

  references:
  << ... >>
```

Архитектура MPS

```
s.data = "value"
```

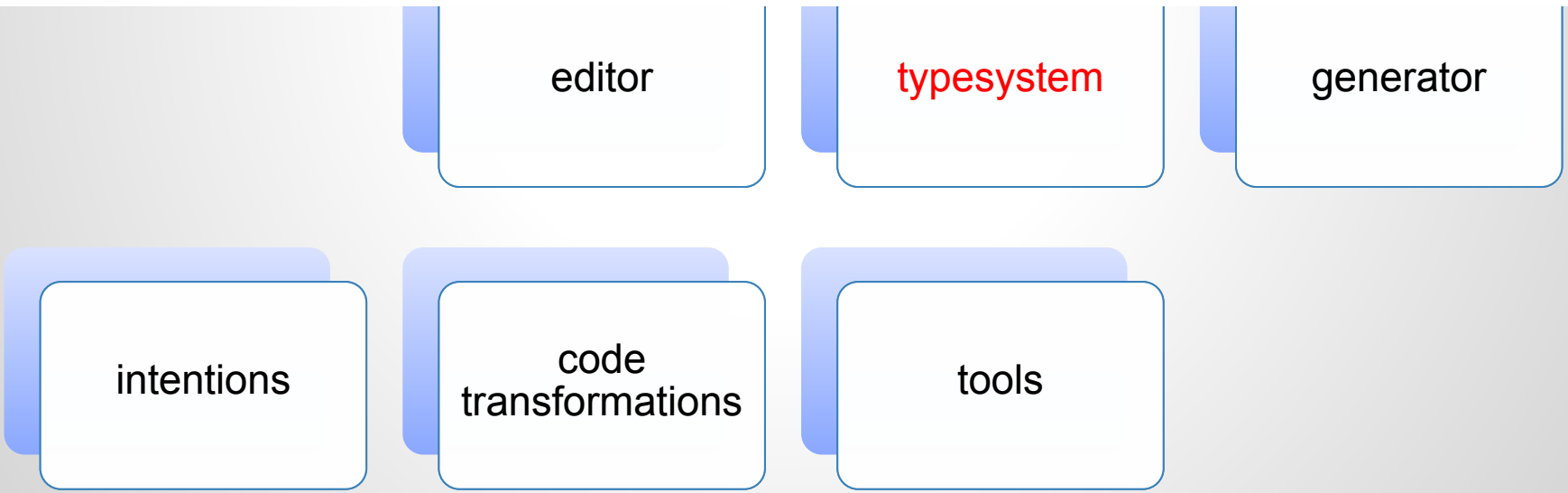
```
<default> editor for concept BaseAssignmentExpression
  node cell layout:
    [- ^% lValue % ^# alias # ^% rValue % -]

  inspected cell layout:
    <choose cell model>
```



Архитектура MPS

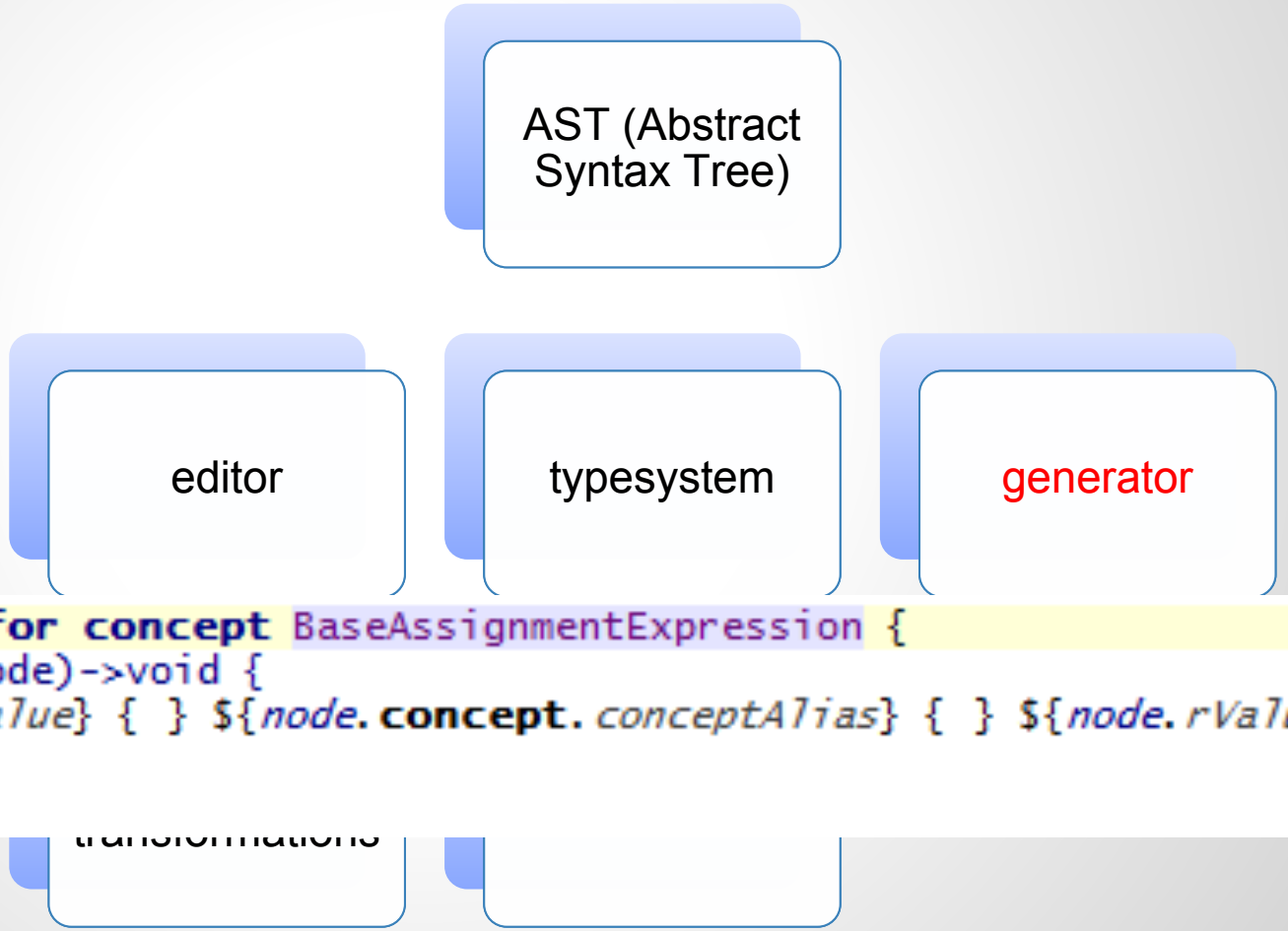
```
s.data = "value"
```

```
rule typeof_BaseAssignmentExpression {
  applicable for concept = BaseAssignmentExpression as baseAssignmentExpression
  overrides false

  do {
    node<BaseAssignmentExpression> ae = baseAssignmentExpression;
    node<Expression> lval = ae.lValue;
    node<Expression> rval = ae.rValue;
    check(typeof(rval) :<=: typeof(lval));
    typeof(ae) :==: typeof(lval);
  }
}
```



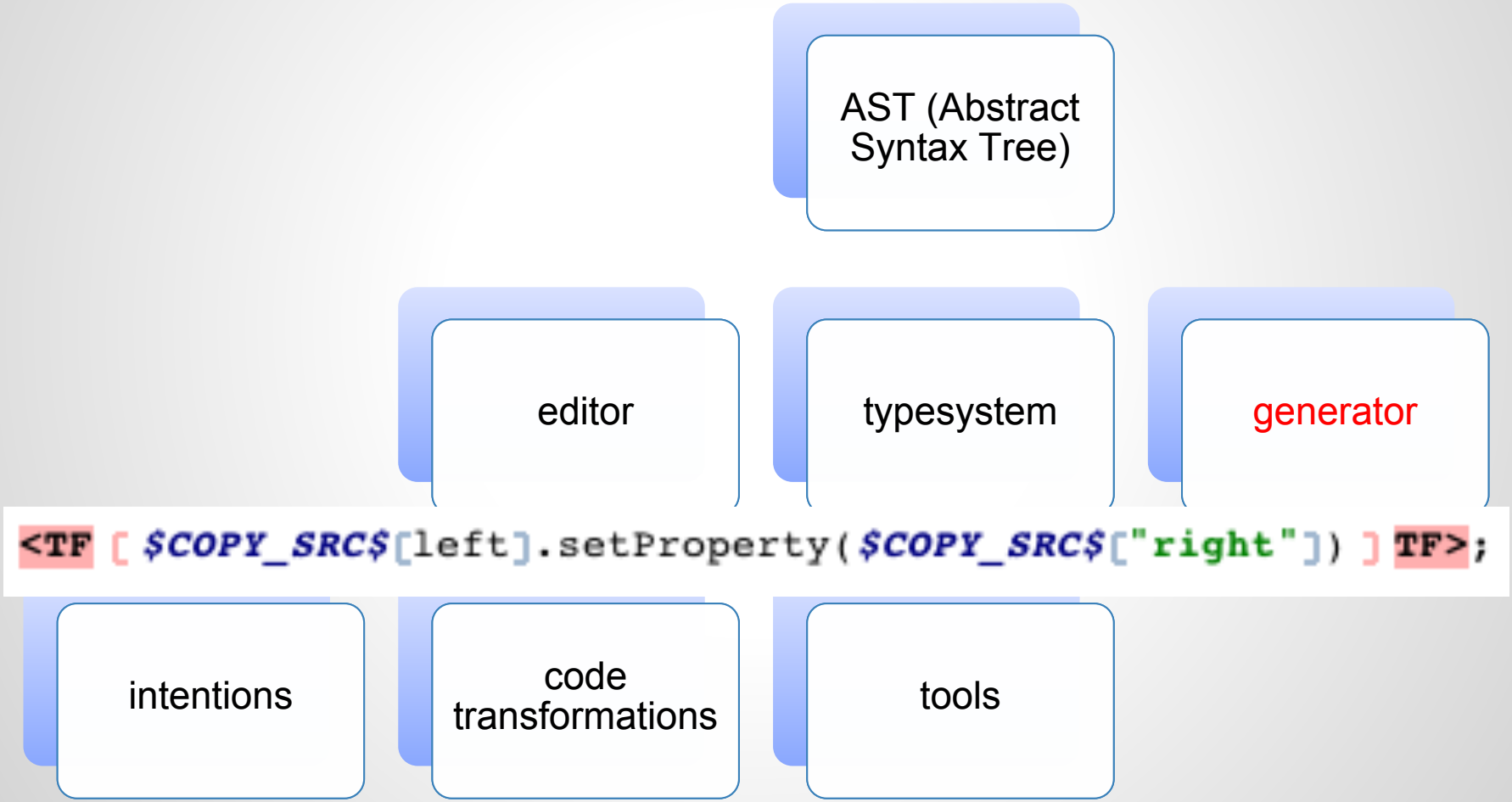
Архитектура MPS

```
s.data = "value"
```



Архитектура MPS

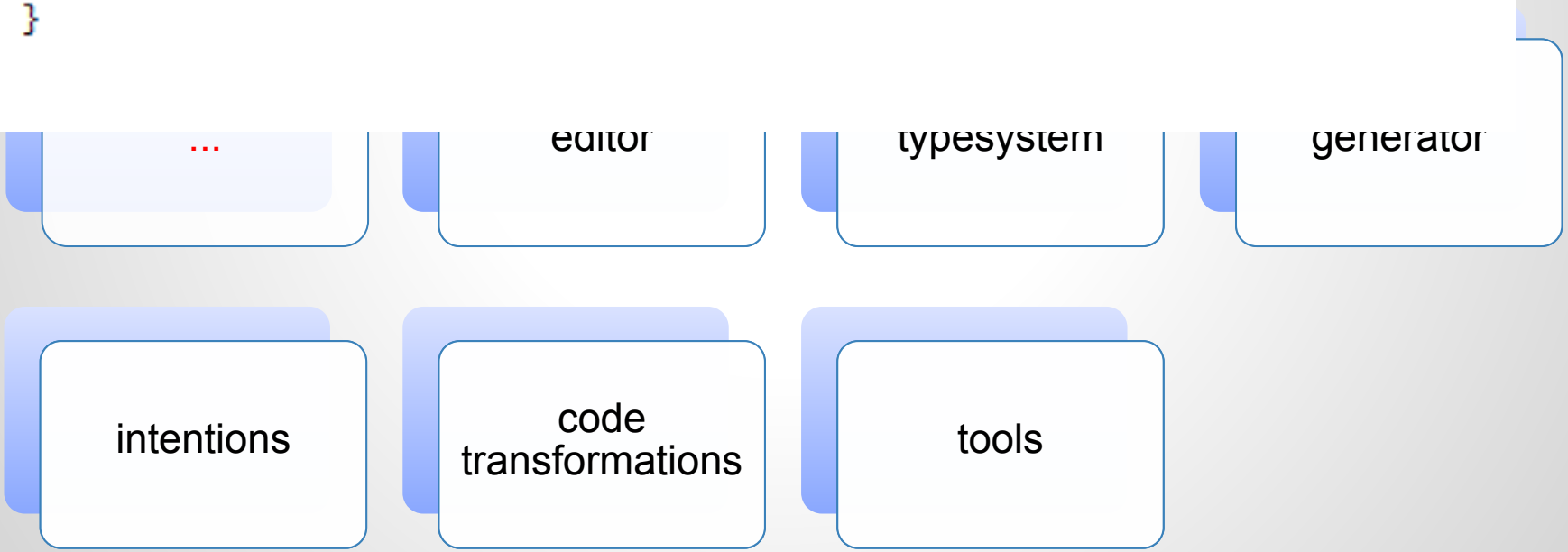
```
s.data = "value"
```



Архитектура MPS

```
s.data = "value"
```

```
data flow builder for BaseAssignmentExpression {
  (node)->void {
    if (node.lValue.isInstanceOf(VariableReference)) {
      if (node.isReadAsignment()) {
        read node.lValue : VariableReference.variableDeclaration
      }
      code for node.rValue
      write node.lValue : VariableReference.variableDeclaration = <unknown>
    } else {
      code for node.rValue
      code for node.lValue
    }
  }
}
```



Примеры языков, реализованных в MPS

- closures
- collections
- dates
- checkedDots
- math
- tuples
- regexp
- ...

Примеры языков, реализованных в MPS

- closures
- collections
- dates
- checkedDots
- math
- tuples

```
public void method( ) {
  apply(1, {int a => a + 1; });
}

public void apply(int data, {int => int} func) {
  func(data);
}
```

Примеры языков, реализованных в MPS

- closures
- collections
- dates
- checkedDots
- math
- tuples
- regexp

```
list<int> data = new arraylist<int>;
int selection = data.where({~it => it > 1; }).select({~it => 2 * it; }).first;
```

Примеры языков, реализованных в MPS

- closures
- collections
- dates
- checkedDots
- math
- tuples
- regexp

```
datetime d = 30 November 2011 19:00:00 in Europe/Tallinn;
System.out.println(d #{date: <fullDate>});
```

Примеры языков, реализованных в MPS

- closures
- collections
- dates
- checkedDots
- math
- tuples
- regexp

```
Object o = null;
Object clone = o.?clone( ).?equals("abc");
```

Примеры языков, реализованных в MPS

- closures
- collections
- dates
- checkedDots
- math

```
int j = 10;

                  j
int sum =  ∑   [[k + 1   k * 2]]  *  [2]
          k = 1 [[exp(k)  ln(k)]]    [j];
```

Примеры языков, реализованных в MPS

- closures
- collections
- dates
- checkedDots
- math
- tuples

```
public HashMap<Integer, [int, int]> a = new HashMap<Integer, [int, int]>( );
<<properties>>
<<initializer>>
public Fubar( ) {
  this.a.put(1, [42, -1]);
}

<<methods>>
public static void main(String[] args) {
  System.out.println(new Fubar( ).a.get(1)[0]);
}
```

Примеры языков, реализованных в MPS

- closures
- collections
- dates
- checkedDots
- math
- tuples
- regexp
-

```
/*package*/ void forEachMatch( ) {
  string test = "123 456 789";
  while (test =~ /(number: \d+)/) {
    System.out.println("number = " + number);
  }
}
```

MPS

- Быстрая разработка языков
- Неограниченное смешивание языков (нет парсера)
- Автоматический "тулинг" – редактор, code completion, refactorings, поиск
- Лаконичное описание системы типов для новых языков
- Кодогенерация по шаблону
- Специальные языки для создания инструментария
- Полная интеграция с системами контроля версий

Пример создания языка lazy

- просто пример, не все проблемы решены
- расширение Java
- "ленивое" вычисление значения переменной - при использовании

```
class Test {
  public static void main(string[] args) {
    int a = 1, b = 2;
    lazy<int> l = a + b;
    a = 100;
    System.out.println(2 * l);
  }
}
```

Пример создания языка lazy

AST

Type ::= **lazy**<Type>

Editor

Представление:

lazy< Type >

Generator

lazy<T>

lazy<T> x = a+b;

print(x);

Typesystem

lazy<T> :<: T

lazy<T> :>: T

Как начать использовать MPS

- Бесплатное программное обеспечение, распространяется под лицензией Apache 2.0
  http://www.jetbrains.com/mps/
- Полностью открытый исходный код
  https://github.com/JetBrains/MPS