General Rascal Introduction

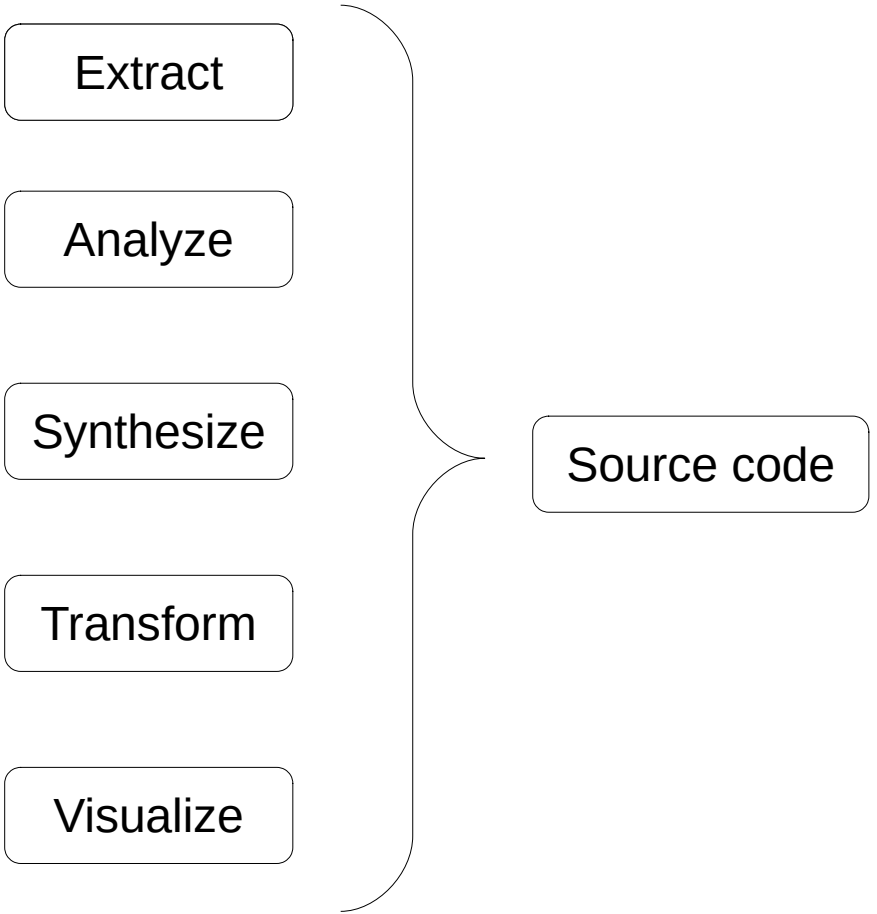
by Simon Baars

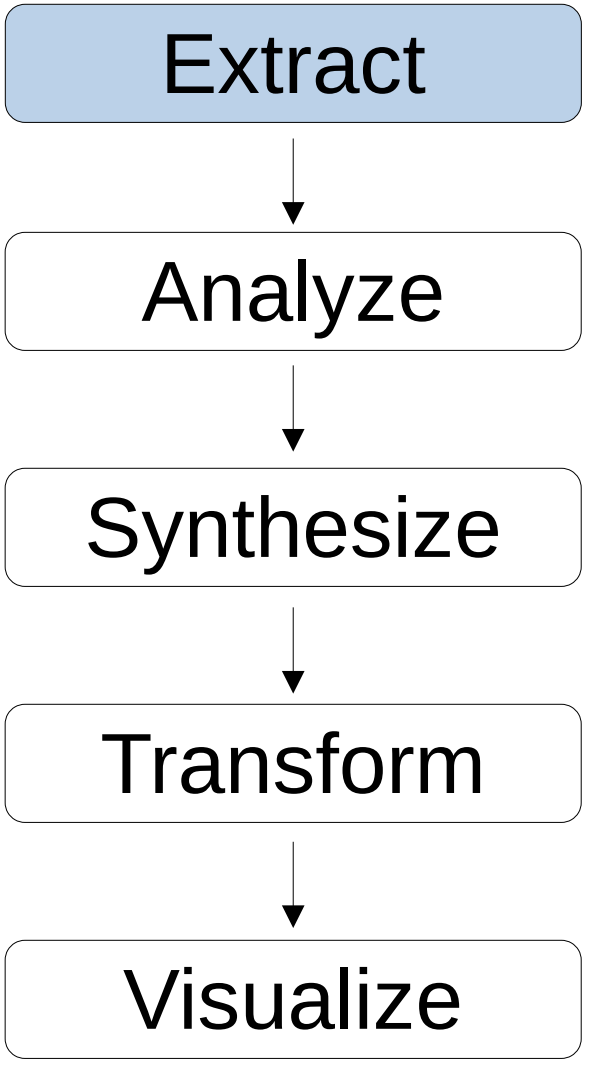
Schedule

- **26 Oct 13:00:** Overview, Installation, Tutor
- **27 Oct 9:30:** Analysis & Series 0
- **27 Oct 16:00:** Series 0 discussion

Installing Rascal

What can Rascal do for us?




Extract
Analyze
Synthesize
Transform
Visualize

```
test bool testSqrt (int randNum) {
    int posNum = abs(randNum);
    if(posNum * posNum == pow(posNum, 2)){
        return true;
    }
    return false;
}
```

```
test bool testSqrt (int randNum) {
    int posNum = abs(randNum);
    if(posNum * posNum == pow(posNum, 2)){
        return true;
    }
    return false;
}
```

= Declaration

```
test bool testSqrt (int randNum) {
    int posNum = abs(randNum);
    if(posNum * posNum == pow(posNum, 2)){
        return true;
    }
    return false;
}
```

```
syntax Declaration
    = Keyword* Type Name "(" {Declaration ","}* ")" Statement
    | Type Name
;
```

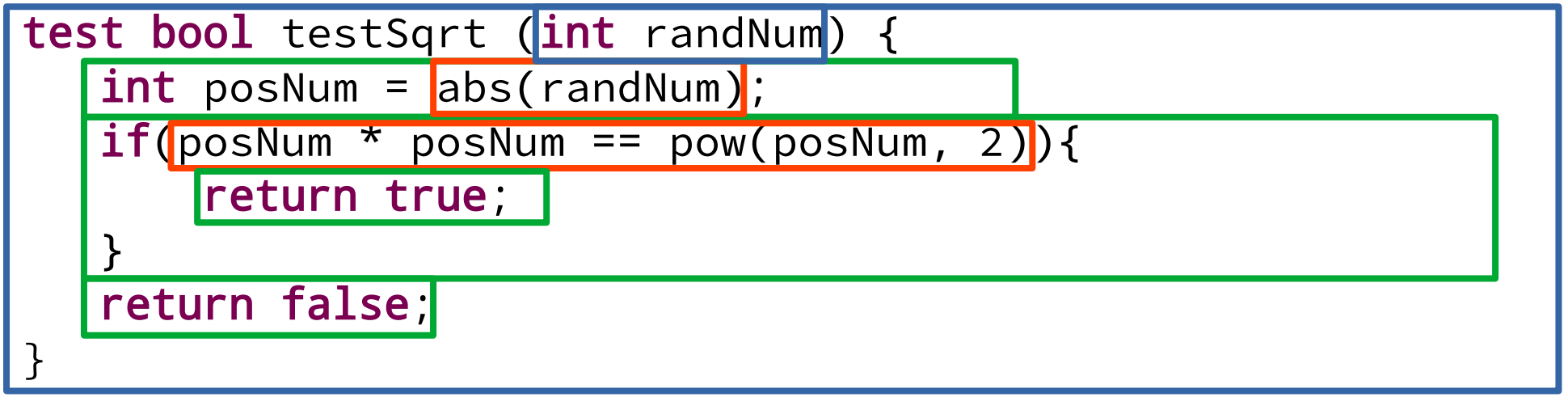
```
test bool testSqrt (int randNum) {
    int posNum = abs(randNum);
    if(posNum * posNum == pow(posNum, 2)){
        return true;
    }
    return false;
}
```

= Declaration

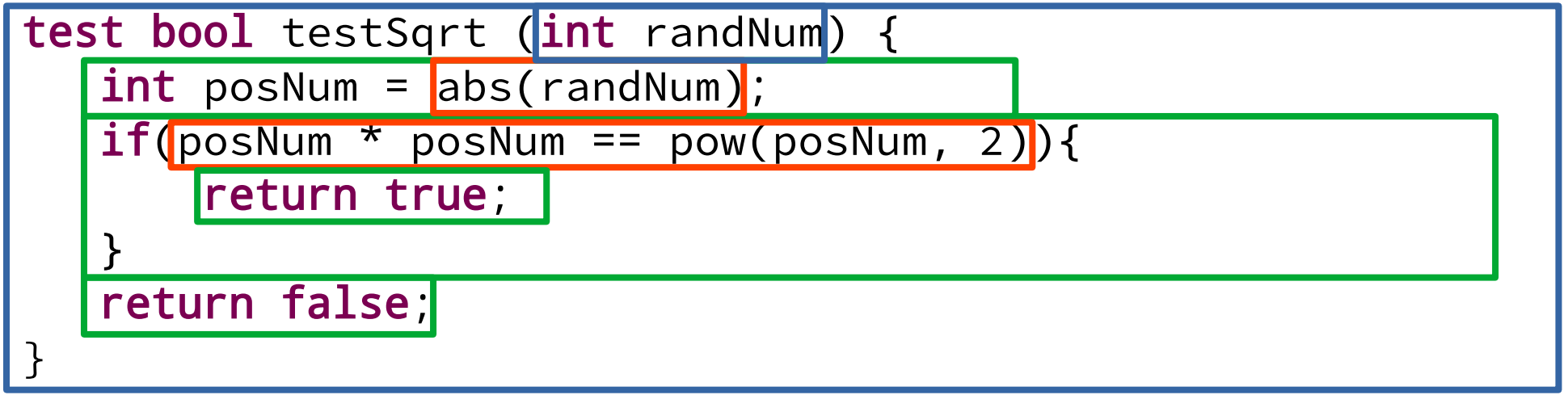
= Expression

```
test bool testSqrt (int randNum) {
    int posNum = abs(randNum);
    if(posNum * posNum == pow(posNum, 2)){
        return true;
    }
    return false;
}
```

```
syntax Expression
    = Name "(" {Expression ","}* ")"
    | Expression Operator Expression
    | Name
    | Literal
;
```

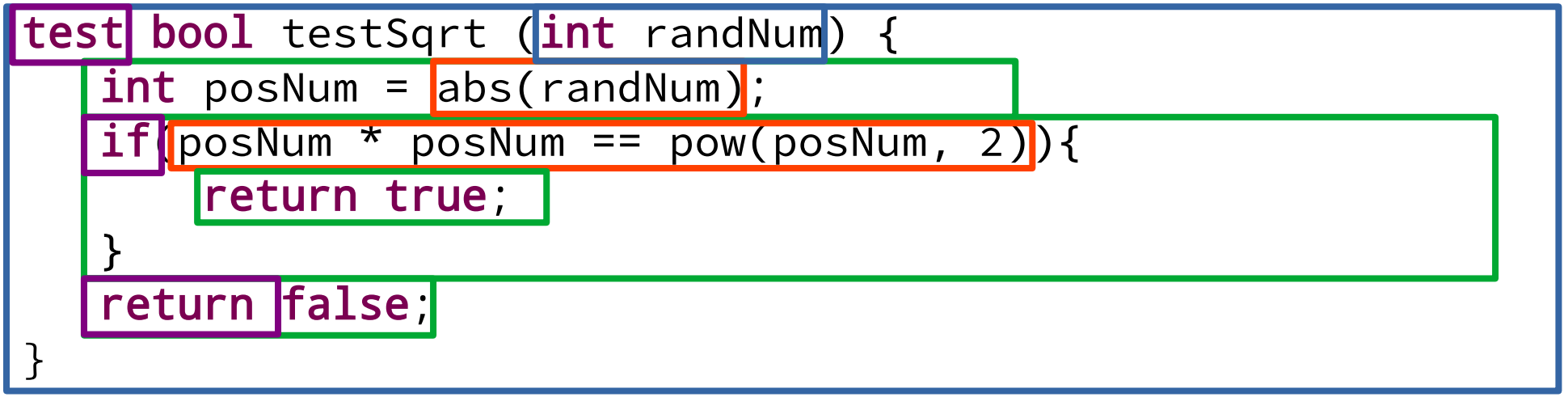


= Declaration = Expression = Statement



```
test bool testSqrt (int randNum) {
    int posNum = abs(randNum);
    if(posNum * posNum == pow(posNum, 2)){
        return true;
    }
    return false;
}
```

```
syntax Statement
   = "{" Declaration* Statement* "}"
   | Expression ";"
   | "if" "(" Expression ")" Statement
   | "if" "(" Expression ")" Statement "else" Statement
   | "return" Expression ";"
   | "int" Name "=" Expression ";"
;
```

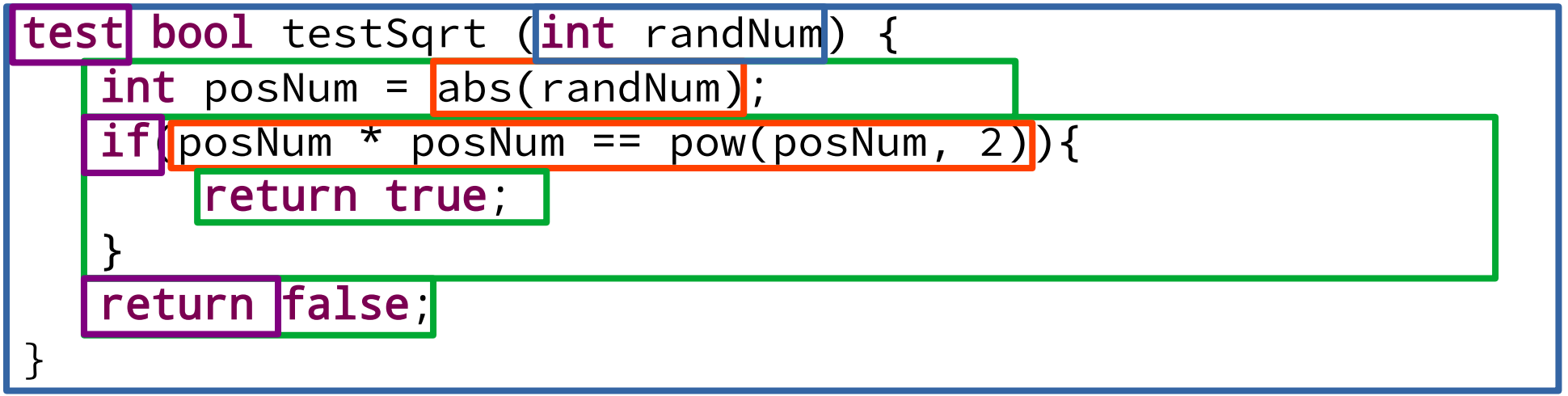


= Declaration

= Expression

= Statement

= Keyword




```
keyword Keyword
    = "test"
    | "if"
    | "return"
;
```

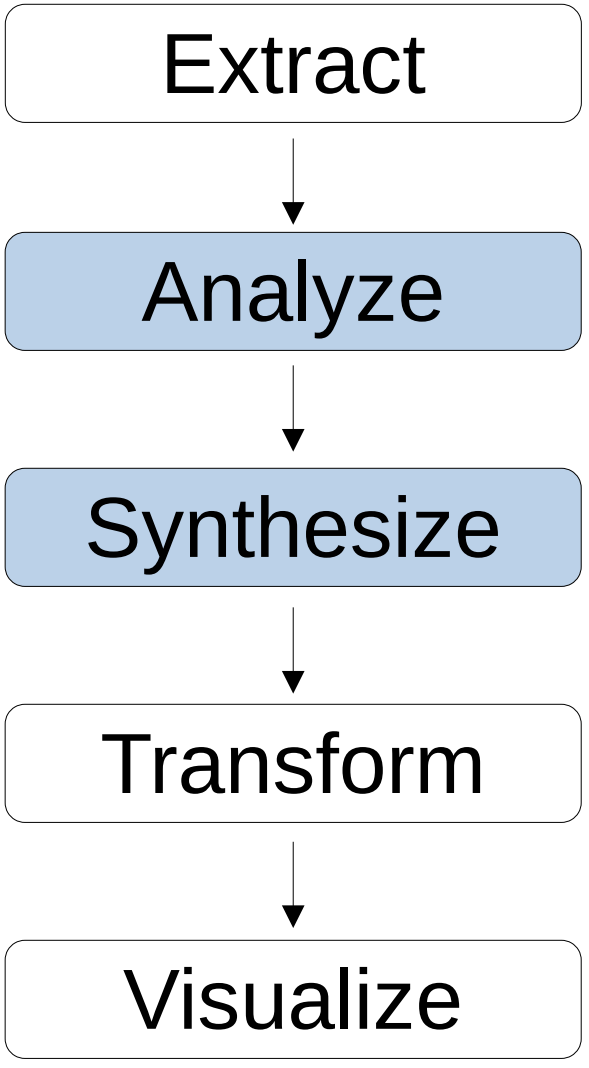
Luckily, many languages are already defined :-)

Loc

http://tutor.rascal-mpl.org/Rascal/Expressions/Values/Location/Location.html#/Rascal/Expressions/Values/Location/Location.html

Node

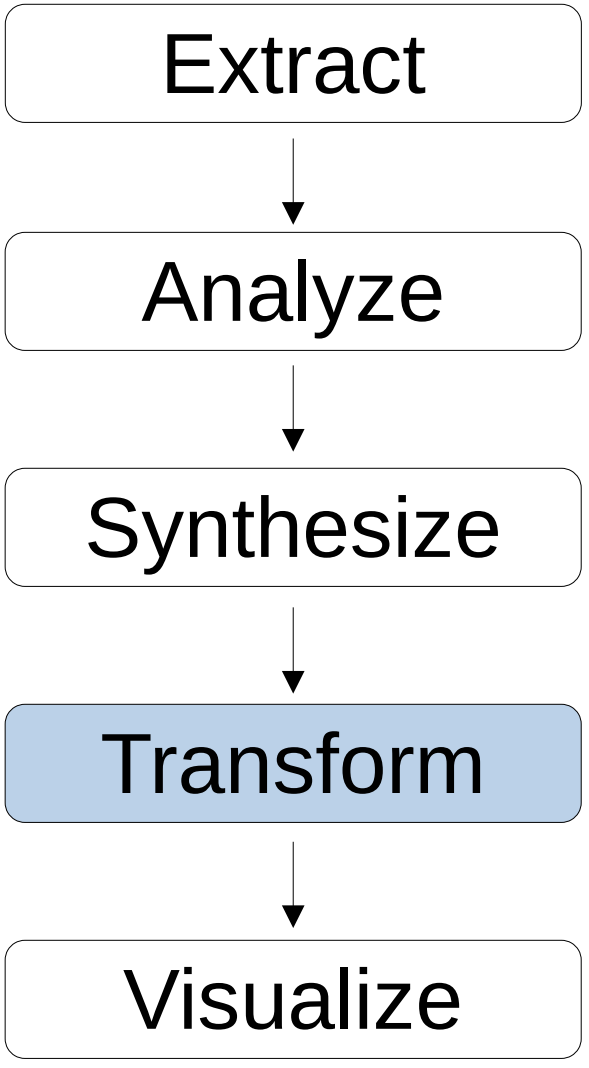
http://tutor.rascal-mpl.org/Rascal/Expressions/Values/Location/Location.html#/Rascal/Expressions/Values/Node/Node.html


Extract
Analyze
Synthesize
Transform
Visualize

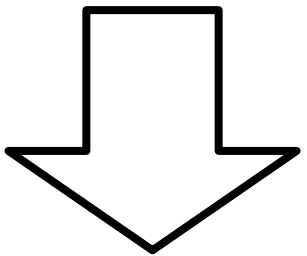
```
test bool testSqrt (int randNum) {
    int posNum = abs(randNum);
    if(posNum * posNum == pow(posNum, 2)){
        return true;
    }
    return false;
}
```

```
int countExpressions(Declaration decl) {
   int counter = 0;
   visit(decl) {
      case Expression e => counter+=1;
   }
   return counter;
}
```

```
int countEverything(Declaration decl) {
    int counter = 0;
    visit(decl) {
        case node e => counter+=1;
    }
    return counter;
}
```


Extract
Analyze
Synthesize
Transform
Visualize

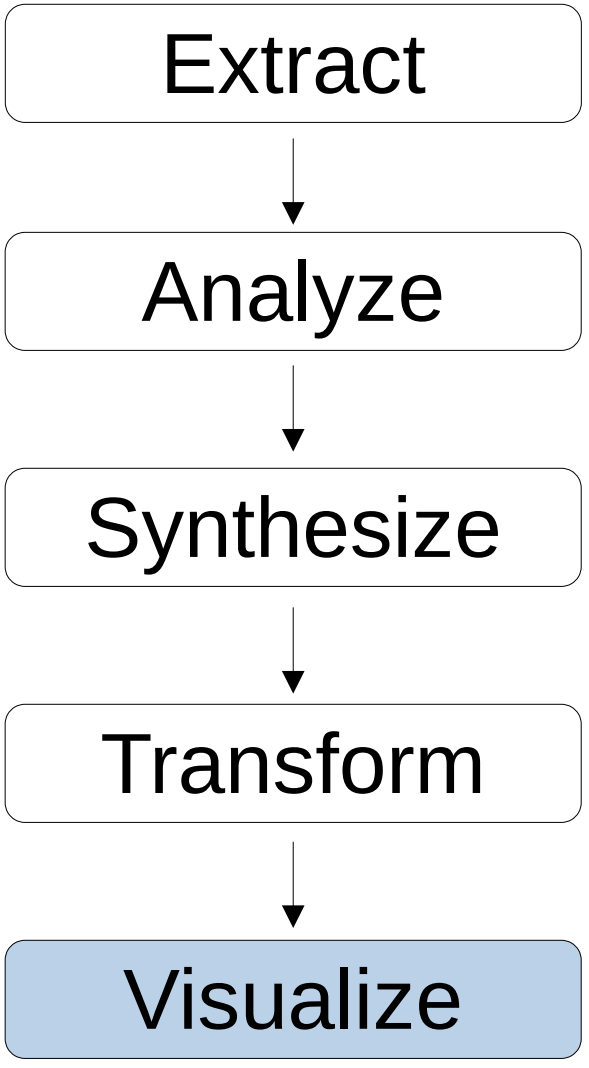
```
test bool testSqrt (int randNum) {
    int posNum = abs(randNum);
    if(posNum * posNum == pow(posNum, 2)){
        return true;
    }
    return false;
}
```



```
test bool testSqrt (int randNum) {
    int posNum = abs(randNum);
    return posNum * posNum == pow(posNum, 2);
}
```

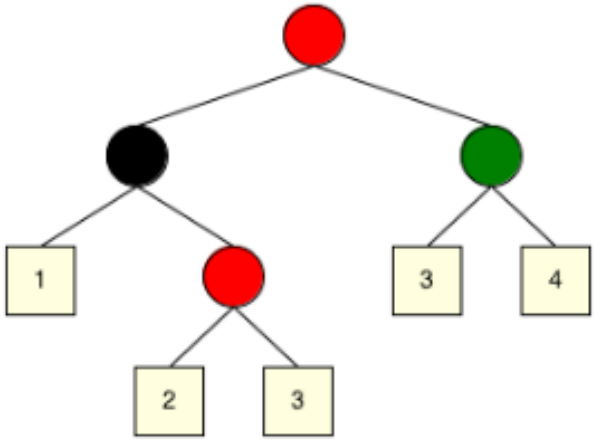
```
Declaration transformNaiveIfStatement(Declaration decl) = visit(decl) {
   case (Statement) `if (<Expression cond>) { return true; } else { return false; }` =>
        (Statement) `return <Expression cond>;`
   case (Statement) `if (<Expression cond>)  return true;  else return false;` =>
        (Statement) `return <Expression cond>;`
   case (Statement) `if (<Expression cond>) { return true; } return false;` =>
        (Statement) `return <Expression cond>;`
   case (Statement) `if (<Expression cond>) return true; return false;` =>
        (Statement) `return <Expression cond>;`
};
```

Inspired by: https://github.com/rafael2k/rascal-C99/


Extract
Analyze
Synthesize
Transform
Visualize

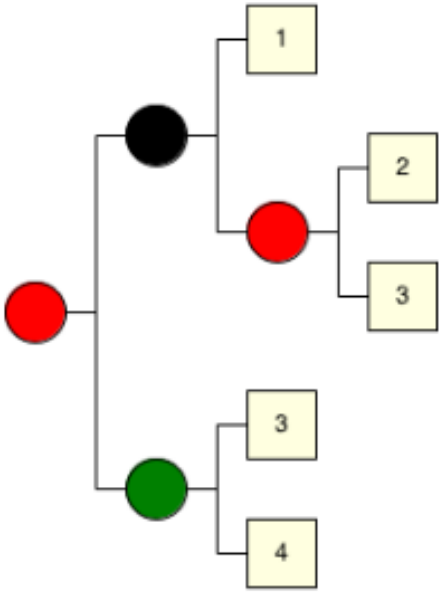
```
                import demo::vis::VisADT;
render(space(visColoredTree(rb), std(size(30)), std(gap(30)), std(manhattan(false))));
```

the result is:



```
                import demo::vis::VisADT;
render(space(visColoredTree(rb), std(size(30)), std(gap(30)), std(orientation(leftRight()))));
```

the result is:



```
import vis::Figure;
import vis::Render;
b1 = box(fillColor("red"), hshrink(0.5));
b2 = box(b1, fillColor("yellow"), size(200,100));
render(b2);
```

Rascal Quickcheck

```
test bool testSqrt (int randNum) {
    int posNum = abs(randNum);
    if(posNum * posNum == pow(posNum, 2)){
        return true;
    }
    return false;
}
```

Some other tips and tricks

- No syntax highlighting? Syntax error!
- Errors show in wrong places? Clean your project! (Project → Clean...)

Looks great...

What now?

Getting started...

- Installation: https://www.rascal-mpl.org/start/
- Documentation: http://docs.rascal-mpl.org/unstable/TutorHome/
- Reference (old): http://tutor.rascal-mpl.org/Rascal/Rascal.html
- Source code: https://github.com/usethesource/rascal/tree/master/src/org/rascalmpl/library

Some examples

- Official demo examples: https://github.com/usethesource/rascal/tree/master/src/org/rascalmpl/library/demo
- Several example applications explained: http://docs.rascal-mpl.org/unstable/Recipes/
- Grammar definition and AST rewriting: https://github.com/rafael2k/rascal-C99





Good luck!

(hopefully you won’t get stung too often)