Rascal Tutorial

Tijs van der Storm
storm@cwi.nl / @tvdstorm

Centrum Wiskunde & Informatica

University of Amsterdam

About me

- Researcher at *Centrum Wiskunde & Informatica (CWI)*, Amsterdam, NL
- Co-designer of Rascal
- Teacher at Universiteit van Amsterdam (UvA)
- Master Software Engineering
- Interests: DSLs, MDE, Meta-programming, PL

Today

- 09:00-10:00: intro + warming up
- 10:15-11:15: syntax + transformation
- 11:30-12:30: extraction + analysis
- lunch
- 14:00-15:00: code generation + closing

More information

- Handout: describes exercises in detail
  - http://www.cwi.nl/~storm/rascal-tutorial
- Cheat sheet: quick ref for Rascal language
- http://tutor.rascal-mpl.org
  - (also in Eclipse, under Rascal menu)
- http://ask.rascal-mpl.org

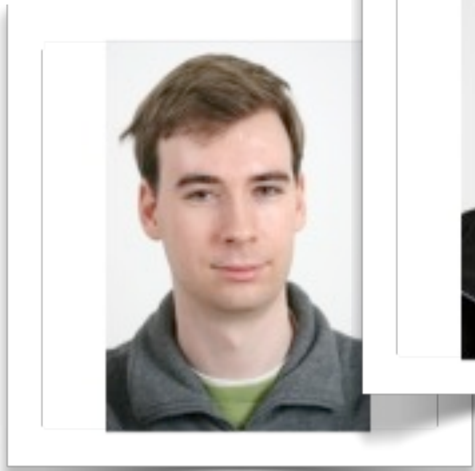
Project template

- Download the project:
- http://www.cwi.nl/~storm/rascal-tutorial/miss-grant.zip
- Import in Eclipse
- Have a look at the Rascal modules
- (More info in the hand out)

Rascal

Rascal Team

Paul Klint | Jurgen Vinju | Tijs v/d Storm | Bob Fuhrer | Atze v/d Ploeg



Arnold Lankamp | Bert Lisser | Bas Basten | Emilie Balland | Mark Hills | Jeroen van den Bos

Rascal is a DSL

- Domain: Meta Programming
- Rascal programs...
  - analyze,
  - transform,
  - visualize,
  - or generate ...other programs
- and nothing less, and nothing more

Rascal is a DSL

- Domain: Meta Programming
- Rascal programs...
  - analyze,
  - transform,
  - visualize,
  - or generate ...other programs
- and nothing less, and nothing more

Meta Software

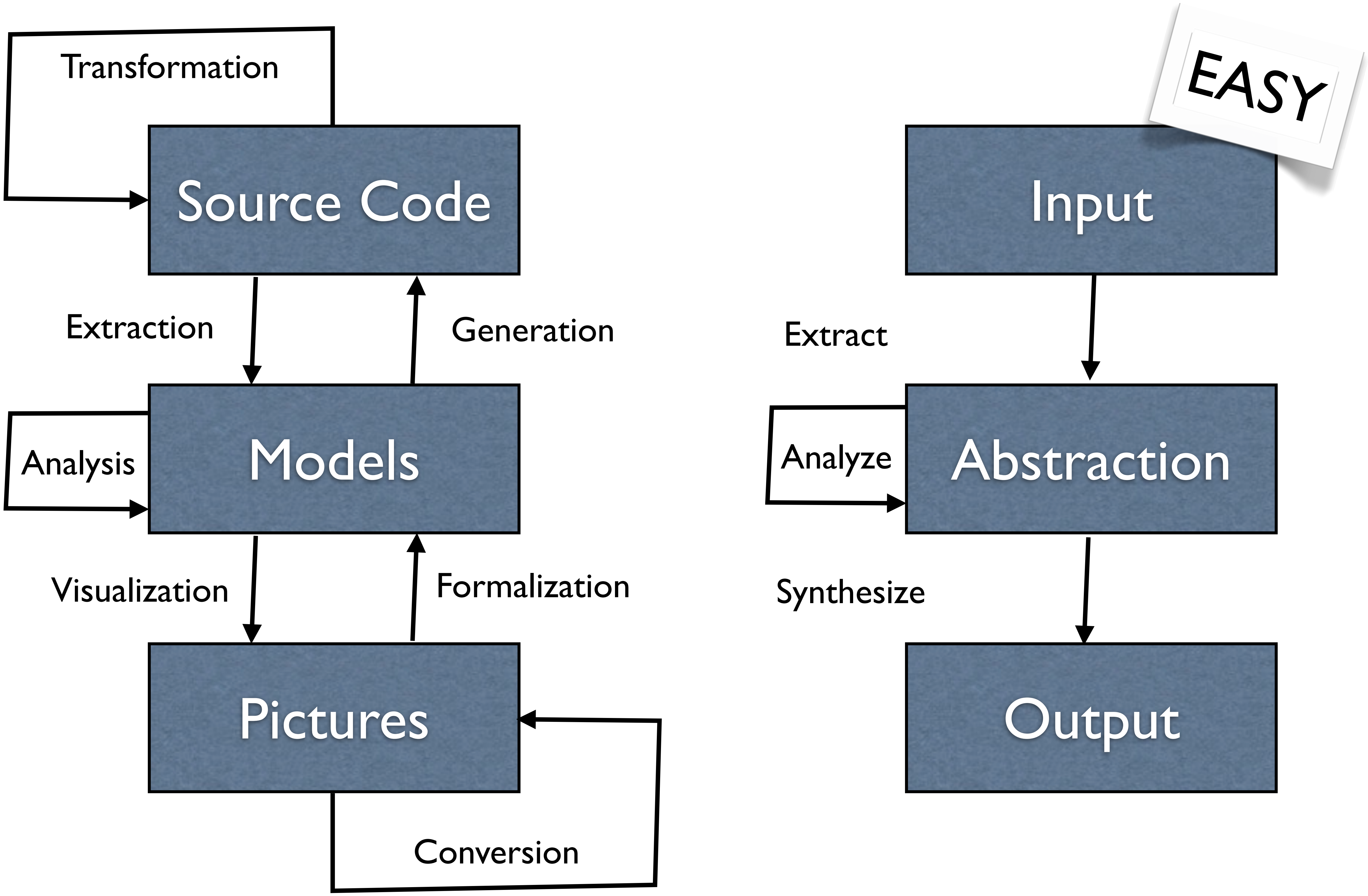
Static)Analysis

Program Transformation

- Dead code detection
- Slicing/Dependence
- Metrics
- Reverse engineering
- Verification
- Architecture recovery
- Code-to-model
- ...

- Goto elimination
- Dialect transformation
- Aspect weaving
- DSL compilers
- API migration
- Model-to-code
- Model-to-model
- ...

Metaprogramming



Use

- File extension: **.rsc**
- Open “Rascal Perspective”
  - Use “New Rascal Project” wizard
  - Use “New Rascal File” wizard
- Context-menu on Rascal editors
  - Start **Console**

Read-Eval-Print

```
rascal>1 + 1
int: 2
rascal>[1,2,3]
list[int]: [1,2,3]
rascal>{1,1,1}
set[int]: {1}
rascal>{ <i,i*i> | i <- [1..10]}
rel[int,int]: {<1,1>,<2,4>,<3,9>,...
```

Read-Eval-Print

```
rascal>import IO;
ok
rascal>for (i <- [1..10]) {
>>>>>>>  println("<i> * <i> = <i * i>");
>>>>>>>}
1 * 1 = 1
2 * 2 = 4
3 * 3 = 9
4 * 4 = 16
5 * 5 = 25
6 * 6 = 36
7 * 7 = 49
8 * 8 = 64
9 * 9 = 81
10 * 10 = 100
list[void]: []
rascal>
```

Modules

```
module path::to::Examples
import IO;

public int fac(int n) {
  if (n == 0) {
    return 1;
  }
  return n * fac(n - 1);
}
```

From coding to declaring

```
list[int] even(int max) {
  list[int] result = [];

  for (int i <- [0..max]) {
    if (i % 2 == 0) {
      result += i;
    }
  }
  return result;
}
```

From coding to declaring

```
list[int] even(int max) {
  list[int] result = [];

  for (int i <- [0..max], i%2 == 0) {
      result += i;
  }
  return result;
}
```

From coding to declaring

```
list[int] even(int max) {
  result = [];

  for (i <- [0..max], i%2 == 0) {
      result += i;
  }
  return result;
}
```

From coding to declaring

```
list[int] even(int max) {
  return for (i <- [0..max], i%2 == 0)
              append i;
}
```

From coding to declaring

```
list[int] even(int max) {
  return [i | i <- [0..max], i%2 == 0];
}
```

From coding to declaring

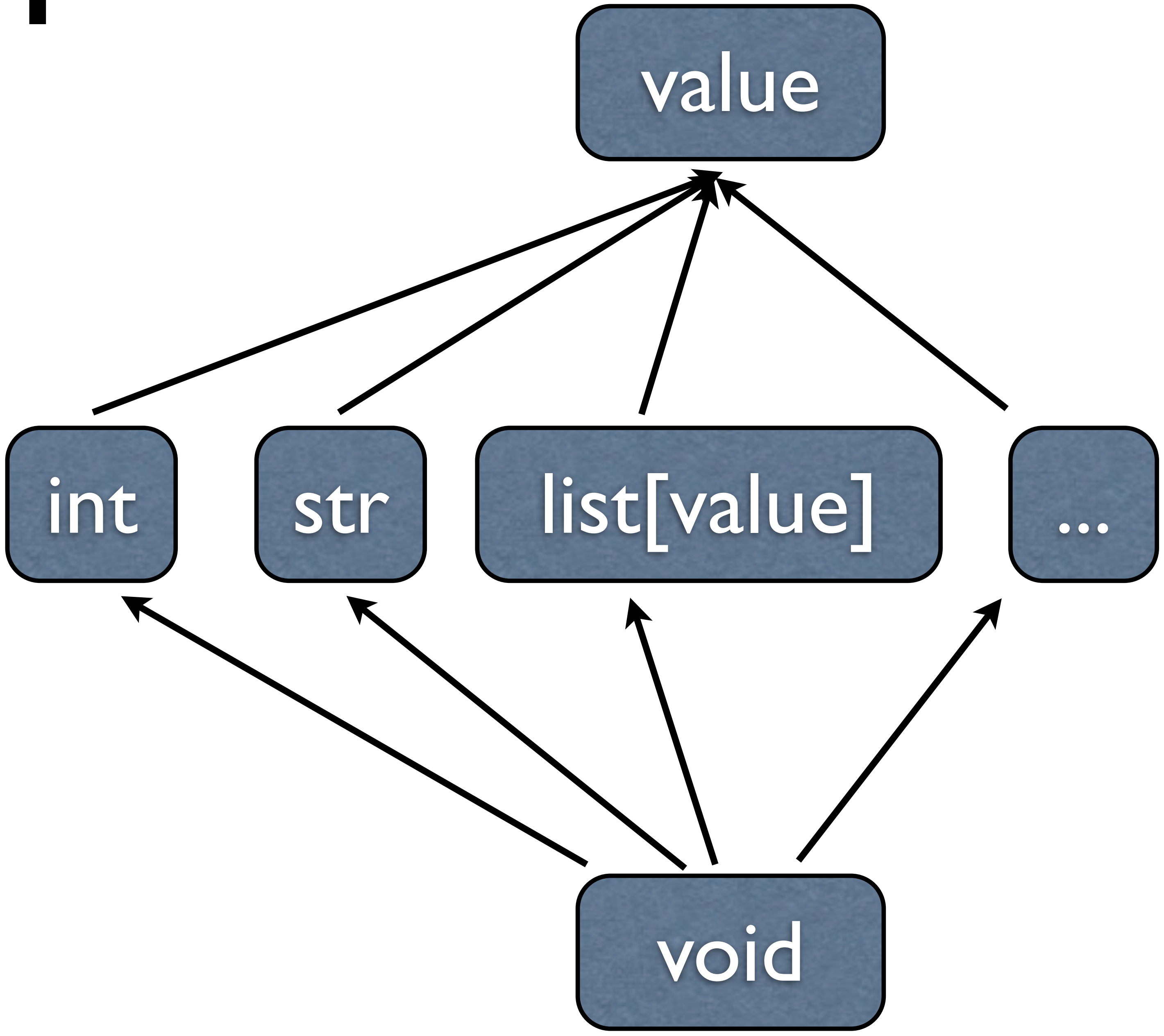
```
list[int] even(int max)
  = [i | i <- [0..max], i%2 == 0];
```

Immutable values

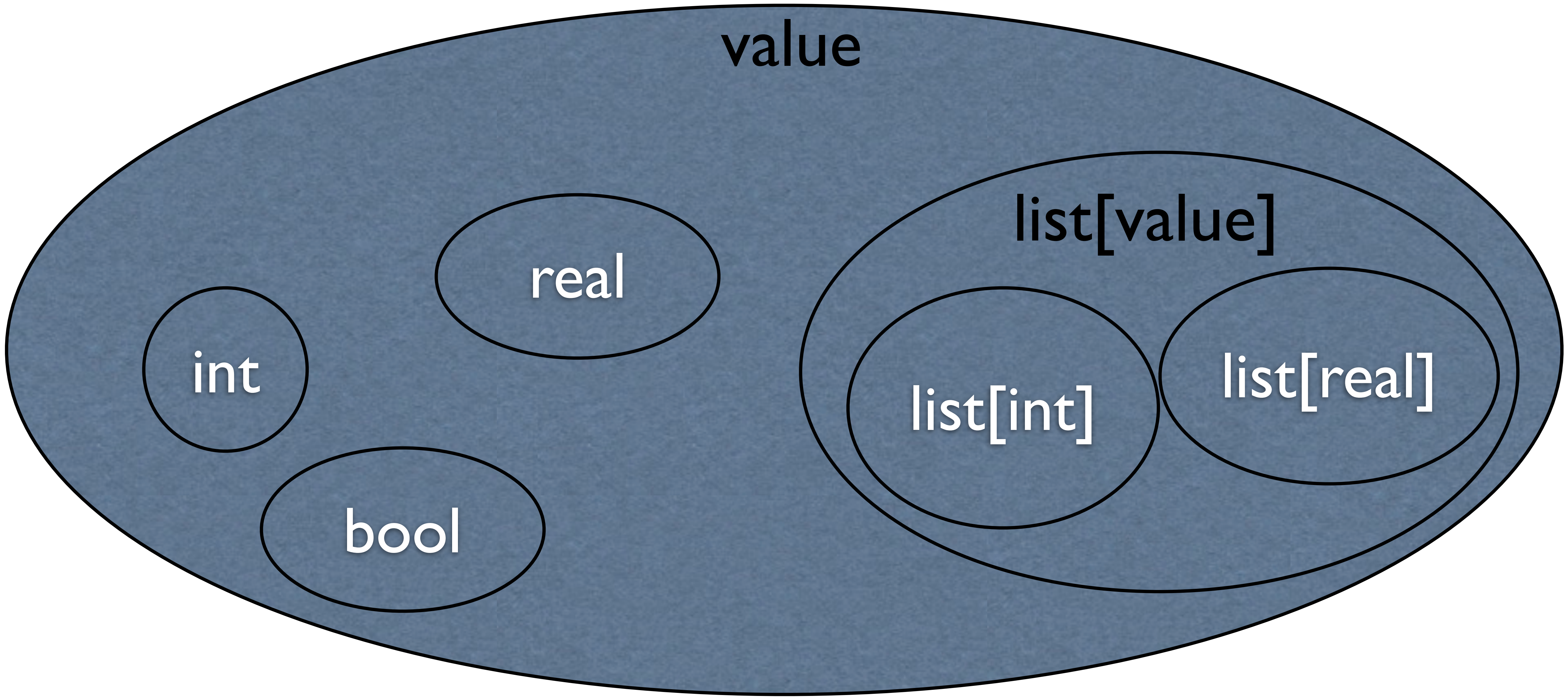
- **WYSIWYG values**
  - true, false
  - 1, 2, 3, ...
  - 1.0, 1.1, 1.11111111
  - [1,2,3]
  - {1,2,3}
  - (“1”:1,“2”:2)
  - name(“Y.T.”)
  - <1,2>, <1,2,1.0>
  - {<1,2>,2,1>}
  - *Nest any way you like*

Types

- list[void]: []
- list[int]: [1]
- list[value]: [1,“1”]
- set[int]: {1}
- set[value]: {1,”1”}



Sub-types



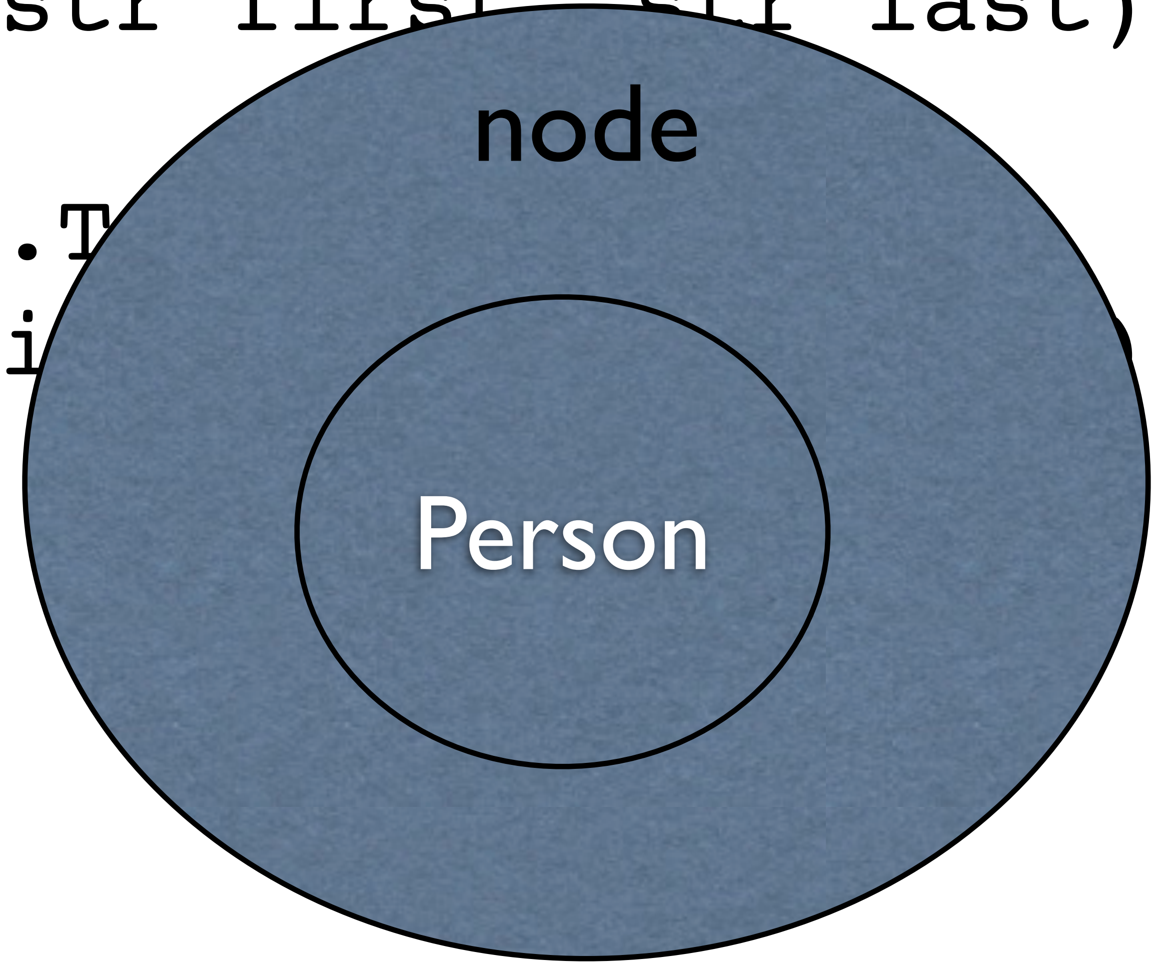
"A sub-type is a sub-set"

Trees and Data

```
node myNode = “person”(“Y.T”, 18);

data Person = person(str name, int age)
            | person(str first, str last);

Person YT = person(“Y.T”, 18);
Person MC = person(“Hiro”, “Protagonist”);
```

Trees and Data

```
node myNode = “person”(“Y.T”, 18);

data Person = person(str name, int age)
            | person(str first, str last);

Person YT = person(“Y.T
Person MC = person(“Hi                     );
```



Notable features

- Switch-case
- Visit
- Pattern matching
- Solve
- Comprehensions
- Grammars
- String templates
- Eclipse JDT interface
- IDE generation
- Concrete syntax
- ....

Warm-up (1)

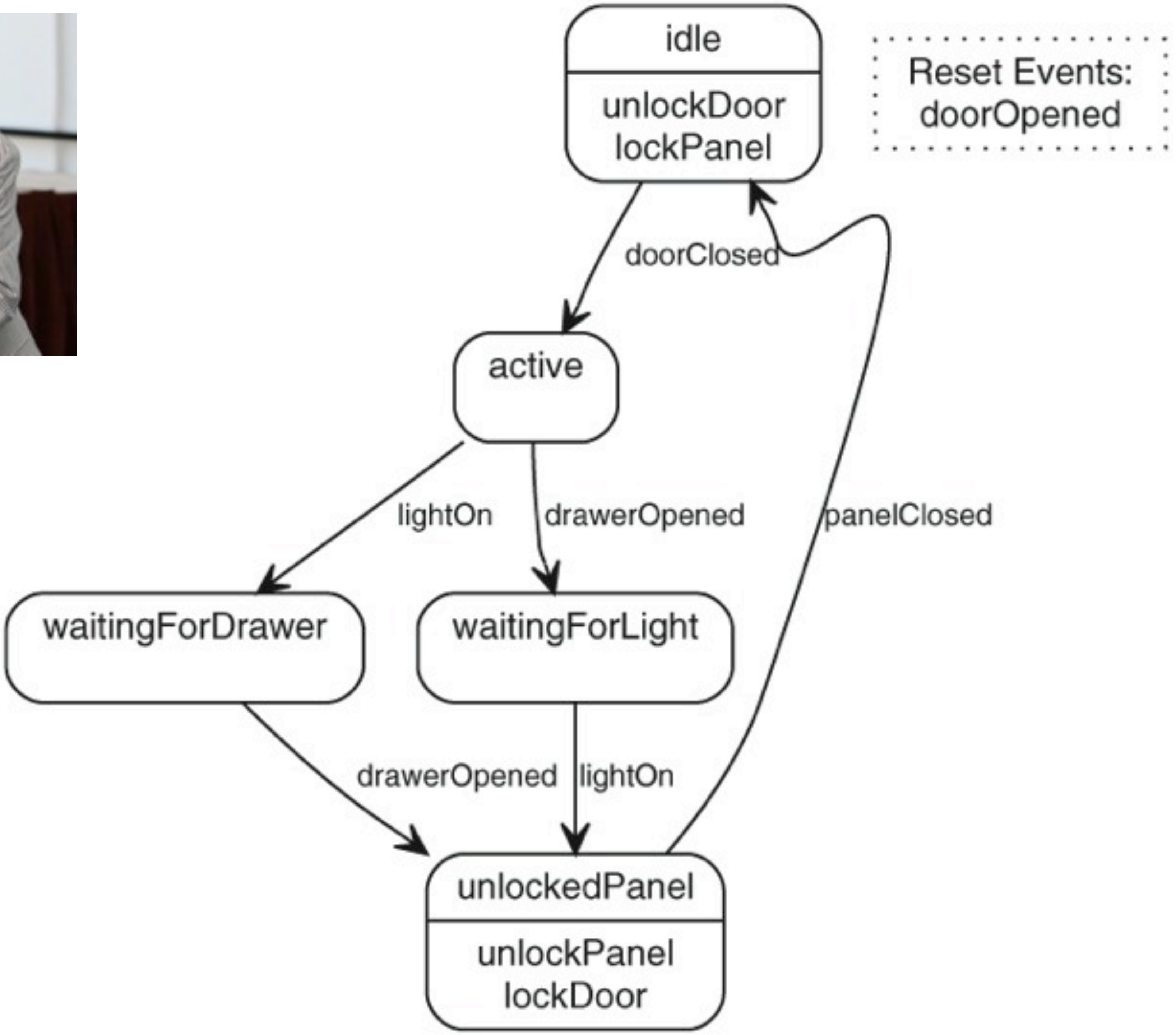
- `1 + 1, 4 / 2, 2 * 2`
- `int a = 1;`
- `b = 2;`
- `import Set;`
- `max({1,2,3})`
- `{ i | i <- [0..100]}`
- `int a := b`
- `<a,b> = <1,2>;`
- `if (1 > 2) println("x");`
- `int fac(int n) { if (n == 0 ) return 1; return n * f(n - 1); }`
- `[ p | p <- [1..100], all(i <- [2..10], i != p ==> p % i != 0)]`

Warm-up (2)

- Create new module Exp
- Add ADT for expressions: add, mul, const
- Write interpreter using switch

Domain-specific languages in Rascal

State machines



Textual notation

```
events
 doorClosed D1CL
 drawerOpened D2OP
 lightOn L1ON
 doorOpened D1OP
 panelClosed PNCL
end

resetEvents
 doorOpened
end

commands
 unlockPanel PNUL
 lockPanel PNLK
 lockDoor D1LK
 unlockDoor D1UL
end

state idle
 actions {unlockDoor lockPanel}
 doorClosed => active
end
```

```
state active
 drawerOpened => waitingForLight
 lightOn => waitingForDrawer
end

state waitingForLight
 lightOn => unlockedPanel
end

state waitingForDrawer
 drawerOpened => unlockedPanel
end


state unlockedPanel
 actions {unlockPanel lockDoor}
 panelClosed => idle
end
```

Aspects of DSL implementation

- Syntax
- Transformation
- Analysis
- Visualization
- Code generation

Syntax

- Lexical syntax
- Context-free syntax
- Abstract syntax
- Parsing

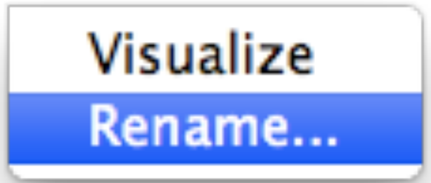
Parsing

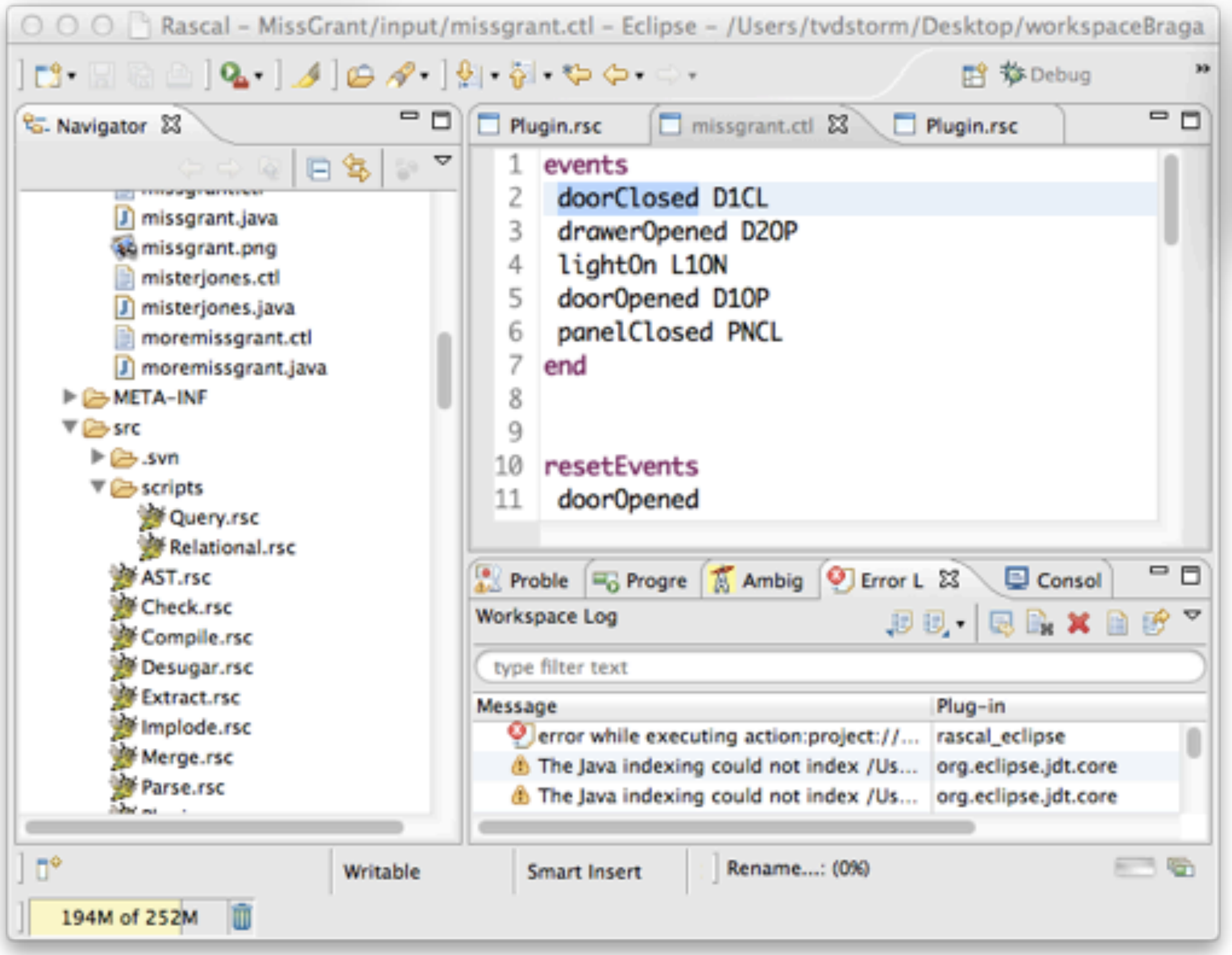
```
events
 doorClosed D1CL
 drawerOpened D2OP
 lightOn L1ON
 doorOpened D1OP
 panelClosed PNCL
end

resetEvents
 doorOpened
end

commands
 unlockPanel PNUL
 lockPanel PNLK
 lockDoor D1LK
 unlockDoor D1UL
end

state idle
 actions {unlockDoor lockPanel}
 doorClosed => active
end

state active
 drawerOpened => waitingForLight
 lightOn => waitingForDrawer
end

state waitingForLight
 lightOn => unlockedPanel
end

state waitingForDrawer
 drawerOpened => unlockedPanel
end


state unlockedPanel
 actions {unlockPanel lockDoor}
 panelClosed => idle
end
```

```
controller(
  [
    event("doorClosed","D1CL"),
    event("drawerOpened","D2OP"),
    event("lightOn","L1ON"),
    event("doorOpened","D1OP"),
    event("panelClosed","PNCL")
  ],
  ["doorOpened"],
  [
    command("unlockPanel","PNUL"),
    command("lockPanel","PNLK"),
    command("lockDoor","D1LK"),
    command("unlockDoor","D1UL")
  ],
  [
    state(
      "idle",
      ["unlockDoor","lockPanel"],
      [transition("doorClosed","active")]),
    state(
      "active",
      [],
      [
        transition("drawerOpened","waitingForLight"),
        transition("lightOn","waitingForDrawer")
      ]),
    state(
      "waitingForLight",
      [],
      [transition("lightOn","unlockedPanel")]),
    state(
      "waitingForDrawer",
      [],
      [transition("drawerOpened","unlockedPanel")]),
    state(
      "unlockedPanel",
      ["unlockPanel","lockDoor"],
      [transition("panelClosed","idle")])
  ])
```

Transformation

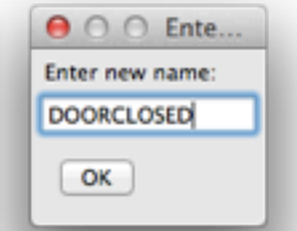
- Desugaring
- Optimization
- Normalization
- Refactoring
- ...

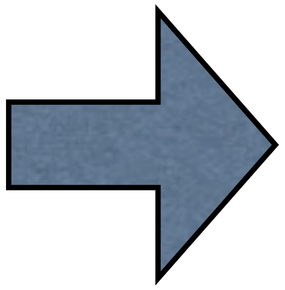




```
events
 doorClosed D1CL
 drawerOpened D2OP
 lightOn L1ON
 doorOpened D1OP
 panelClosed PNCL
end


resetEvents
 doorOpened
```

Message	Plug-in
error while executing action:project://...	rascal_eclipse
The Java indexing could not index /Us...	org.eclipse.jdt.core
The Java indexing could not index /Us...	org.eclipse.jdt.core

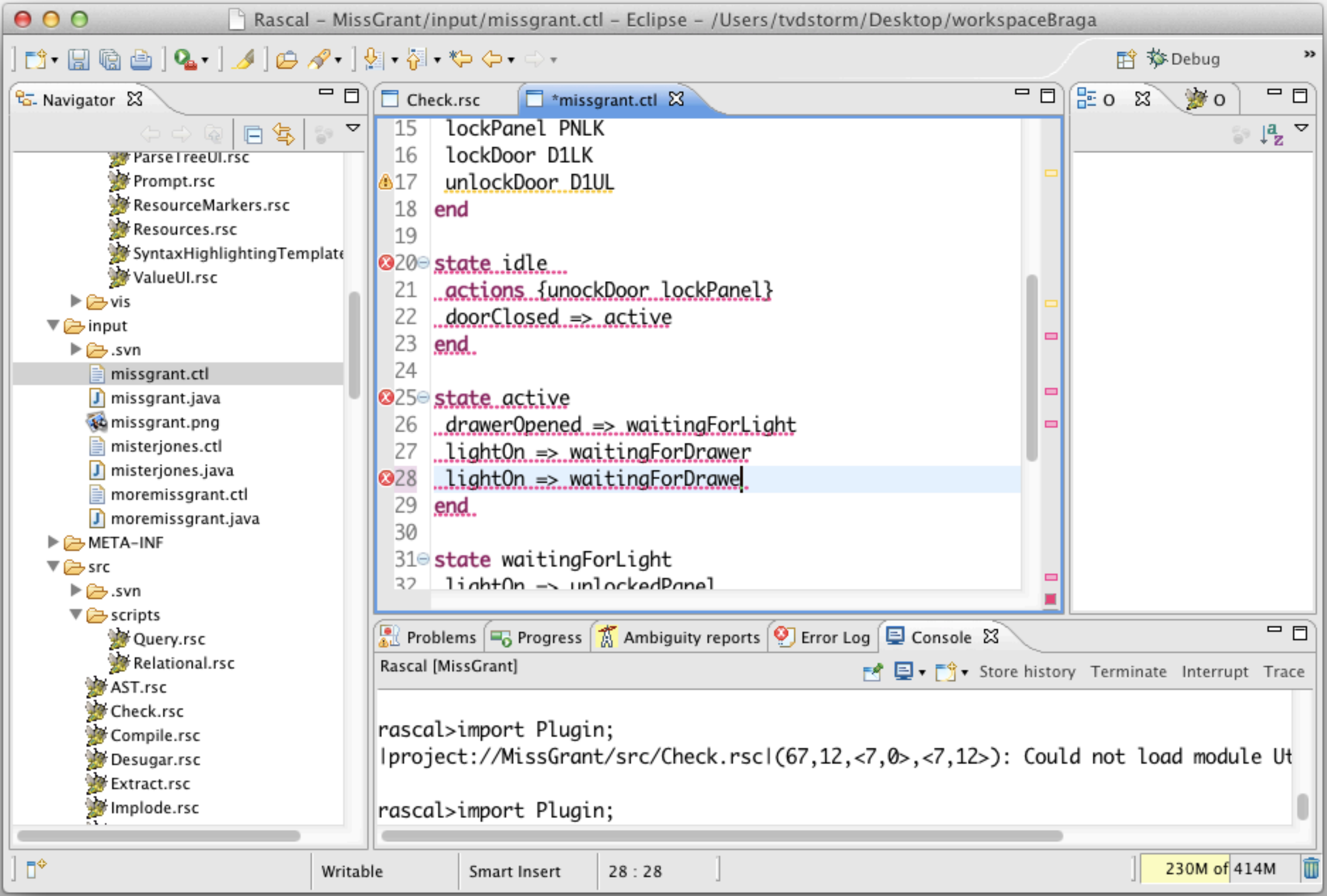




```
events
 DOORCLOSED D1CL
 drawerOpened D2OP
 lightOn L1ON
 doorOpened D1OP
 panelClosed PNCL
end


resetEvents
 doorOpened
end

commands
 unlockPanel PNUL
 lockPanel PNLK
 lockDoor D1LK
 unlockDoor D1UL
end

state idle
 actions {unlockDoor lockPanel}
 DOORCLOSED => active
end
```

Analysis

- Type checking
- Verification
- Model checking
- Metrics
- Smell detection
- ...

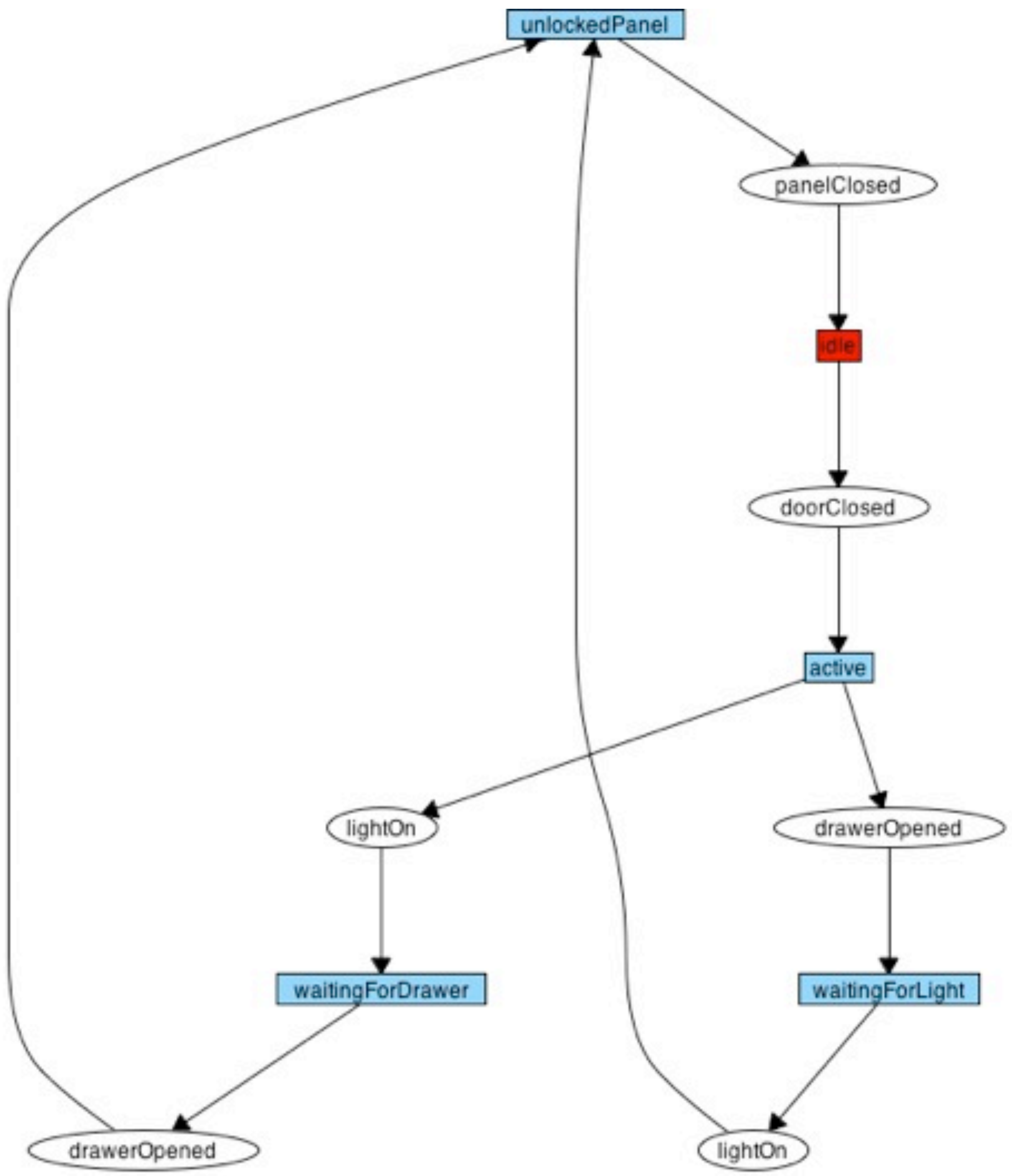

Rascal – MissGrant/input/missgrant.ctl – Eclipse – /Users/tvdstorm/Desktop/workspaceBraga

```
 lockPanel PNLK
 lockDoor D1LK
 unlockDoor D1UL
end

state idle
 actions {unockDoor lockPanel}
 doorClosed => active
end

state active
 drawerOpened => waitingForLight
 lightOn => waitingForDrawer
 lightOn => waitingForDrawe
end

state waitingForLight
 lightOn => unlockedPanel
```

```
rascal>import Plugin;
|project://MissGrant/src/Check.rsc|(67,12,<7,0>,<7,12>): Could not load module Ut

rascal>import Plugin;
```

Wednesday, May 23, 12

Visualization



Code generation

```
events
 doorClosed D1CL
 drawerOpened D2OP
 lightOn L1ON
 doorOpened D1OP
 panelClosed PNCL
end

resetEvents
 doorOpened
end

commands
 unlockPanel PNUL
 lockPanel PNLK
 lockDoor D1LK
 unlockDoor D1UL
end

state idle
 actions {unlockDoor lockPanel}
 doorClosed => active
end

state active
 drawerOpened => waitingForLight
 lightOn => waitingForDrawer
end

state waitingForLight
 lightOn => unlockedPanel
end

state waitingForDrawer
 drawerOpened => unlockedPanel
end


state unlockedPanel
 actions {unlockPanel lockDoor}
 panelClosed => idle
end
```

```java
public class missgrant {
    public static void main(String args[]) throws java.io.IOException {
        new missgrant().run(new java.util.Scanner(System.in),
                new java.io.PrintWriter(System.out));
    }

    private static final int state$idle = 0;
    private static final int state$active = 1;
    private static final int state$waitingForLight = 2;
    private static final int state$waitingForDrawer = 3;
    private static final int state$unlockedPanel = 4;

    public void run(java.util.Scanner input, java.io.Writer output)
            throws java.io.IOException {
        int state = state$idle;
        while (true) {
            String token = input.nextLine();
            switch (state) {

            case state$idle: {
                unlockDoor(output);
                lockPanel(output);
                if (doorClosed(token)) {
                    state = state$active;
                }
                if (doorOpened(token)) {
                    state = state$idle;
                }
                break;
            }

            case state$active: {
                if (drawerOpened(token)) {
                    state = state$waitingForLight;
                }
                if (lightOn(token)) {
                    state = state$waitingForDrawer;
                }
                if (doorOpened(token)) {
                    state = state$idle;
                }
                break;
            }

            case state$waitingForLight: {
```

Outlook

- Next three hours:
- 1: syntax + transformation
- 2: extraction + analysis
- 3: code generation