CWI

Creating a DSL using Rascal

Concrete syntax

```
extend lang::std::Layout;
extend lang::std::Id;
start syntax Machine = machine: State+;
syntax State = state: "state" Id Trans*;
syntax Trans = trans: Id "=\>" Id;
```

Abstract syntax

```
data Machine = machine(list[State] states);
data State = state(str name, list[Trans] out);
data Trans = trans(str event, str to);
```

Analysis

```
set[str] unreachable(Machine m) {
  r = { <q1,q2> | state(q1, ts) <- m.states,
                  trans(_, q2) <- ts }+;
  q0 = m.states[0].name;
  qs = { q | state(q, _) <- m.states };
  return { q | q <- qs, q notin r[q0] };
}
```

Code generation

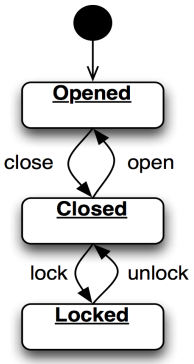
```
str compile(Machine m) =
  "while (true) {
  '  event = input.next();
  '  switch (current) {
  '    <for (q <- m.states) {>
  '    case \"<q.name>\":
  '      <for (t <- q.out) {>
  '      if (event.equals(\"<t.event>\"))
  '        current = \"<t.to>\";
  '      <}>
  '      break;
  '    <}>
  '  }
  '}";
```

DSL Code

```
state Opened
  close => Closed

state Closed
  open => Opened
  lock => Locked

state Locked
  unlock => Closed
```




```
compile(implode(parse(src)))
```

Java Code

```
while (true) {
    event = input.next();
    switch (current) {
    case "Opened":
        if (event.equals("close"))
            current = "Closed";
        break;
    case "Closed":
        if (event.equals("open"))
            current = "Opened";
        if (event.equals("lock"))
            current = "Locked";
        break;
    case "Locked":
        if (event.equals("unlock"))
            current = "Closed";
        break;
    }
}
```

www.rascal-mpl.org