Bloqqi: Feature-Based Automation Programming

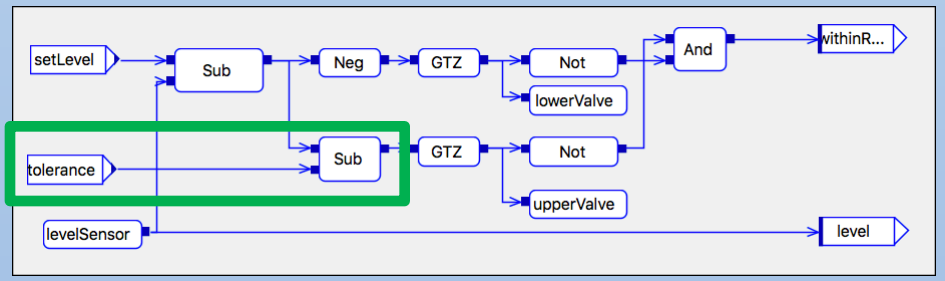
Niklas Fors, Lund University, @LangDev Meetup, 2018-03-08

Bloqqi: Feature-based data-flow programming

Tolerance feature

Bloqqi program for tank control

Visual view



Textual view

```
diagramtype  Tank(setLevel:  Int, tolerance:  Int
    => level:  Int, withinRange:  Bool) {
  upperValve:  Valve;
  lowerValve:  Valve;
  levelSensor:  Sensor;
  ...
  connect(setLevel,  Sub_1.in1);
  connect(levelSensor.out,  Sub_1.in2);
  connect(levelSensor.out,  level);
  ...
}
```

Runs in

Real world

1. Read liquid level

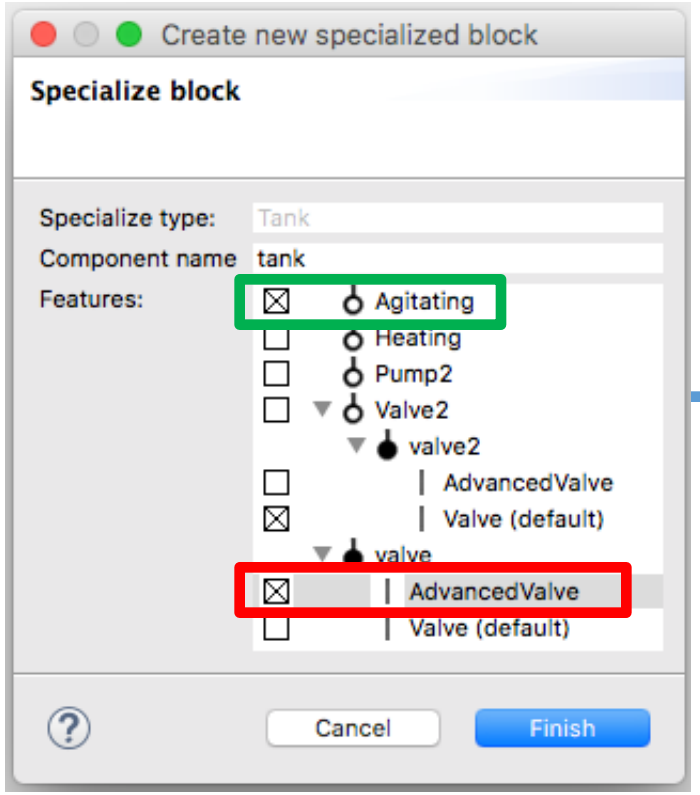
3. Open/ close valves

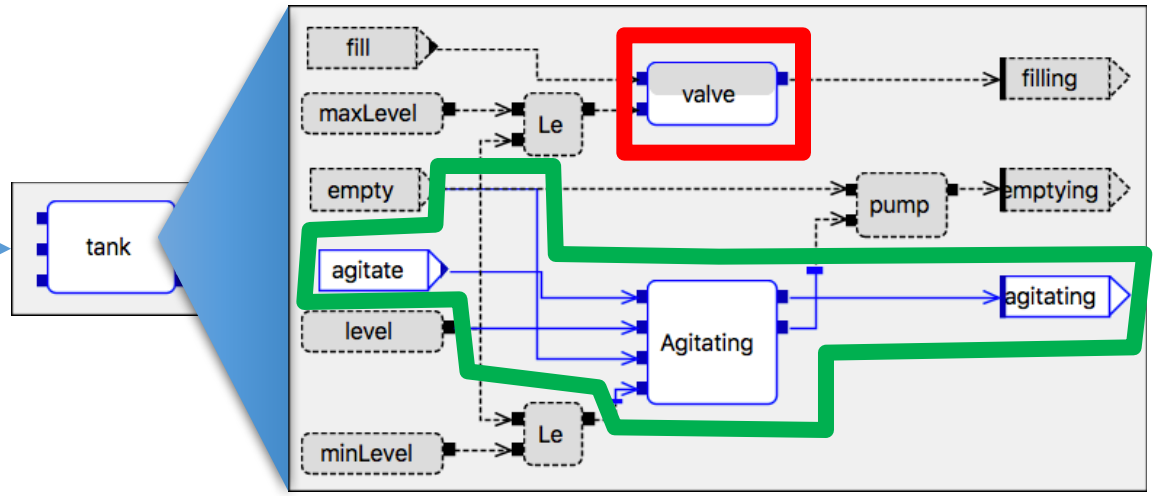
2. Compute control signal

Control system

Feature wizards

- Derived automatically from library
- Selected features automatically wired





- Implemented in JastAdd
- Compiler, editor, feature-wizard, ...

Mechanisms in Bloqqi

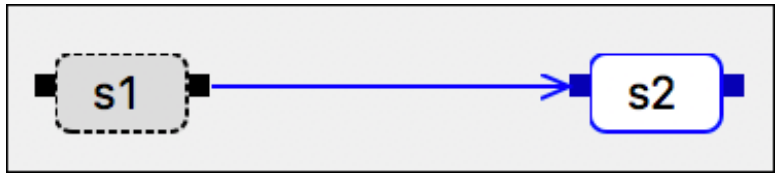
- Data-flow diagrams with ports, blocks, connections, variables, i/o
- Diagram inheritance (connection interception, block redeclaration)
- Recommendations – optional features for variability
- Feature interaction resolution
- Modular feature libraries
- Automatic feature wizards

Diagram inheritance

A

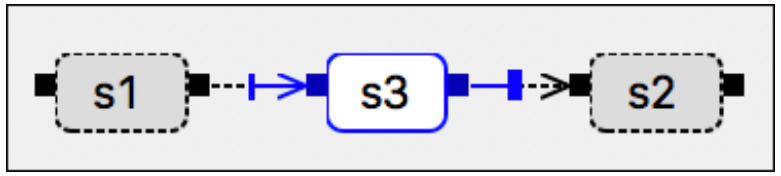

```
diagramtype A {
  s1: S;
}
```

B extends A




```
diagramtype B extends A {
  s2: S;
  connect(s1.out, s2.in);
}
```

C extends B

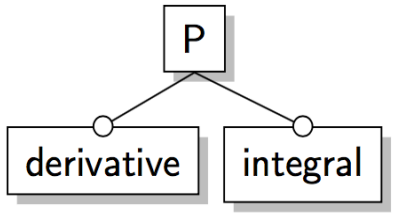


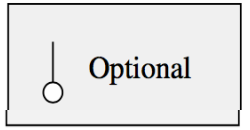

```
diagramtype C extends B {
  s3: S;
  intercept s2.in with s3.in,s3.out;
}
```

Connection interception

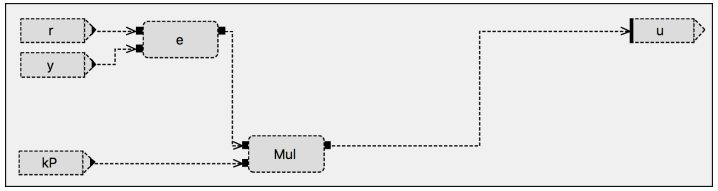
4 Control Variants

Feature model

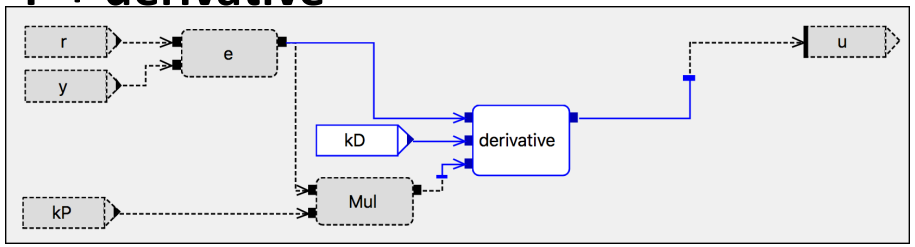




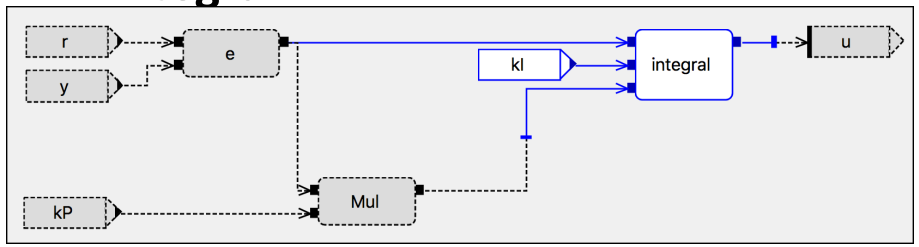
Proportional (base diagram)



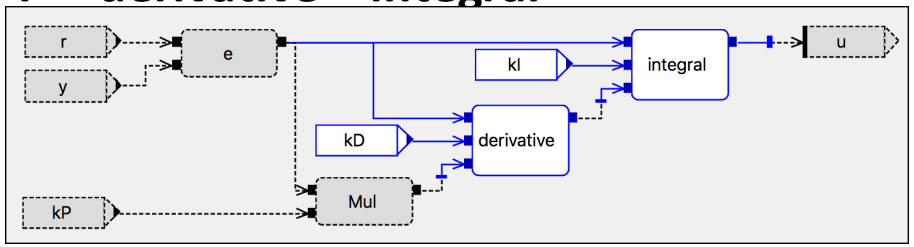
P + derivative



P + integral



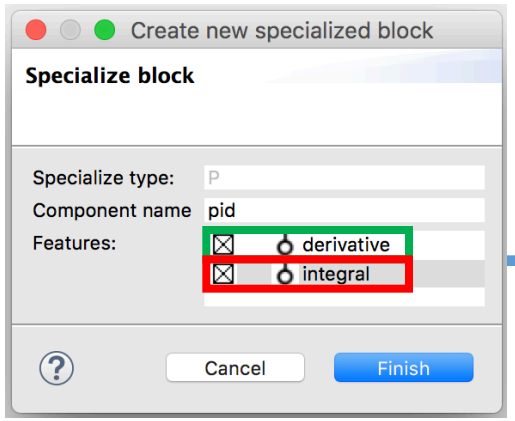
P + derivative + integral



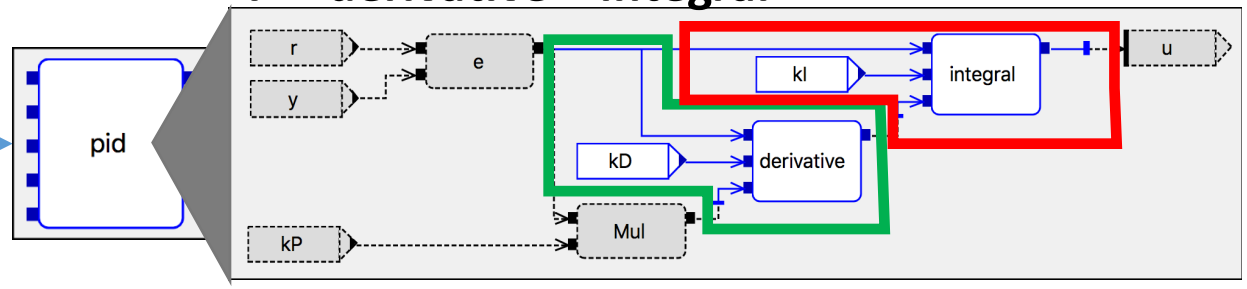
base type + feature selection = anonymous subtype (variant)

The wizard wires features automatically

Feature wizard for P



P + derivative + integral

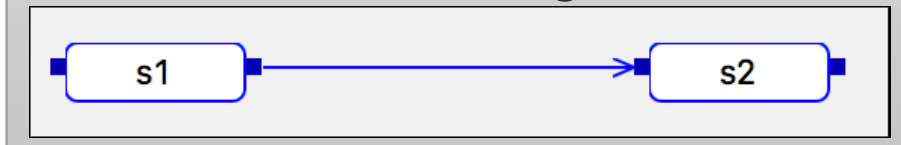


Automatic wiring of features

The wizard is computed from library code

Recommendations – simple example

Base diagram **A** with mandatory features




```
diagramtype A {
  s1: S;
  s2: S;
  connect(s1.out, s2.in);
}
```

Recommendations – optional features

```
recommendation A {
  f: F[s2.in];
}
```

Feature name

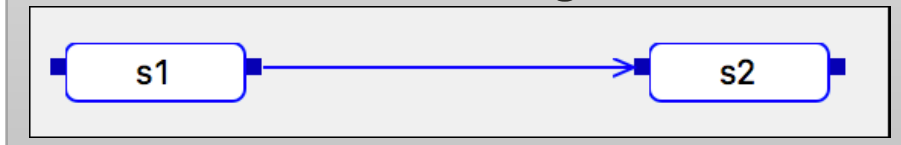
Feature type (other diagram)

How the wiring is done

A has an optional feature **f** that is added before **s2**

Recommendations – simple example

Base diagram **A** with mandatory features



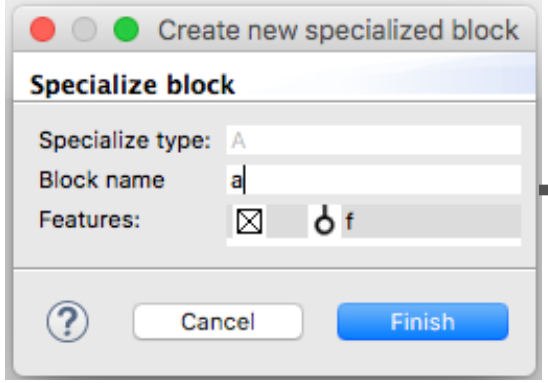

```
diagramtype A {
  s1: S;
  s2: S;
  connect(s1.out, s2.in);
}
```

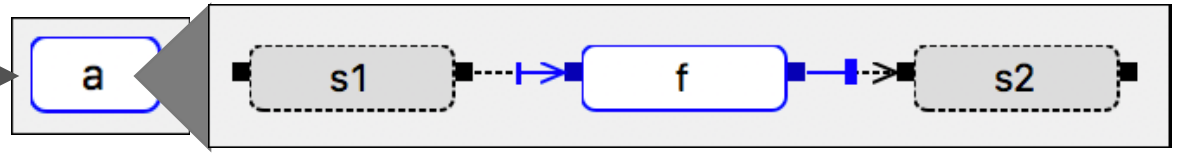
Recommendations – optional features

```
recommendation A {
  f: F[s2.in];
}
```

A has an optional feature **f** that is added before **s2**

Computed feature wizard for A





f is wired before s2

[Demo]

Modular Tool Implementation

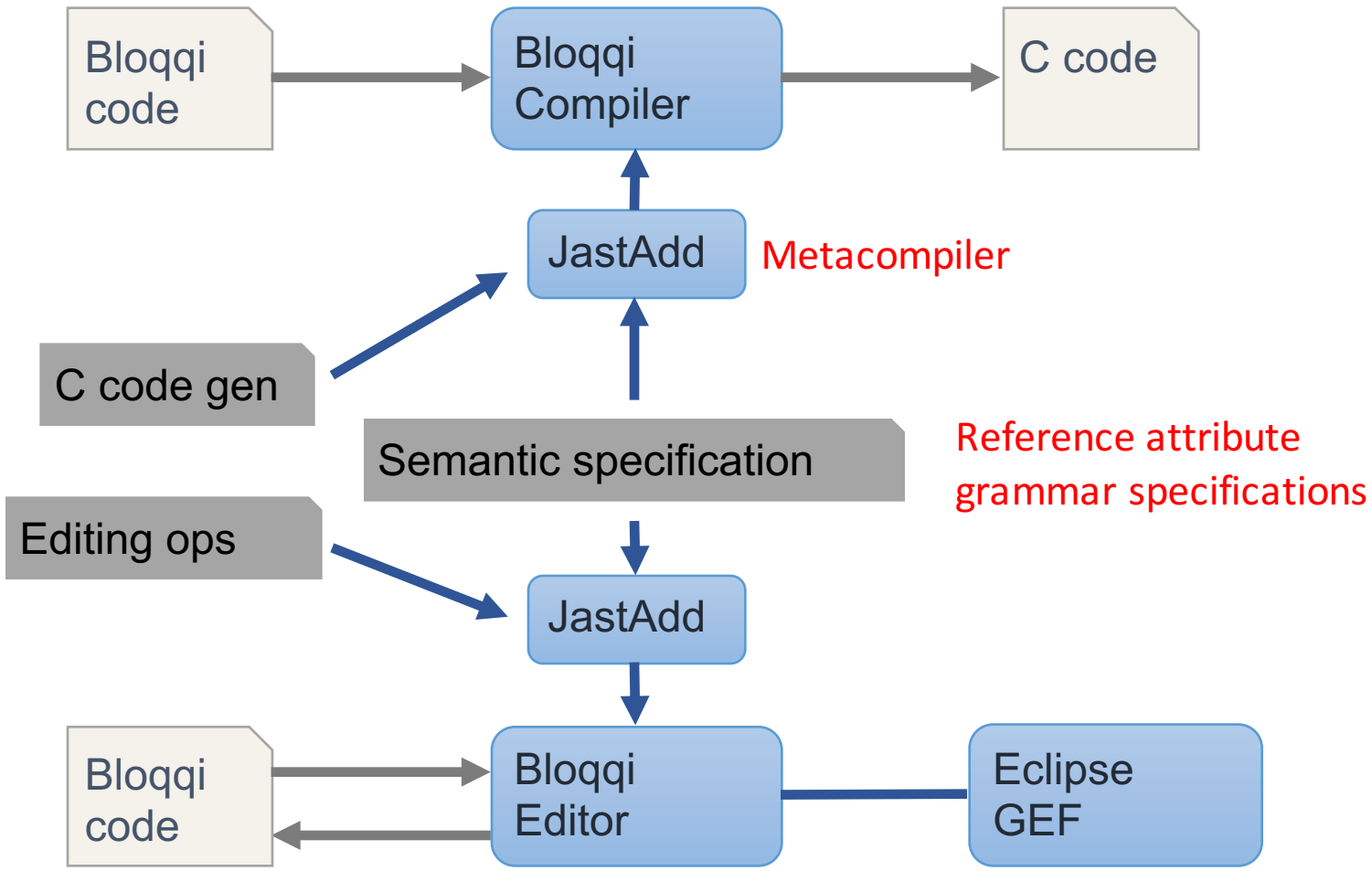
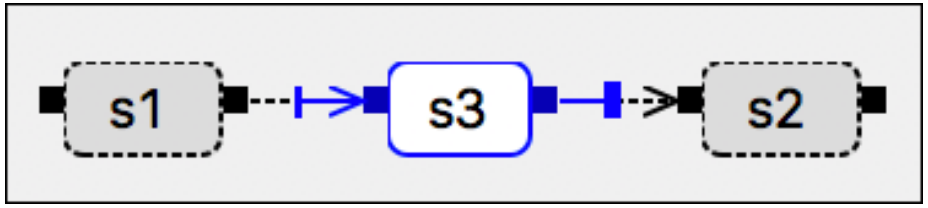


Diagram Views are Computed

C extends **B**




```
diagramtype C extends B {
  s3: S;
  intercept s2.in with s3.in,s3.out;
}
```

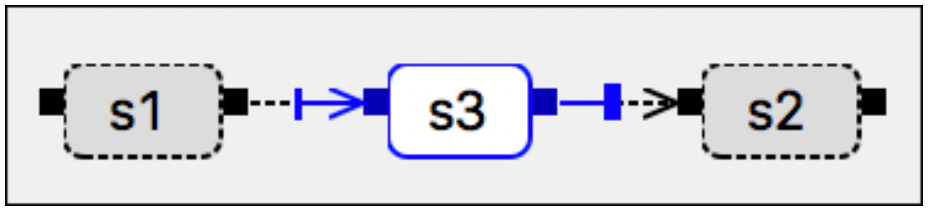
Diagram view computed based on semantic analysis

- Block **s1** from **A**
- Block **s2** from **B**
- Connection **s1->s2** from **B** is intercepted
- Ports from block type **S**

Implemented using **non-terminal attributes (NTAs)**
(computed subtrees)

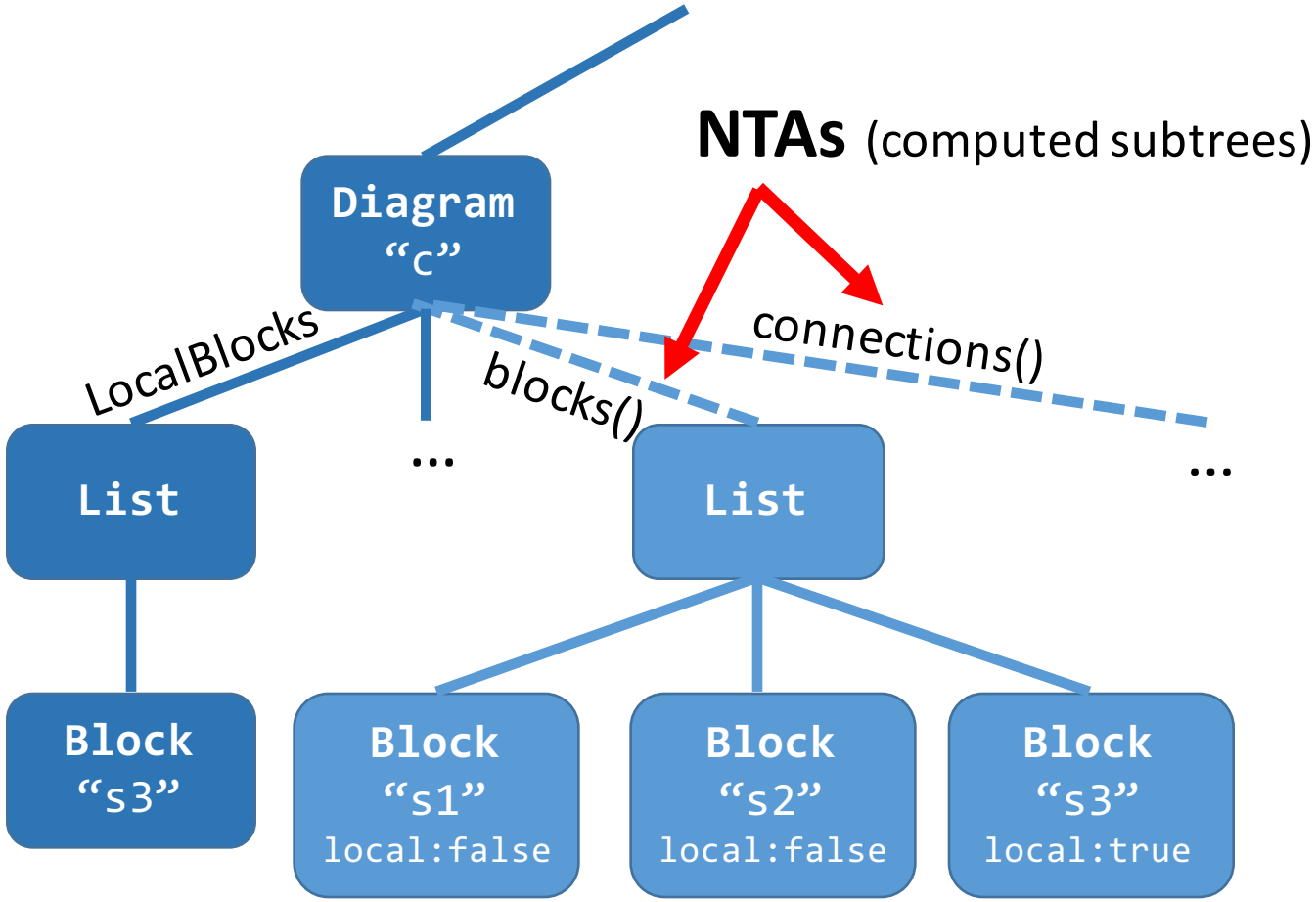
Computed Subtrees

C extends B



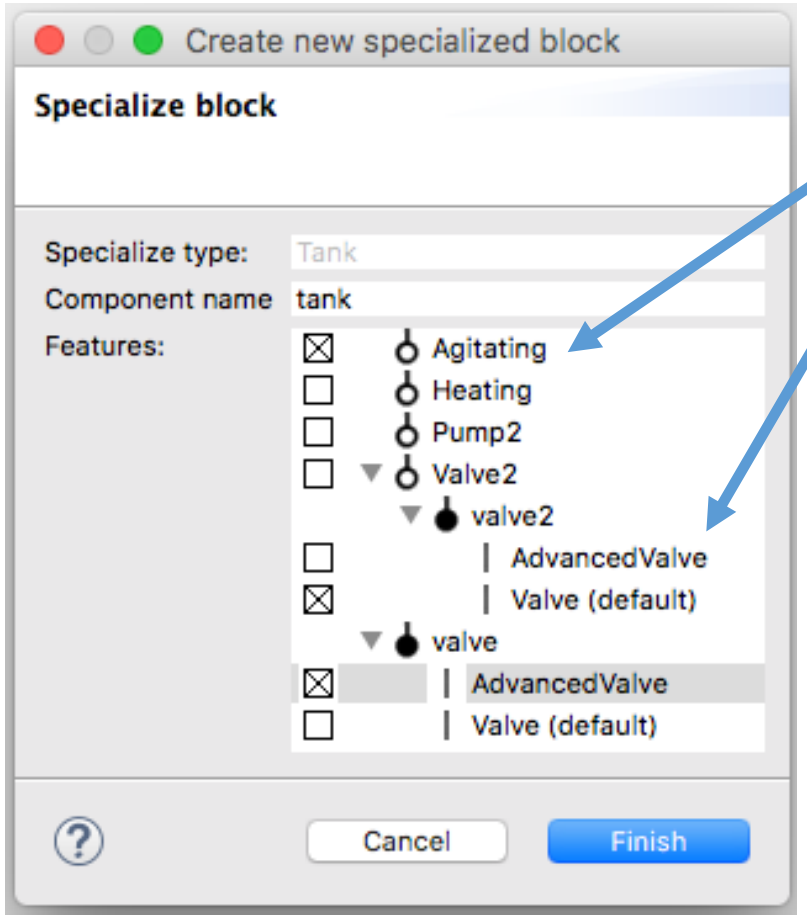

```
diagramtype C extends B {
  s3: S;
  intercept s2.in with s3.in,s3.out;
}
```

Attributed AST



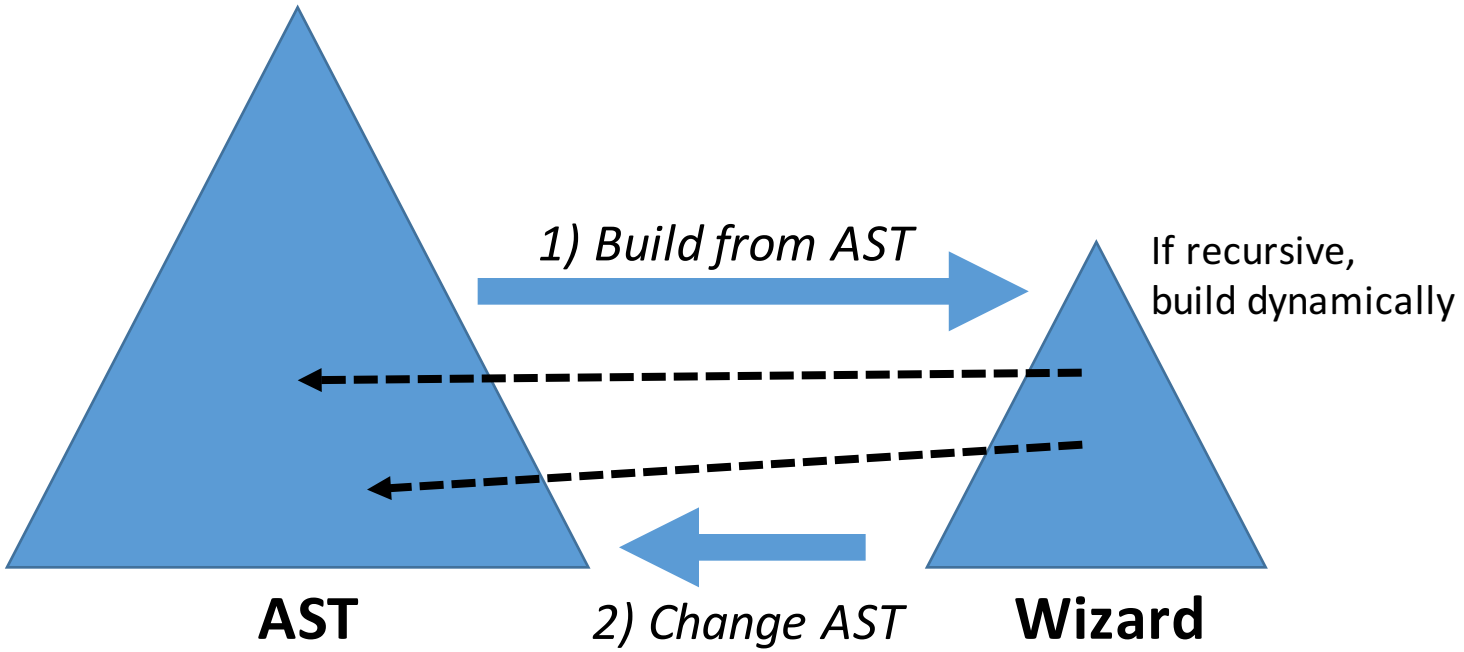
Attribute **local** is used to color the block

Computed Wizard



Wizard computed based on semantics:

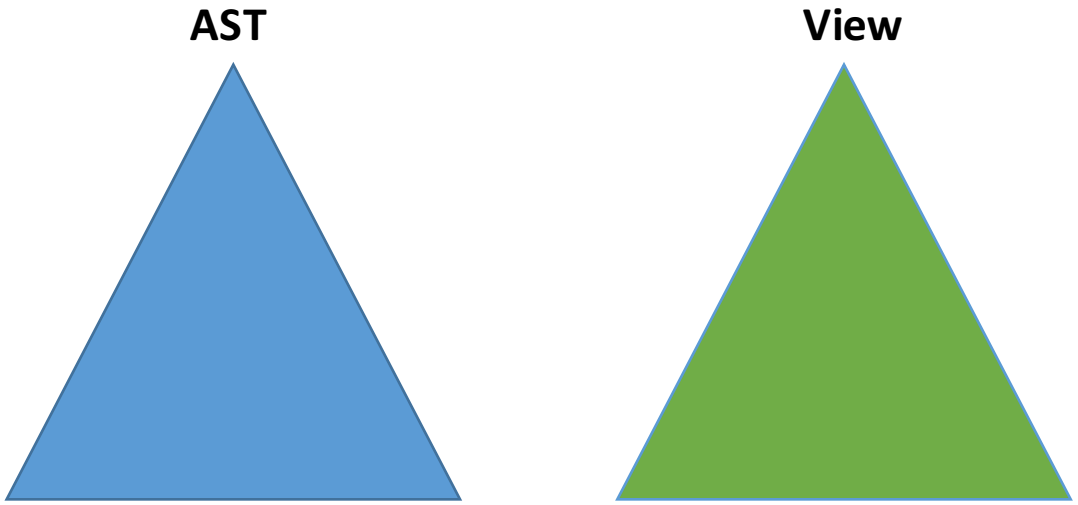
- **Optional features**: computed from recommendations
- **Alternatives**: computed from subtype relationships



Change Handling: Flush All

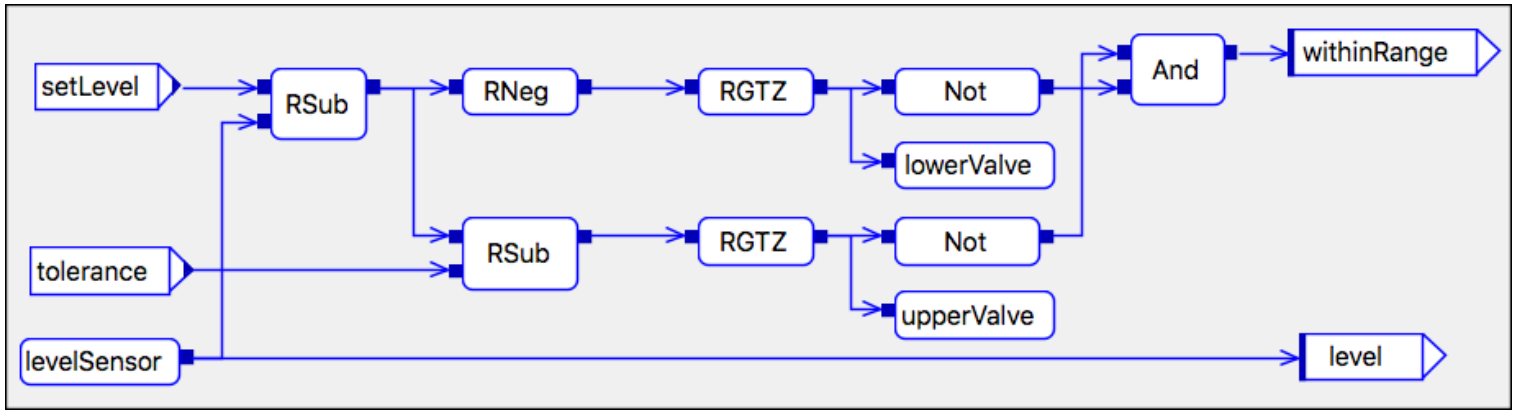
For each change in editor:

- Change AST
- Flush all attributes
- Notify view that AST has changed
- Update view using attributes
- Re-compute attributes on-demand



Simulation with Tank Model

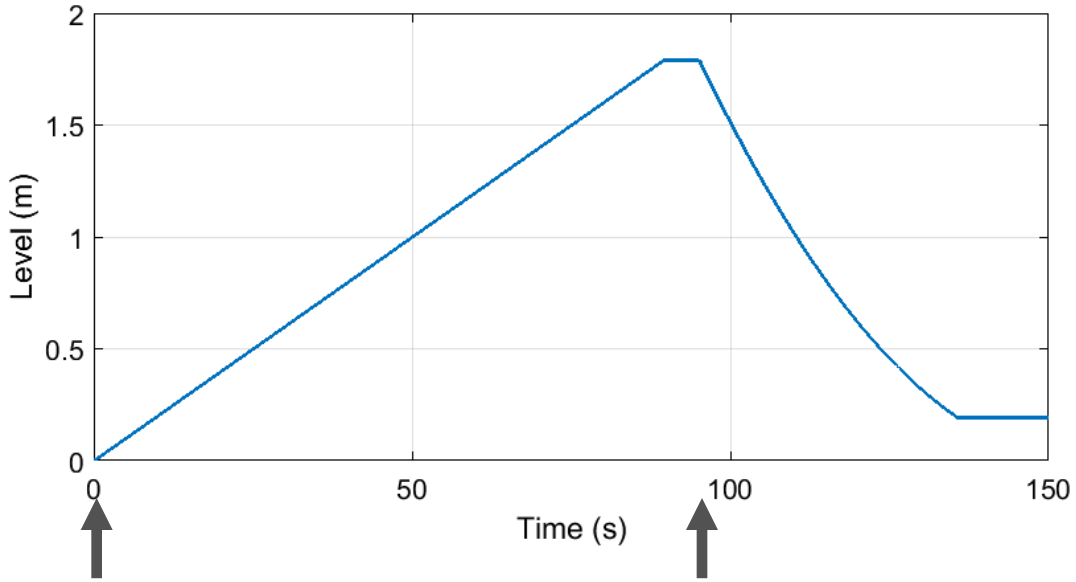
Tank regulator (as before)



Tank model (in Modelica)

```
der(level) = (inFlow-outFlow)/AREA;
inFlow = if upperValveOpen
  then IN_FLOW
  else 0.0;
outFlow = if lowerValveOpen
  then (OUT_VALVE_AREA)*sqrt(2*9.82*level))
  else 0.0;
```

Tank level simulation result



Setpoint changes: 1.8 0.2

Conclusions

Bloqqi

- Feature-based language for automation based on
  - Inheritance
  - Connection interception
  - Block redeclaration
  - Recommendations
- Implemented using JastAdd
  - Diagram view computed using attributes
  - Attributes flush after each update