LIDL

LIDL Interaction Description Language

Vincent Lecrubier

ONERA, Toulouse

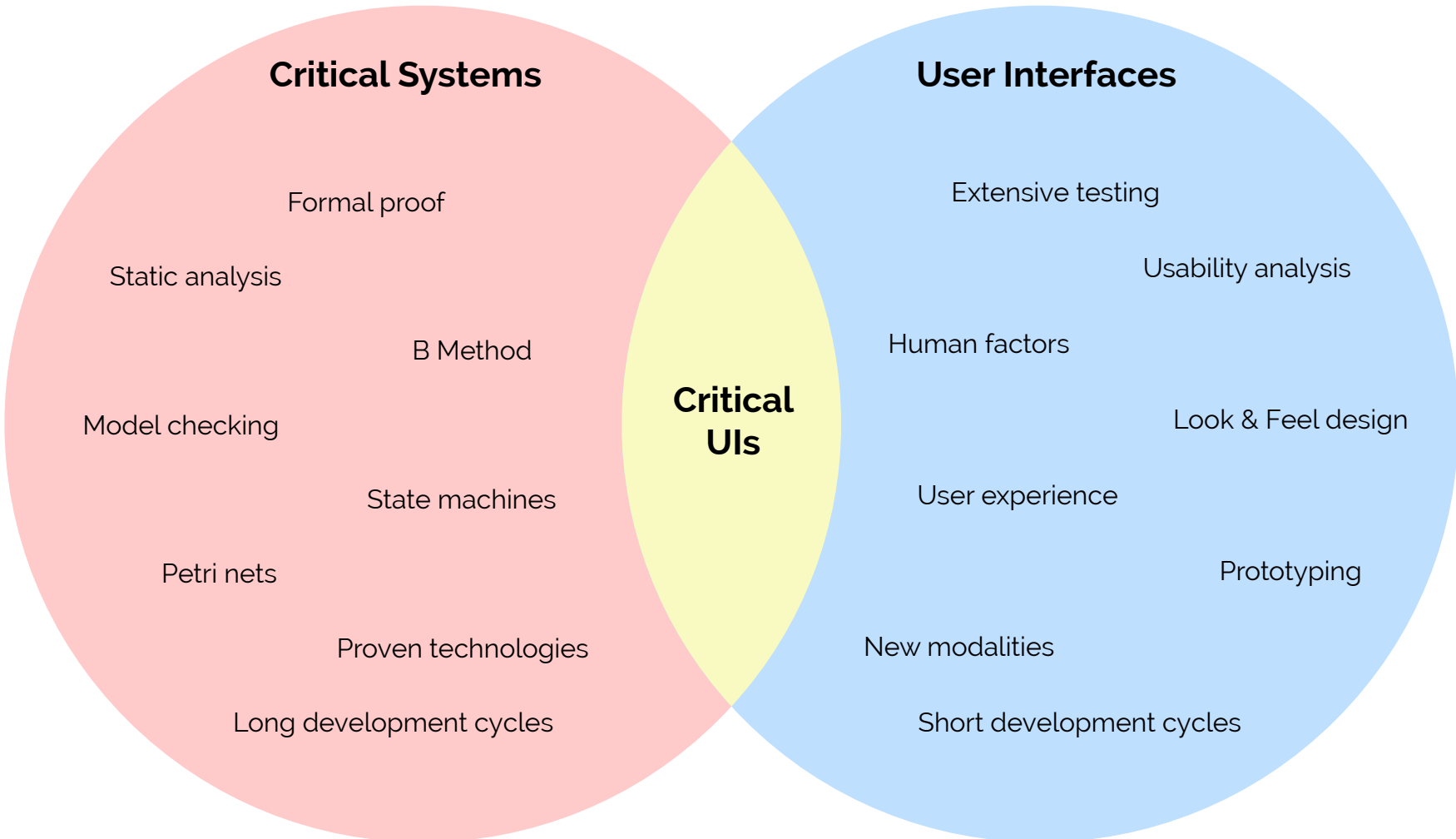
Contents

Why

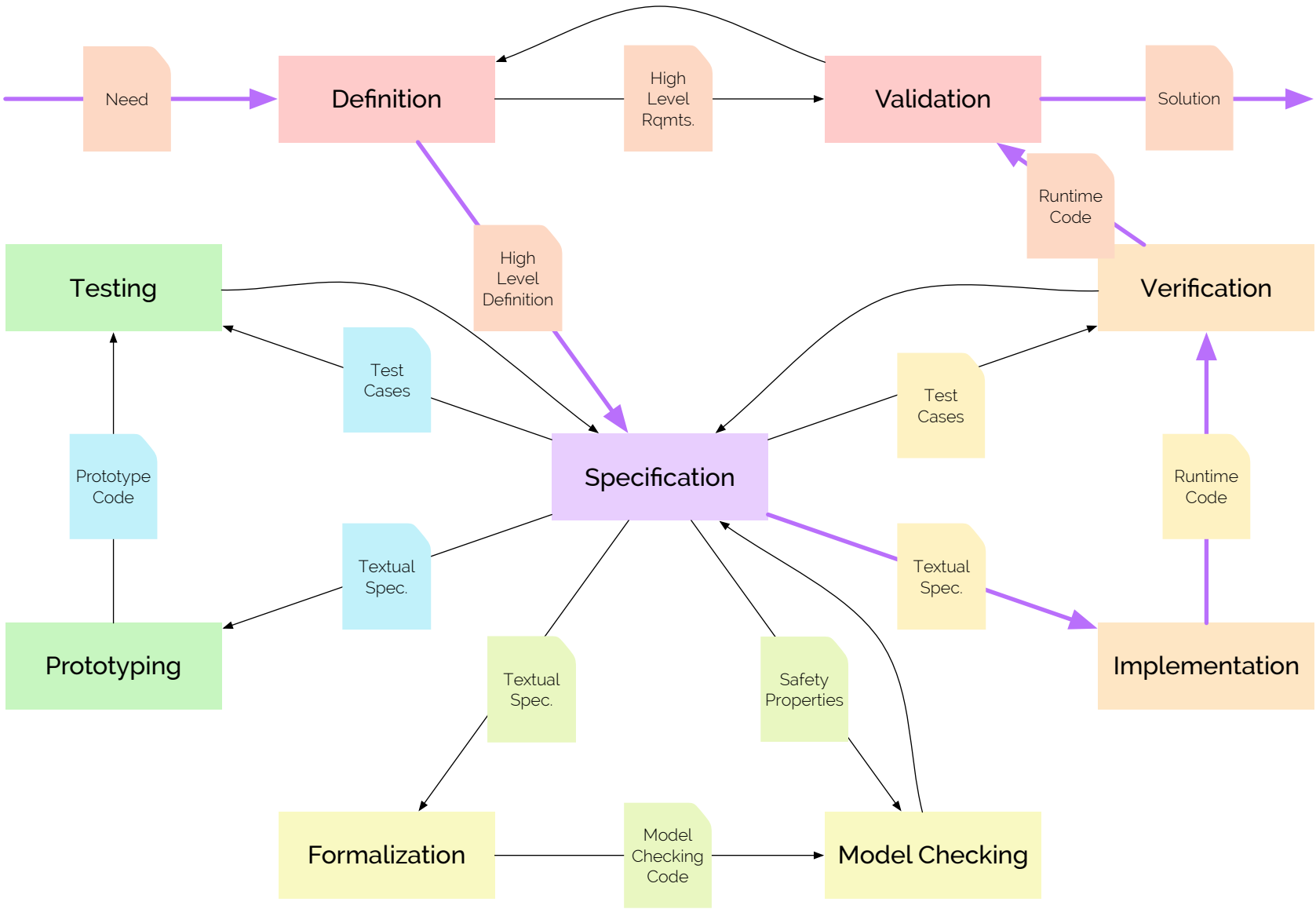
What

How

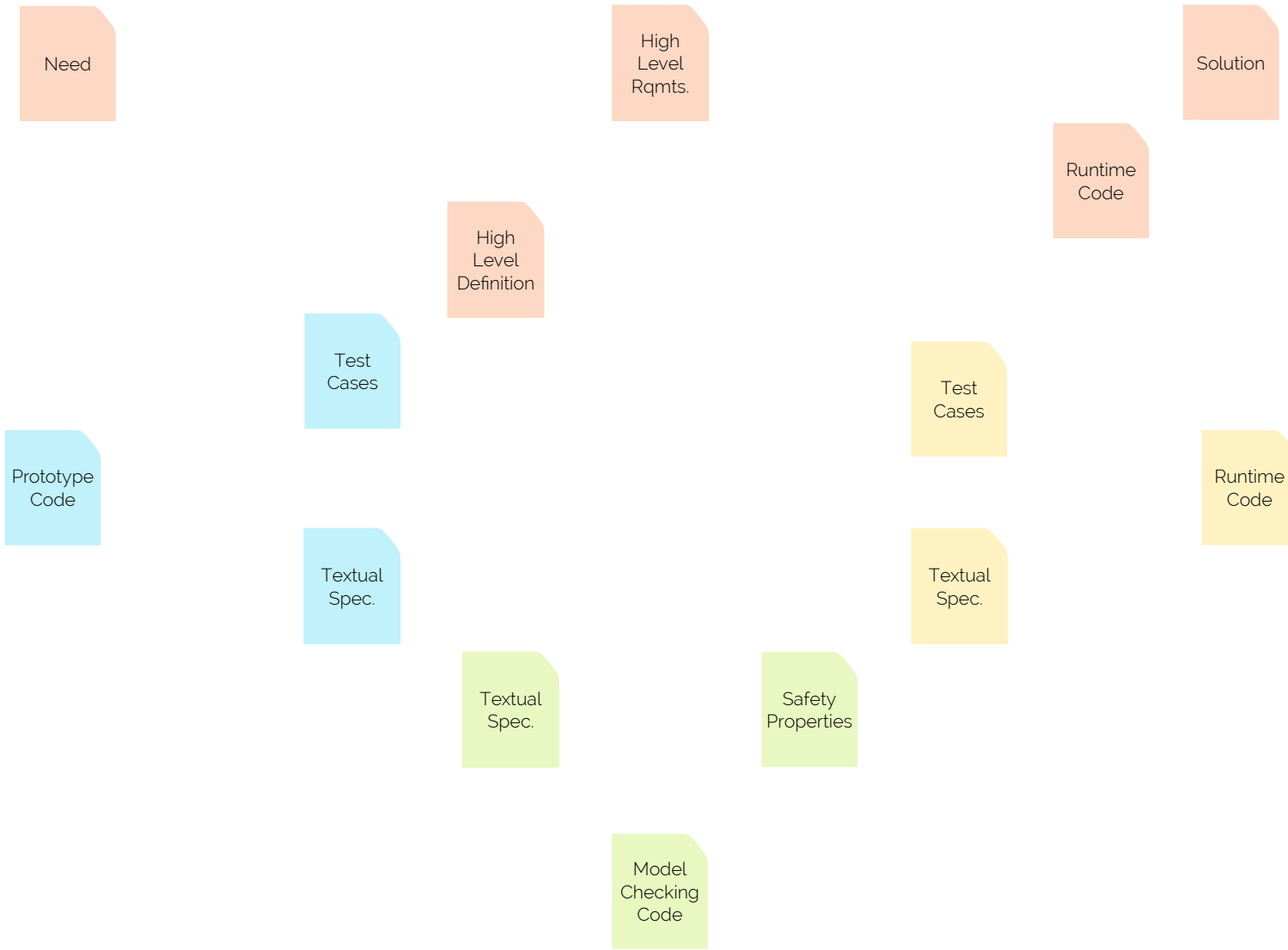
Context


Critical Systems
Formal proof
Static analysis
B Method
Model checking
State machines
Petri nets
Proven technologies
Long development cycles
Critical UIs
User Interfaces
Extensive testing
Usability analysis
Human factors
Look & Feel design
User experience
Prototyping
New modalities
Short development cycles

Context

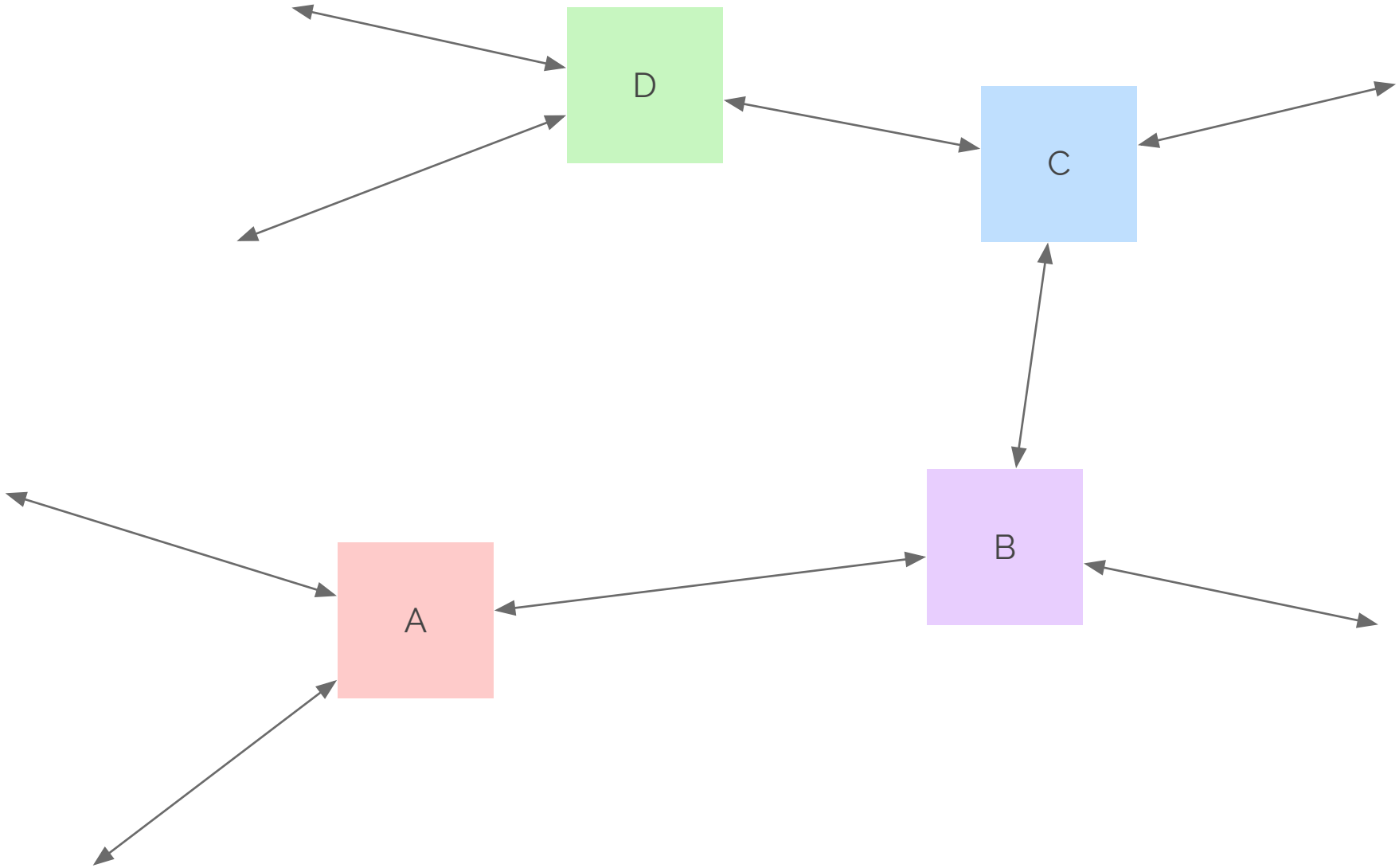

Need
Definition
High Level Rqmts.
Validation
Solution
Runtime Code
High Level Definition
Testing
Verification
Test Cases
Test Cases
Prototype Code
Specification
Runtime Code
Textual Spec.
Textual Spec.
Prototyping
Implementation
Textual Spec.
Safety Properties
Formalization
Model Checking Code
Model Checking

Context


Need
High Level Rqmts.
Solution
Runtime Code
High Level Definition
Test Cases
Test Cases
Prototype Code
Runtime Code
Textual Spec.
Textual Spec.
Textual Spec.
Safety Properties
Model Checking Code

Interactive systems

Interactive systems



Interactive systems

Data

Interactive systems

Data

Computation

Interactive systems

Data

Computation

Interaction

Interactive systems

Data

XML, JSON, CSV, HTML, SQL…

Computation

C, Lisp, Java, ML, Javascript, Lustre…

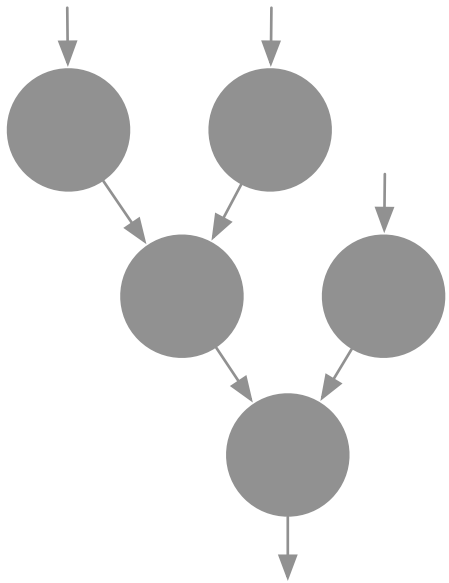
Interaction

?

Computation vs Interaction

Computation vs Interaction

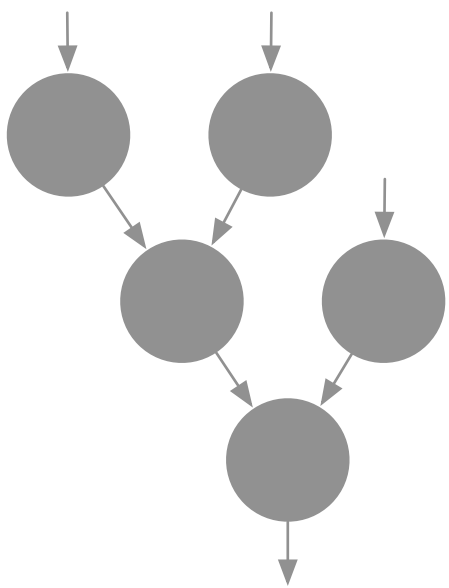
Parameters



Result

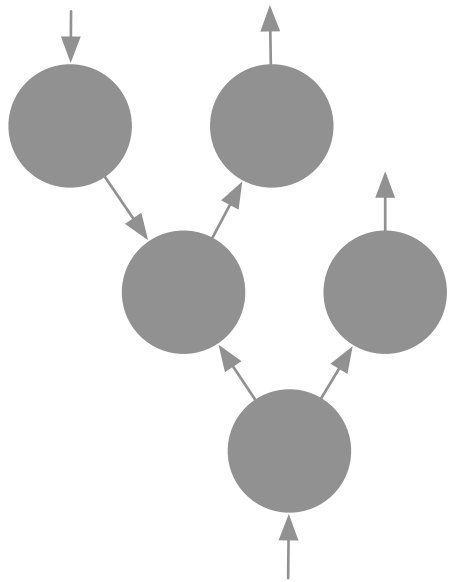
Computation vs Interaction

Parameters



Result

Simple interactions



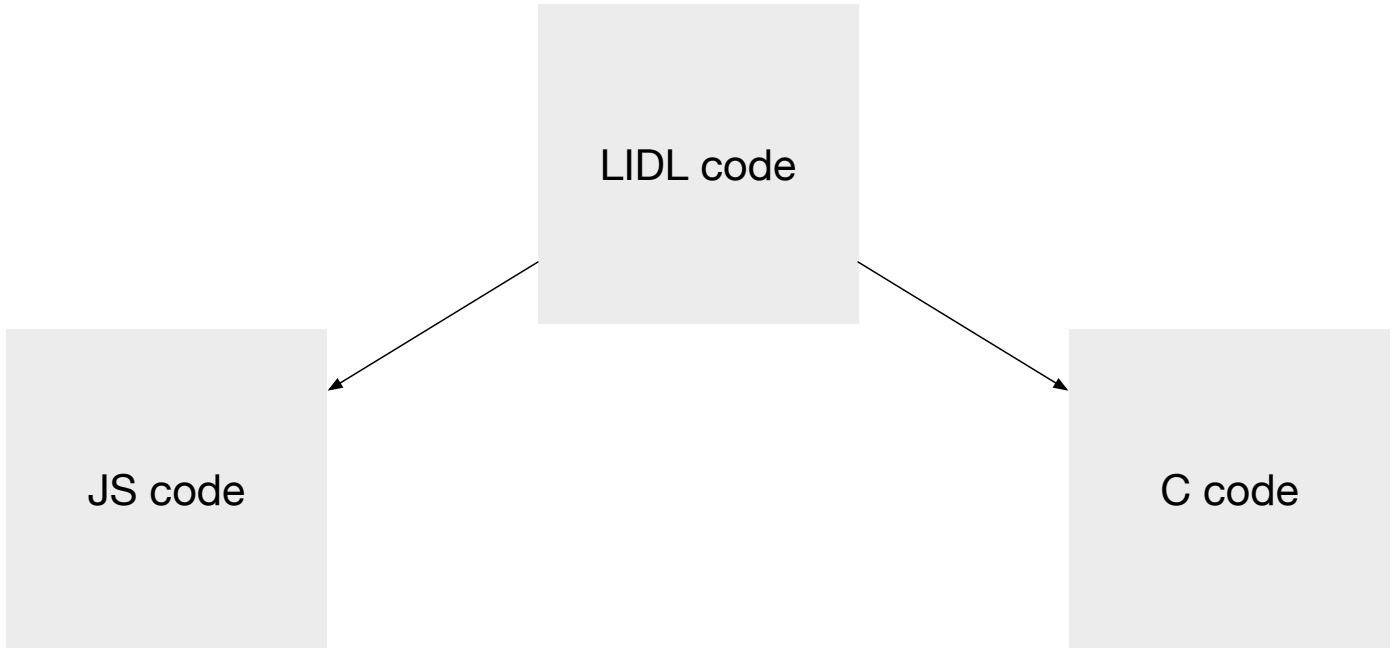
Compound interaction

Why

What

How

LIDL



LIDL Programs structure

LIDL Programs structure

```
data
    TheDataType
is
    …
```

LIDL Programs structure

```
data
    TheDataType
is
    …


interface
    TheInterface
is
    …
```

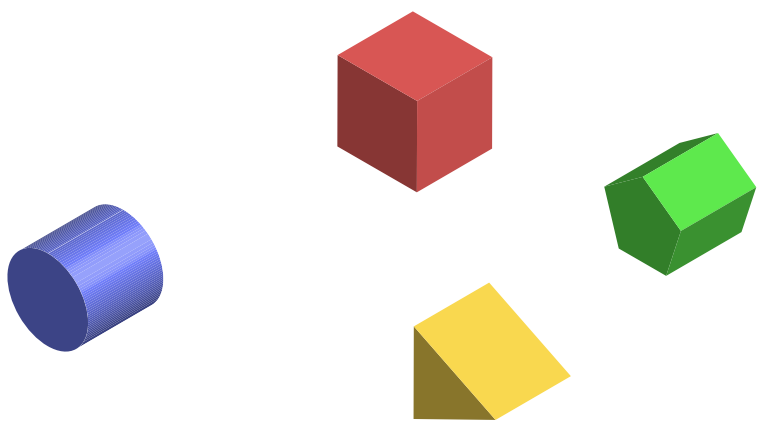
LIDL Programs structure

```
data
    TheDataType
is
    …


interface
    TheInterface
is
    …


interaction
    (say (something:Text in) to (someone:Text out)):Activation in
is
    …
```

Data types

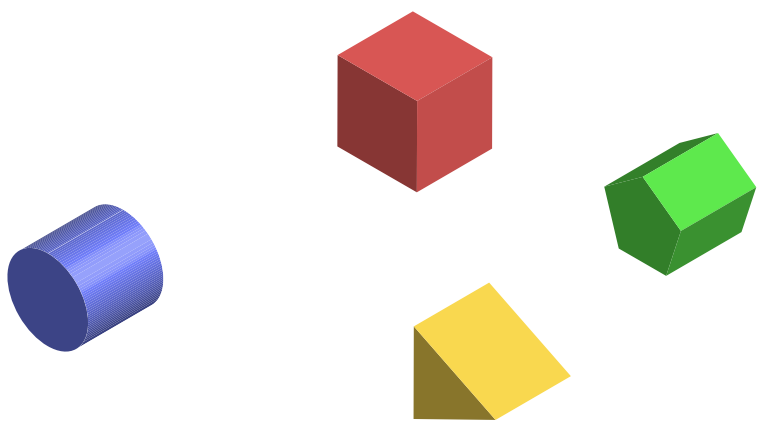
Data types

```
data Square
data Cylinder
data Pentagon
data Triangle
```

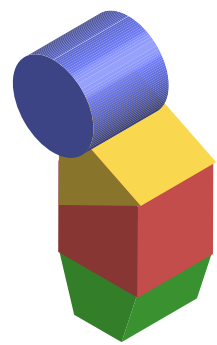


Data types

```
data Square
data Cylinder
data Pentagon
data Triangle
```

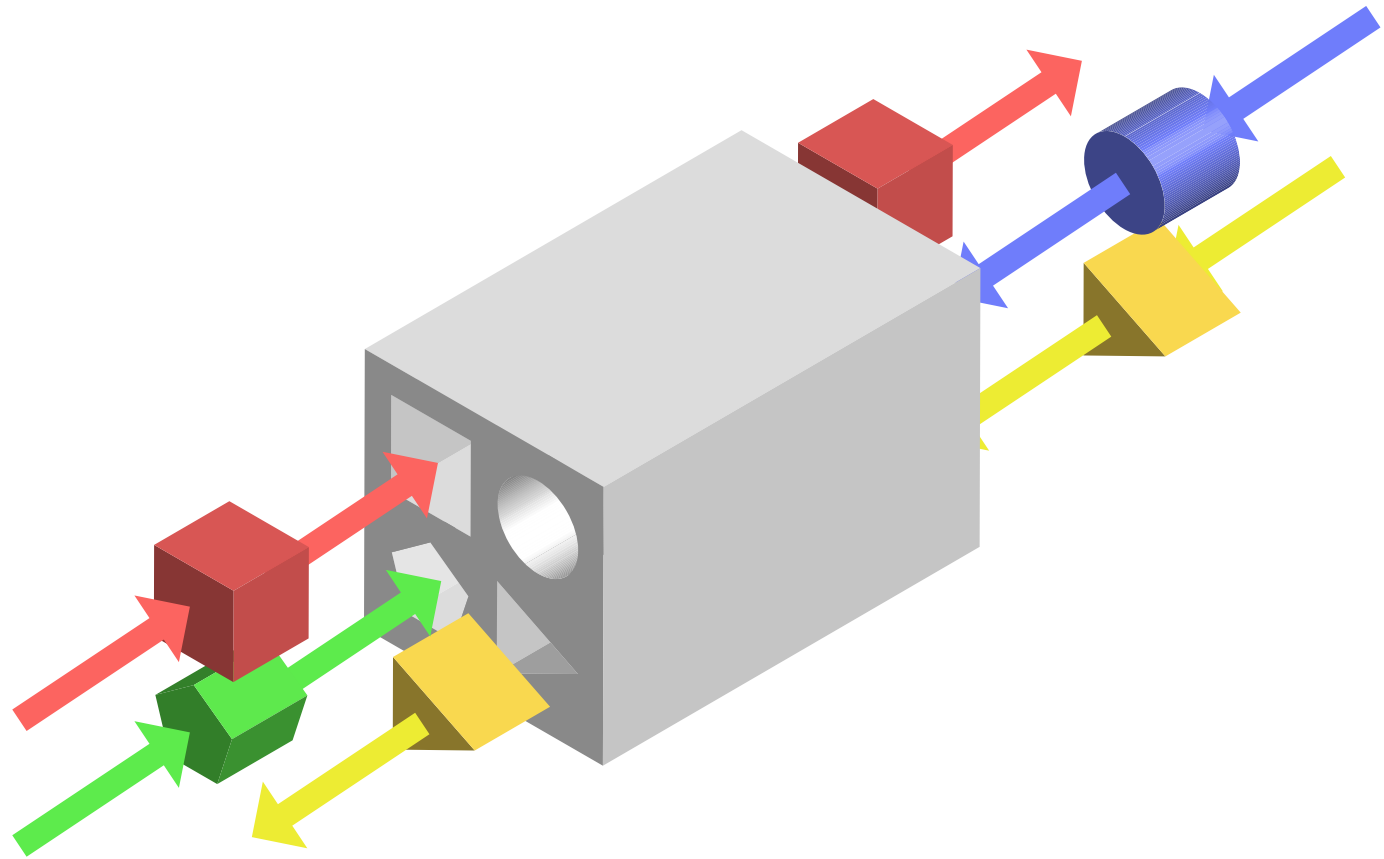


```
data MyCompoundType is
  {
    redSquare      : Square,
    greenPentagon  : Pentagon,
    yellowTriangle : Triangle,
    blueCylinder   : Cylinder
  }
```



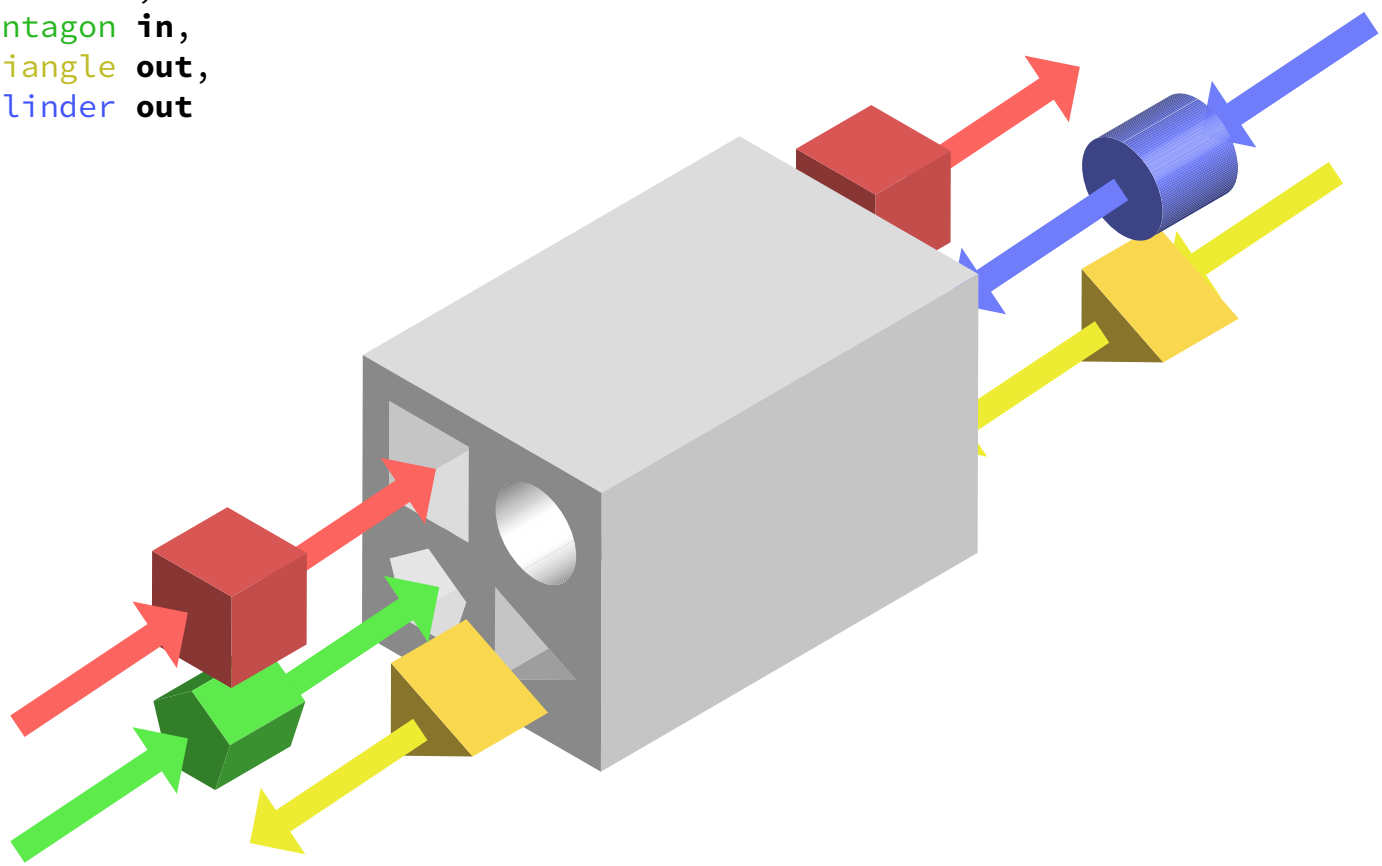
Interfaces

Interfaces



Interfaces

```
interface Example is
  {
    redSquares      : Square in,
    greenPentagons  : Pentagon in,
    yellowTriangles : Triangle out,
    blueCylinders   : Cylinder out
  }
```



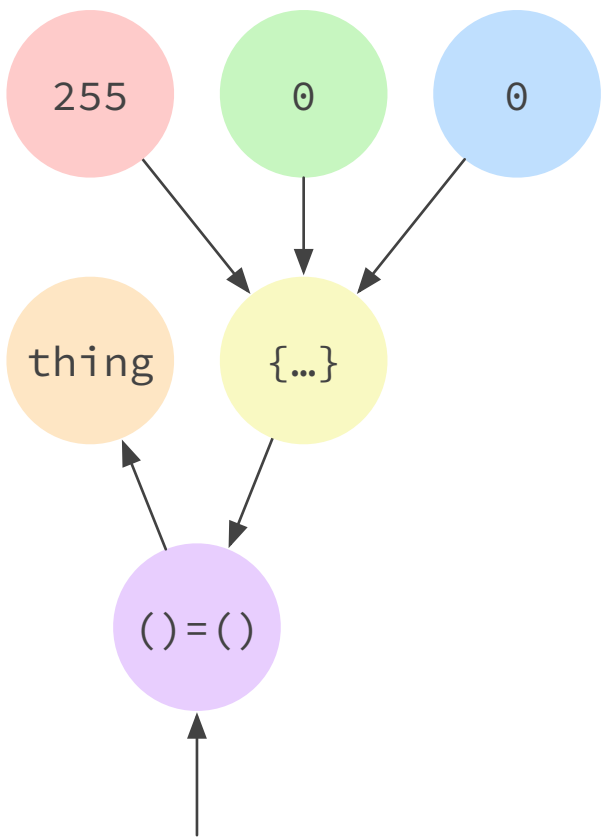
Interaction

Interaction

```
interaction
    (turn (thing: Color out) red): Activation in
is
    ((thing)=({red:(255),green:(0),blue:(0)}))
```

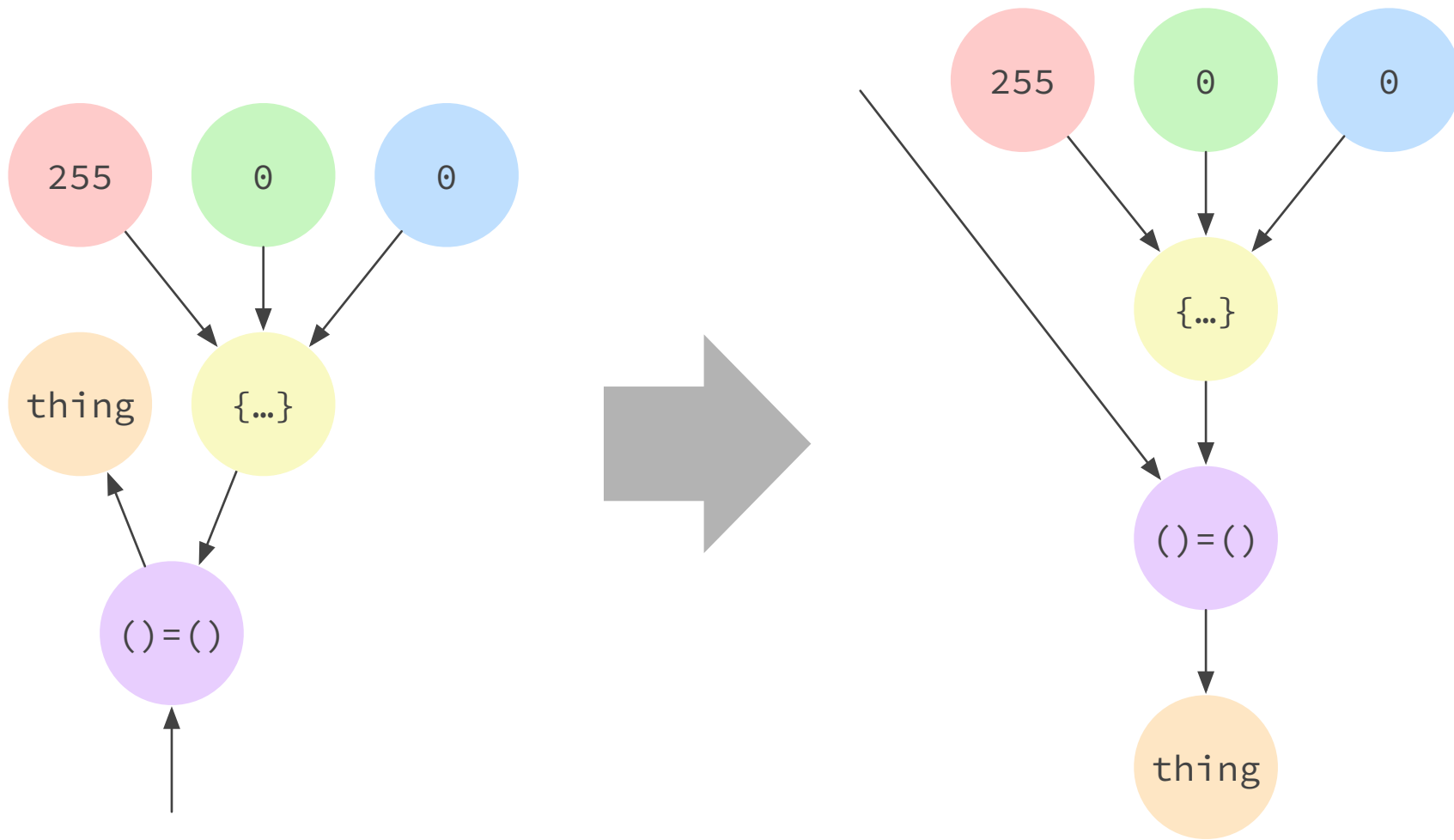
Interaction

```
interaction
    (turn (thing: Color out) red): Activation in
is
    ((thing)=({red:(255),green:(0),blue:(0)}))
```



Interaction

```
interaction
    (turn (thing: Color out) red): Activation in
is
    ((thing)=({red:(255),green:(0),blue:(0)}))
```



Syntax

Syntax

C, Java

```
computeBMI(83,185)
```

Syntax

C, Java

```
computeBMI(83,185)
```

Javascript

```
computeBMI({weight:83,height:185})
```

Syntax

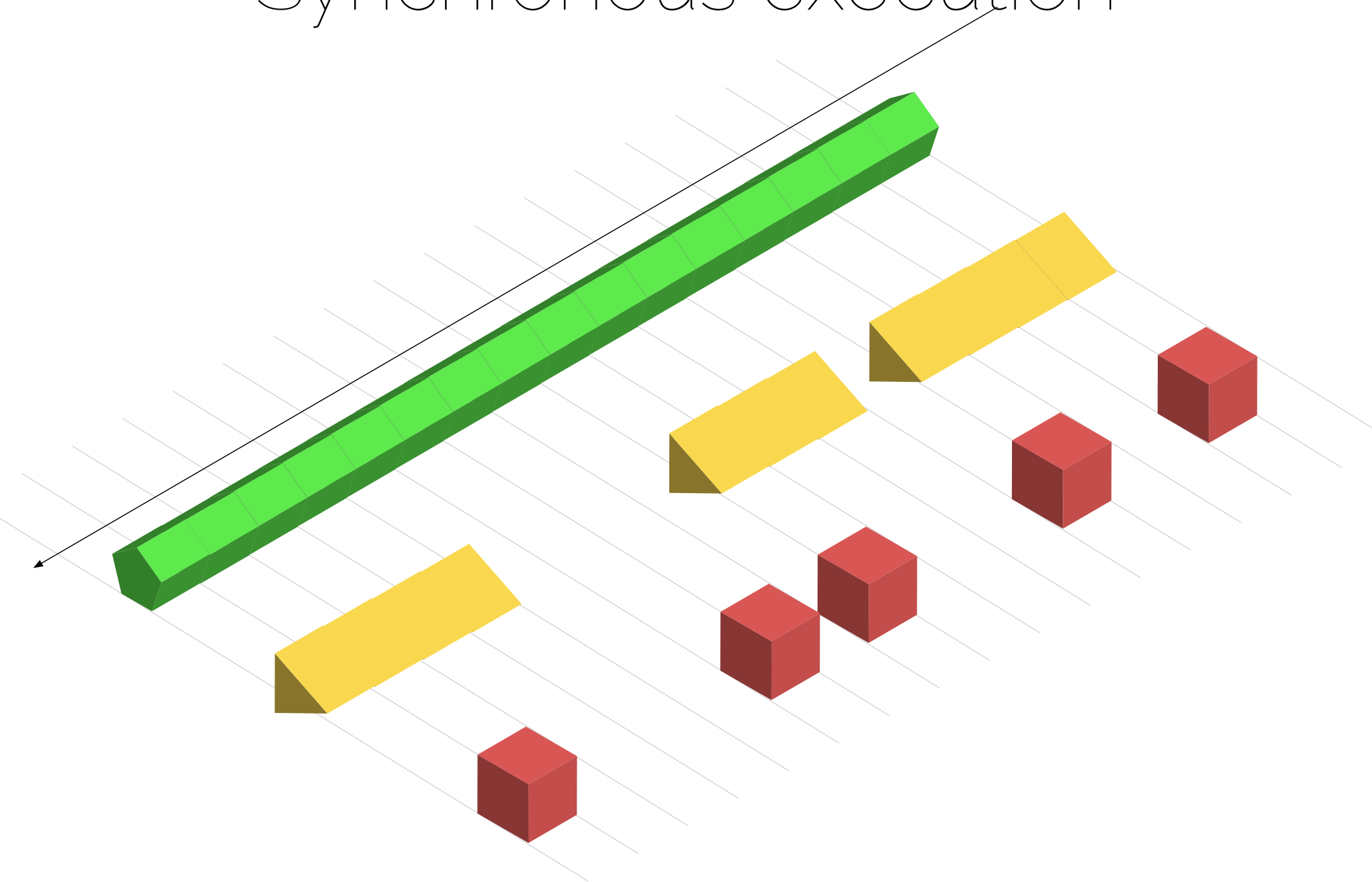
C, Java	`computeBMI(83,185)`
Javascript	`computeBMI({weight:83,height:185})`
C#, Swift	`computeBMI(weight:83,height:185)`

Syntax

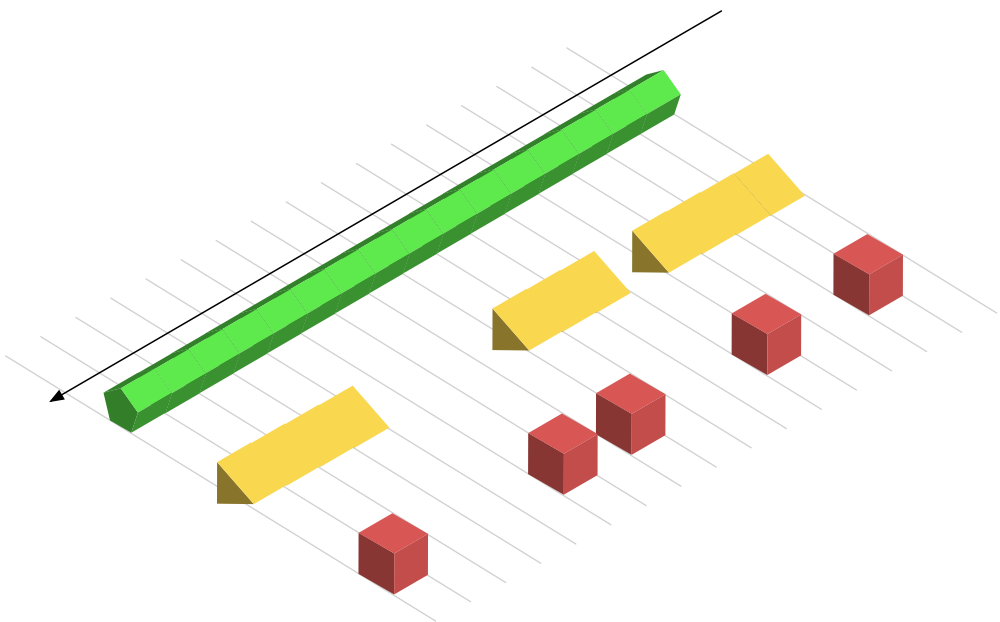
C, Java	`computeBMI(83,185)`
Javascript	`computeBMI({weight:83,height:185})`
C#, Swift	`computeBMI(weight:83,height:185)`
LIDL	`(BMI of someone who weights (83)kg and is (185)cm high)`

Synchronous execution

Synchronous execution



Synchronous execution



pentagon	triangle	square
2	4	~
3	5	~
3	~	~
2	~	2
5	~	~
5	2	~
2	3	~
3	1	~
4	~	~

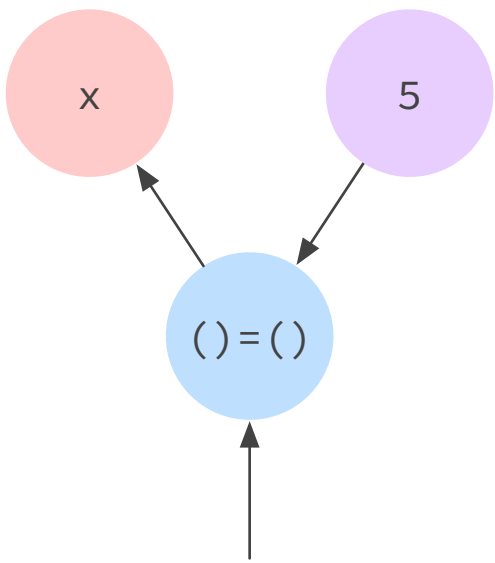
Everything is an interaction

Everything is an interaction

((x)=(5))

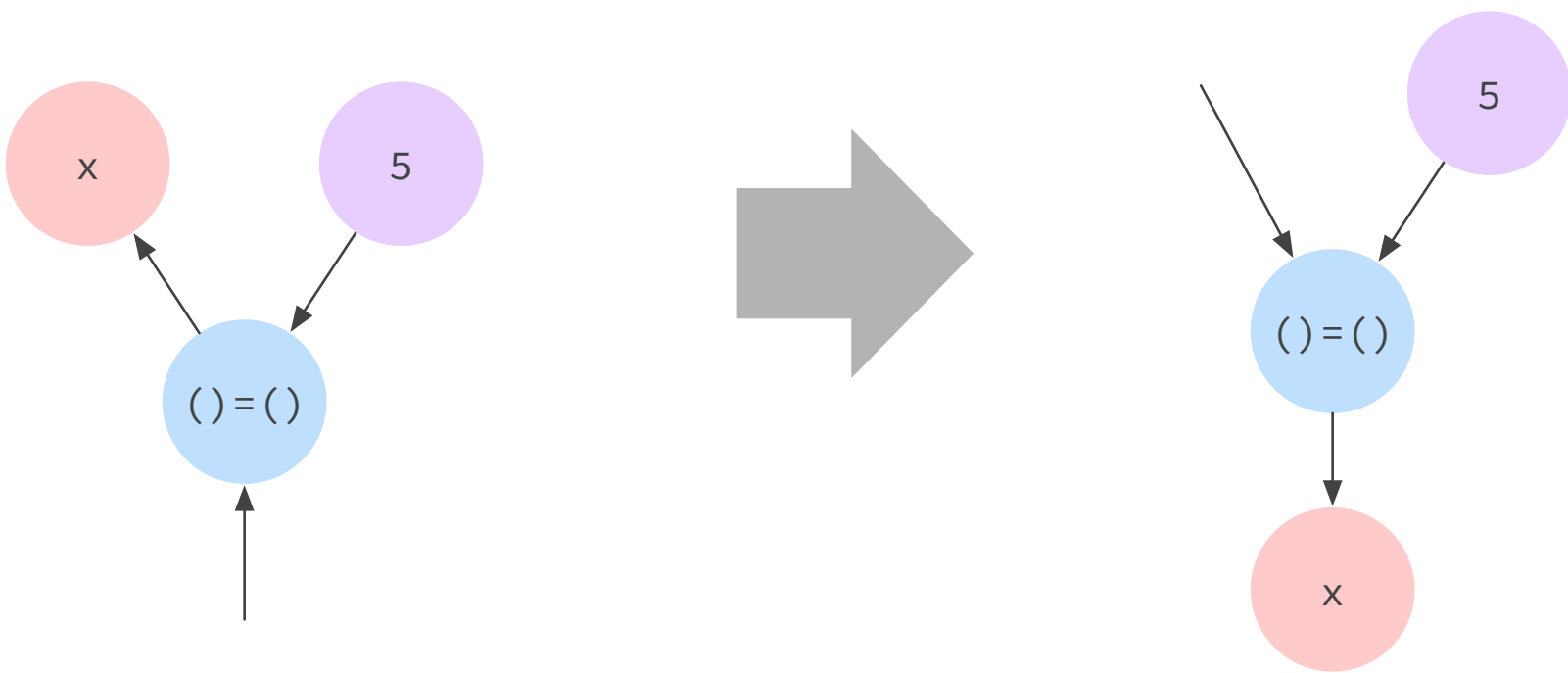
Everything is an interaction

((x)=(5))



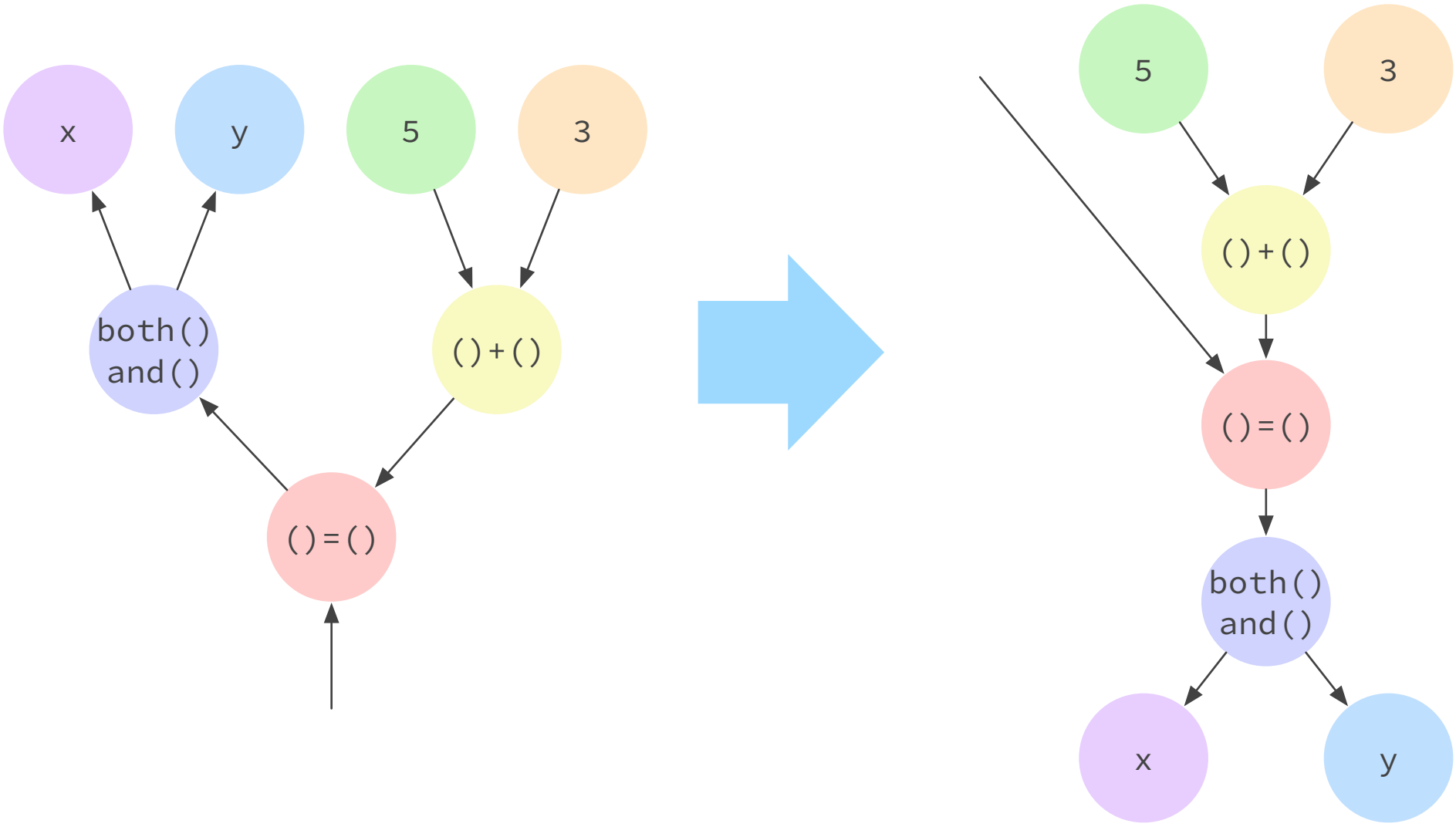
Everything is an interaction

((x)=(5))



Everything is an interaction

((both(x)and(y))=((5)+(3)))



Base interactions

Composition/Decomposition

```
({x()y()})
```

Selection/Deselection

```
(().x)
```

Previous

```
(previous())
```

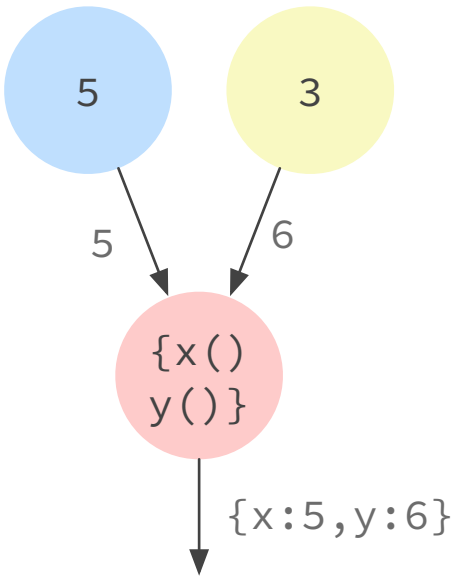
Function application

```
(()in()()=())
```

Composition/Decomposition

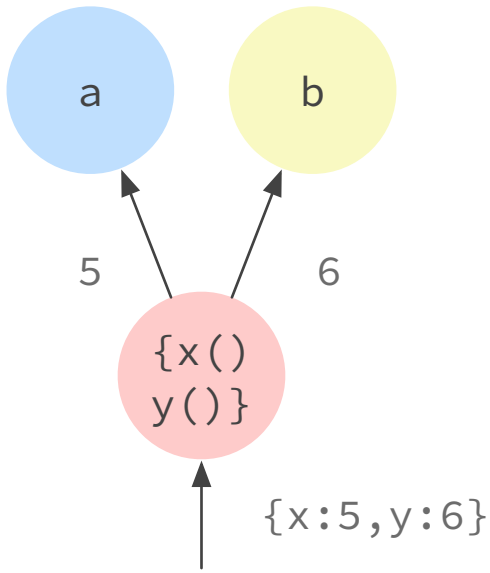
Composition

```
({x(5)y(6)})
```



Decomposition

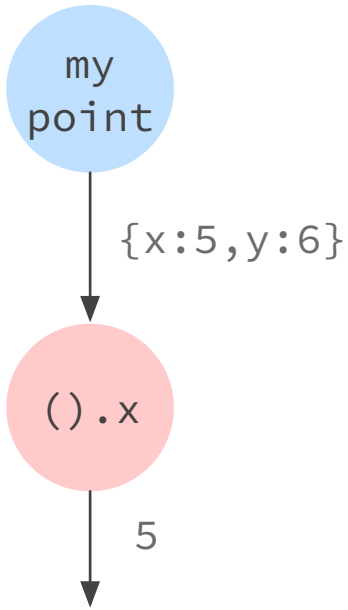
```
({x(a)y(b)})
```



Selection/Deselection

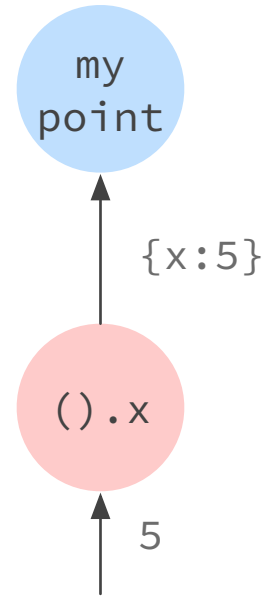
Selection

```
((mypoint).x)
```



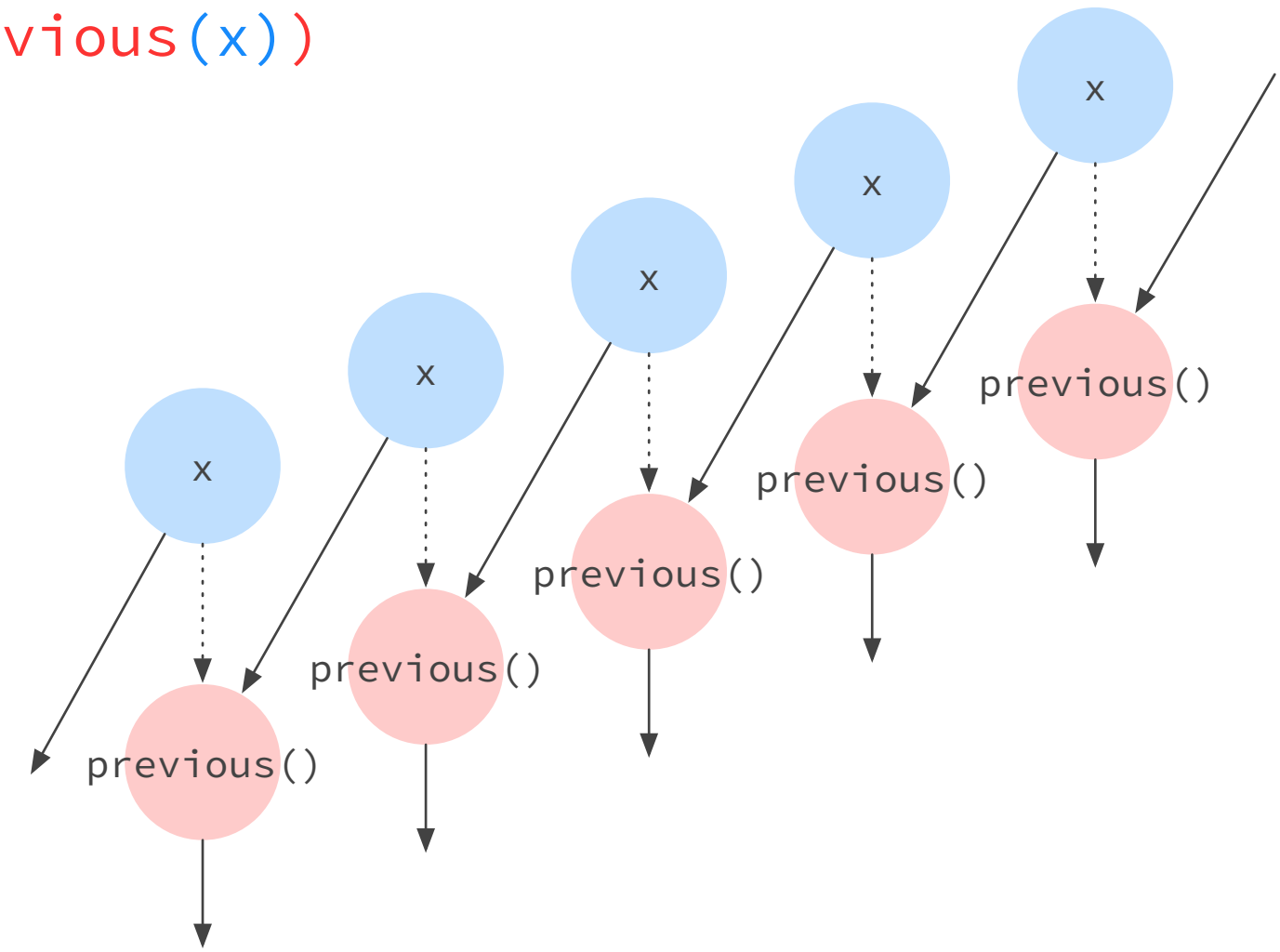
Deselection

```
((mypoint).x)
```



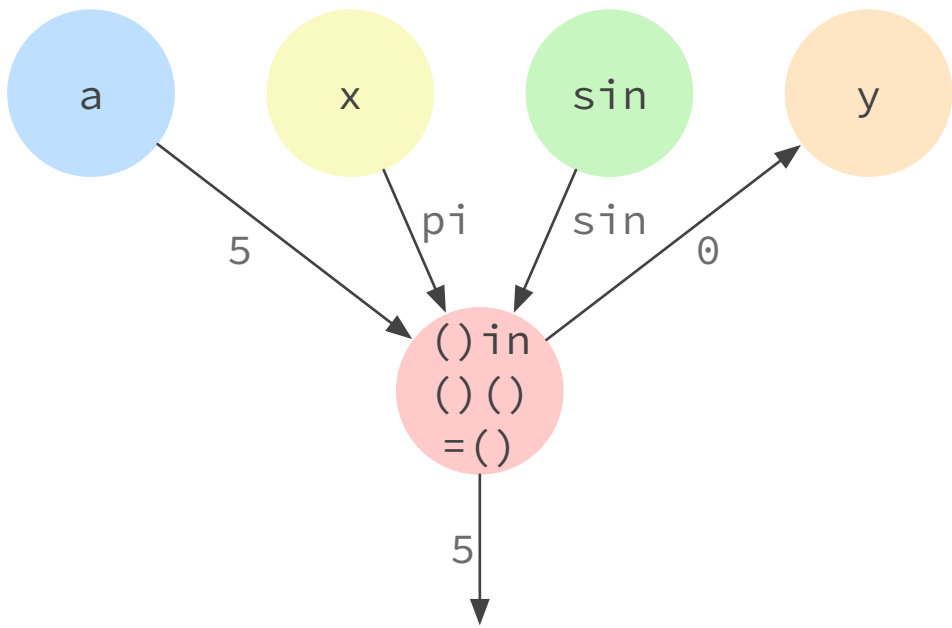
Previous

```
(previous(x))
```



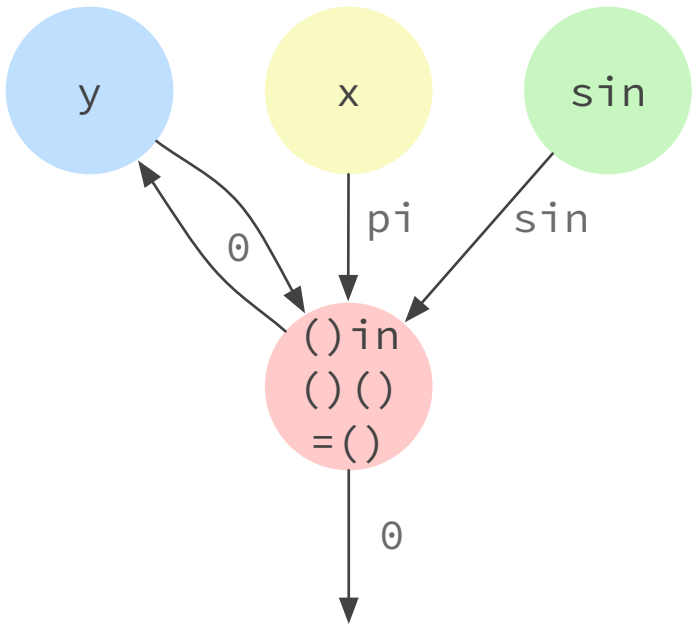
Function application

```
((a)in(sin)(x)=(y))
```

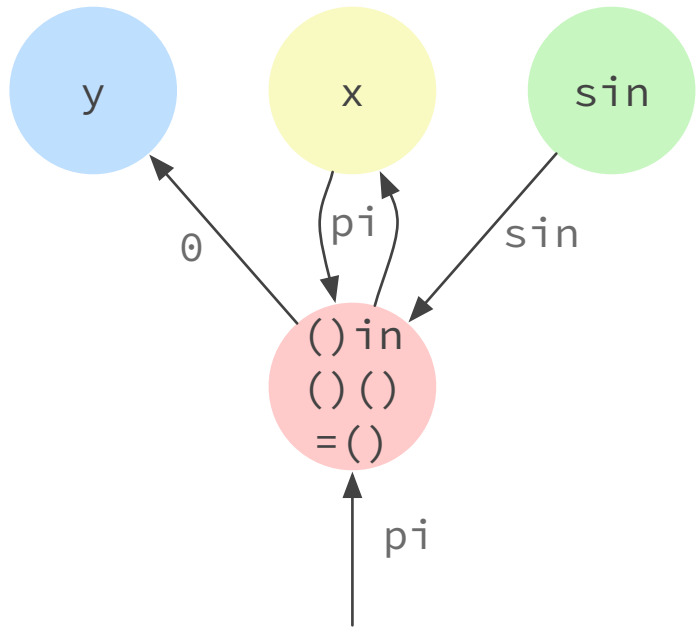


Function application

```
((y)in(sin)(x)=(y))
```




```
((x)in(sin)(x)=(y))
```



Summary

LIDL

Generally describe any interactive system

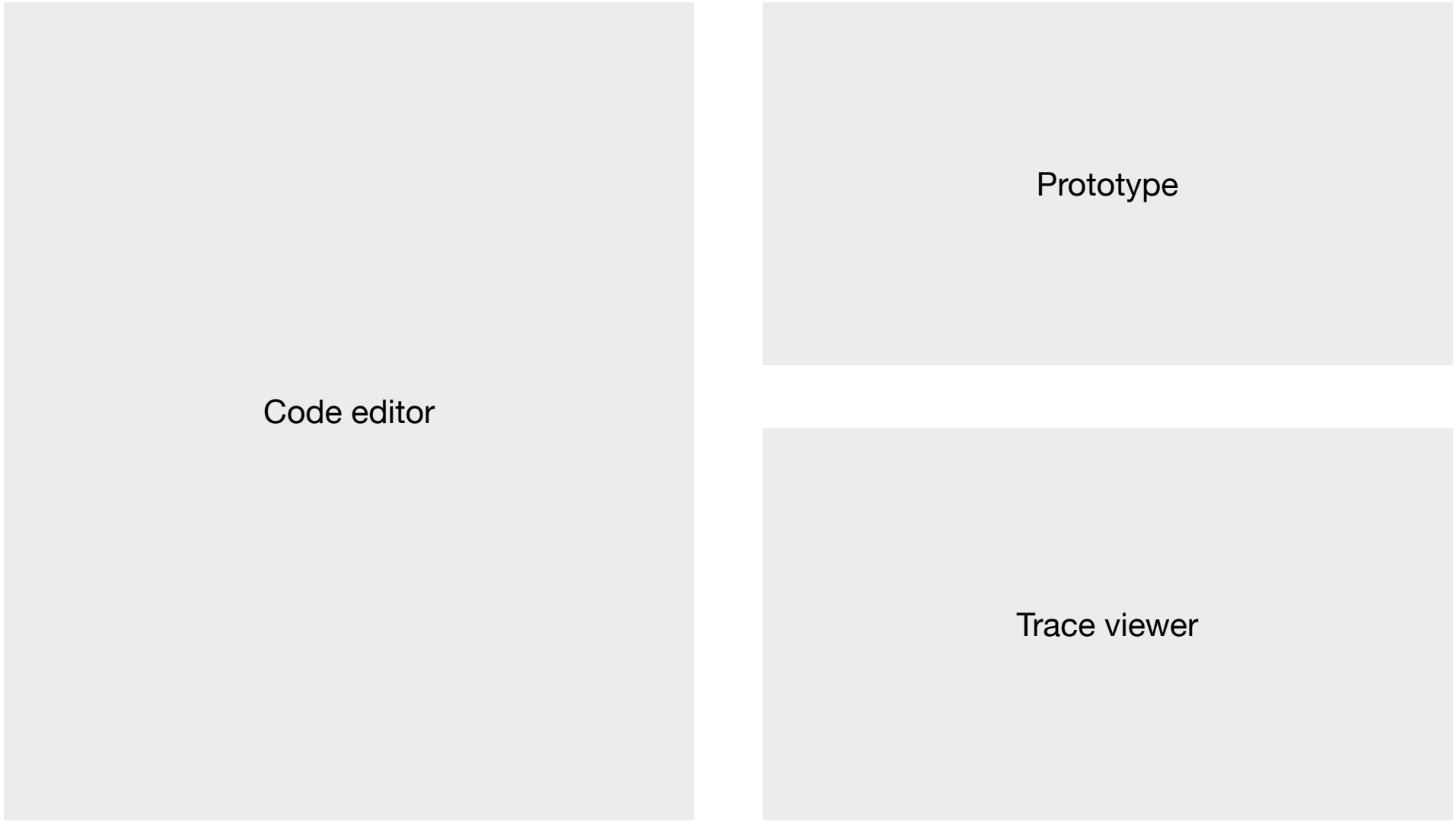
Based on the concept of interface and interaction

Synchronous execution

Declarative

Think differently

LIDL workshop



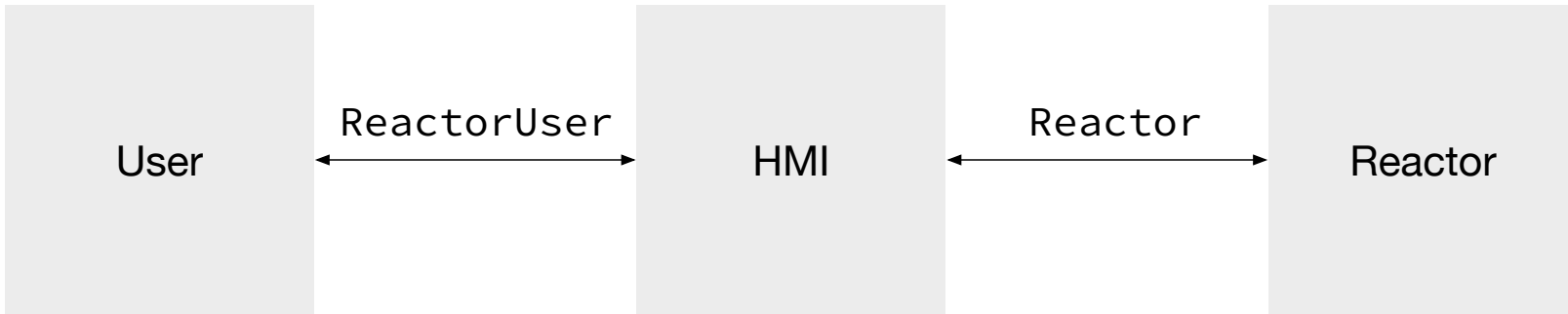
Why

What

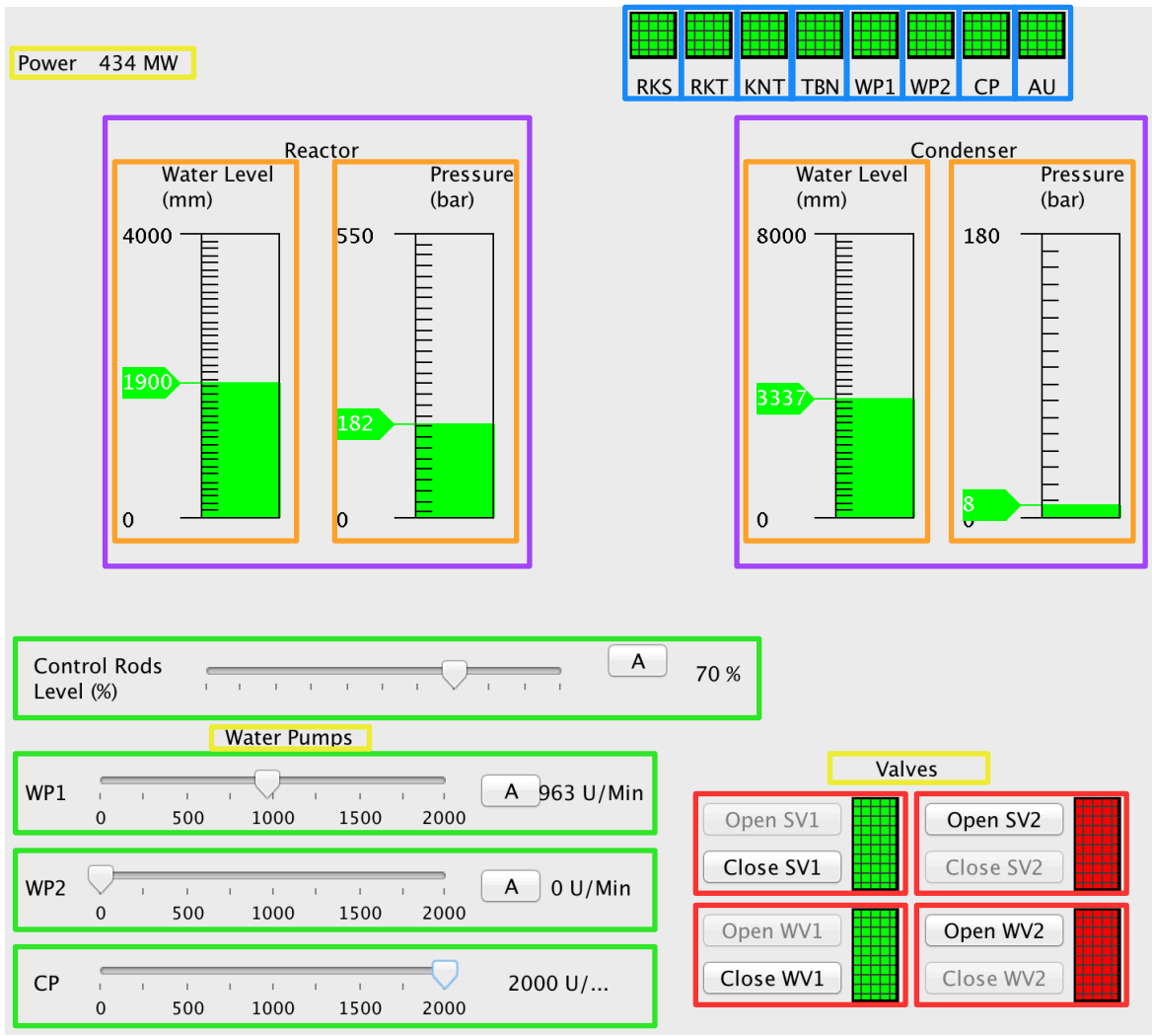
How

Using LIDL

Using LIDL

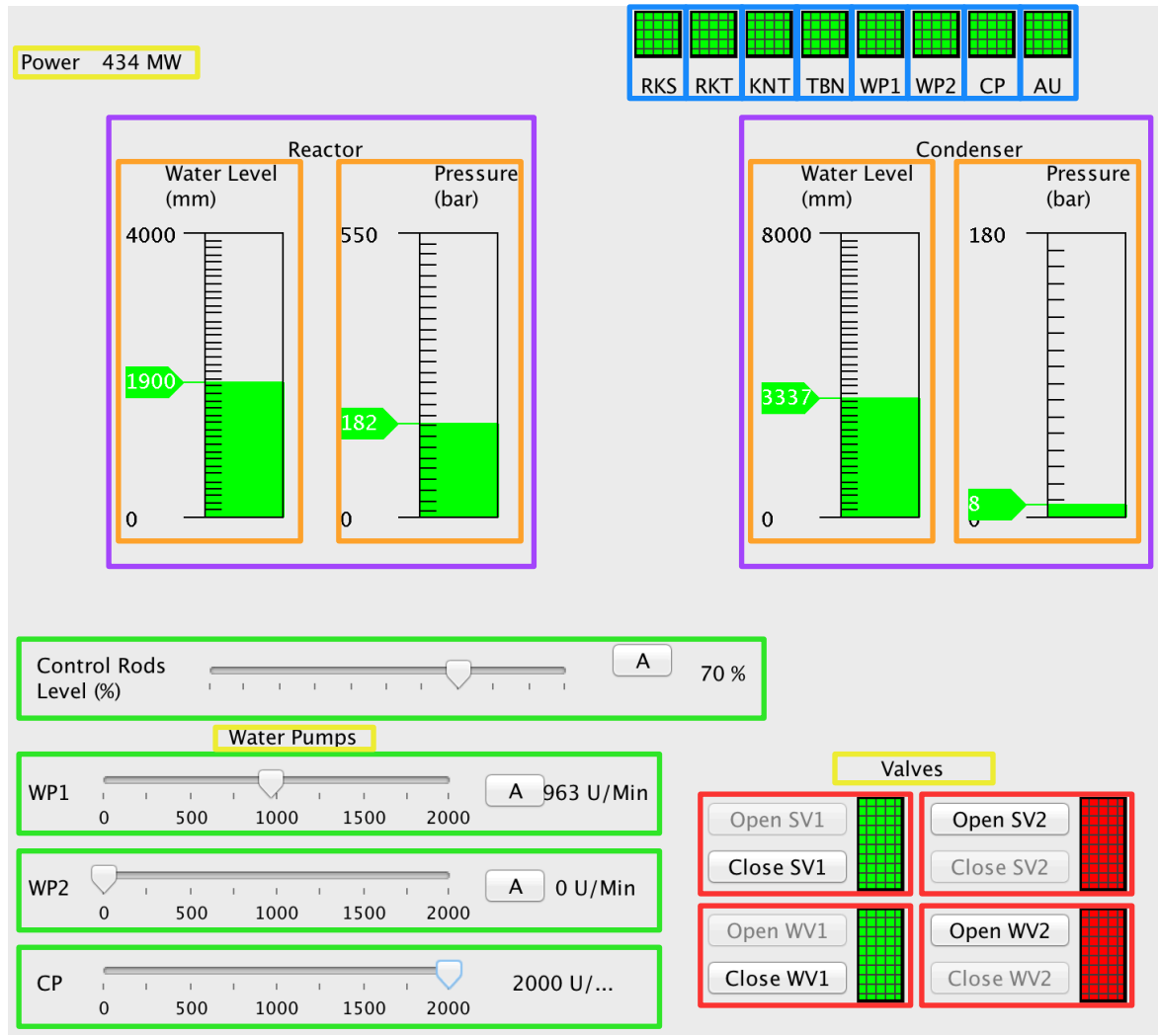


Using LIDL


Power 434 MW
RKS
RKT
KNT
TBN
WP1
WP2
CP
AU
Reactor
Water Level (mm)
4000
1900
0
Pressure (bar)
550
182
0
Condenser
Water Level (mm)
8000
3337
0
Pressure (bar)
180
8
Control Rods Level (%)
A
70 %
Water Pumps
WP1
0
500
1000
1500
2000
A
963 U/Min
WP2
0
500
1000
1500
2000
A
0 U/Min
CP
0
500
1000
1500
2000
2000 U/...
Valves
Open SV1
Close SV1
Open SV2
Close SV2
Open WV1
Close WV1
Open WV2
Close WV2

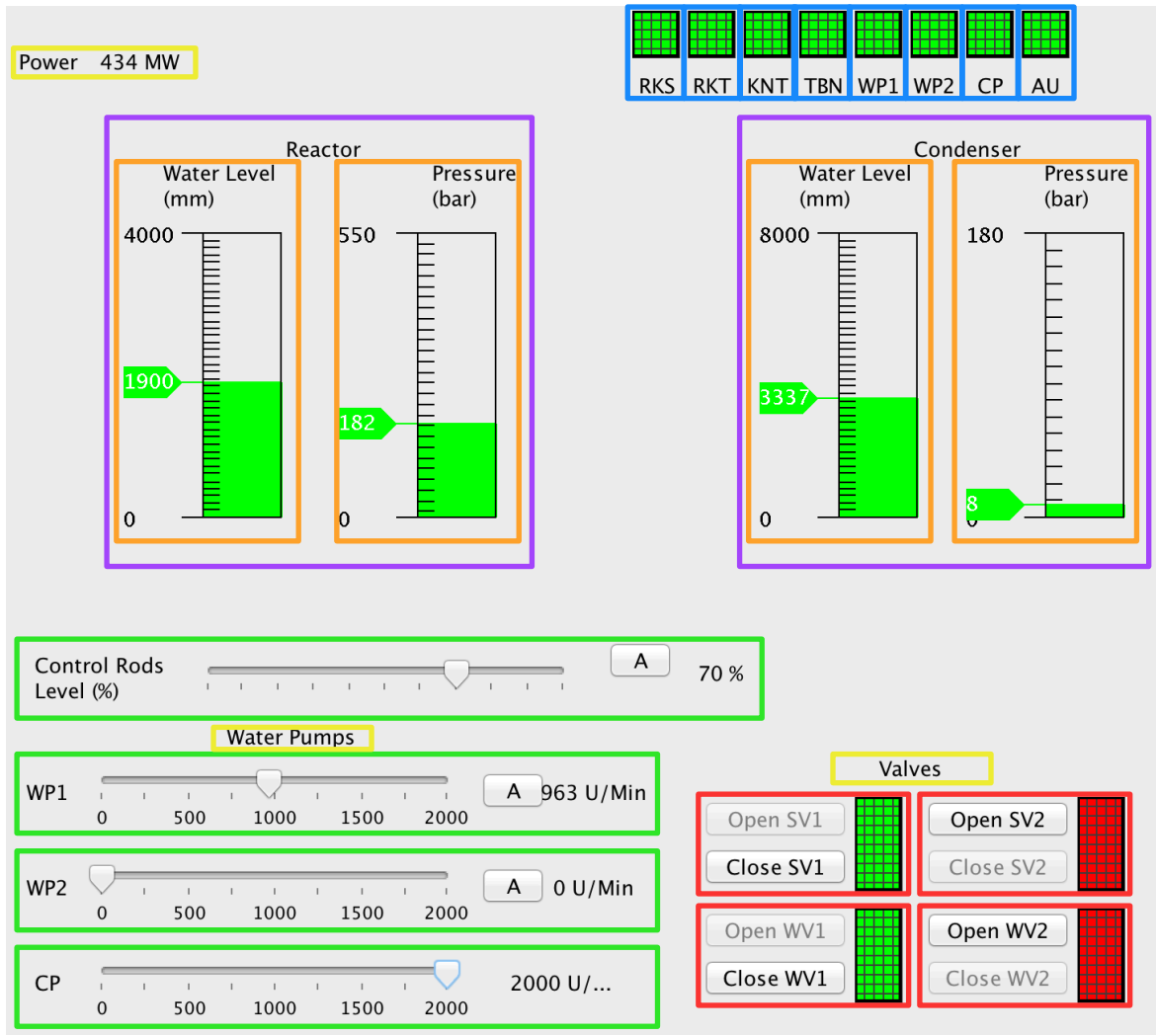
Using LIDL

```
interface Reactor is
{
  command:{
    sv1: Boolean,
    sv2: Boolean,
    wv1: Boolean,
    wv2: Boolean,
    wp1: Number,
    wp2: Number,
    cp: Nubmer,
    rodPosition: Number
  } in,
  status:{
    sv1: Boolean,
    sv2: Boolean,
    wv1: Boolean,
    wv2: Boolean,
    cpUmin: Number,
    wp1Umin:Number,
    wp2Umin:Number,
    rodPosition:Number,
    outputPower:Number,
    reactorWaterLevel:Number,
    reactorPressure:Number,
    condenserWaterLevel:Number,
    condenserPressure:Number
  } out
}
```



Using LIDL

```
interface ReactorUser is
{
  powerDisplay: Label,
  reactor: DualGaugeWidget,
  condenser: DualGaugeWidget,
  controlRods: ComplexSlider,
  wp1: ComplexSlider,
  wp2: ComplexSlider,
  cp: ComplexSlider,
  sv1: ValveWidget,
  sv2: ValveWidget,
  wv1: ValveWidget,
  wv1: ValveWidget,
  leds: MultipleLedWidget
}
```



Using LIDL

```
interaction
  (human machine interface connecting (user:ReactorUser) to (reactor:Reactor)):Activation in
is
  ((user)=({
    powerDisplay:
      (label (active) displaying (reactor.status.outputPower))
    reactor:
      (dual gauge (active) with water level (reactor.status.reactorWaterLevel))
    condenser:
      …
      …
    }))
```

```
interface ReactorUser is
{
  powerDisplay: Label,
  reactor: DualGaugeWidget,
  condenser: DualGaugeWidget,
  controlRods: ComplexSlider,
  wp1: ComplexSlider,
  wp2: ComplexSlider,
  cp: ComplexSlider,
  sv1: ValveWidget,
  sv2: ValveWidget,
  wv1: ValveWidget,
  wv1: ValveWidget,
  leds: MultipleLedWidget
}
```

Using LIDL

```
interface Task is
  {
    start: Activation in,
    abort: Activation in,
    progress: Number out,
    running: Activation out,
    finished: Activation out
  }

interaction
  (reactor user manual (reactor:co(ReactorUser))):Task
is
  ( sequentially
    (start up (reactor))
    (operate (reactor))
    (shut down (reactor))
  )

interaction
  (start up (reactor:co(ReactorUser))):Task
is
  ( sequentially
    ( all
      (open valve (reactor.sv1))
      (open valve (reactor.wv1))
    )
    ( all
      ( set slider (reactor.wp1) to (1000))
      ( set slider (reactor.cp) to (1000))
    )
    ( set slider (reactor.controlRods) to (90))
  )
```

Questions