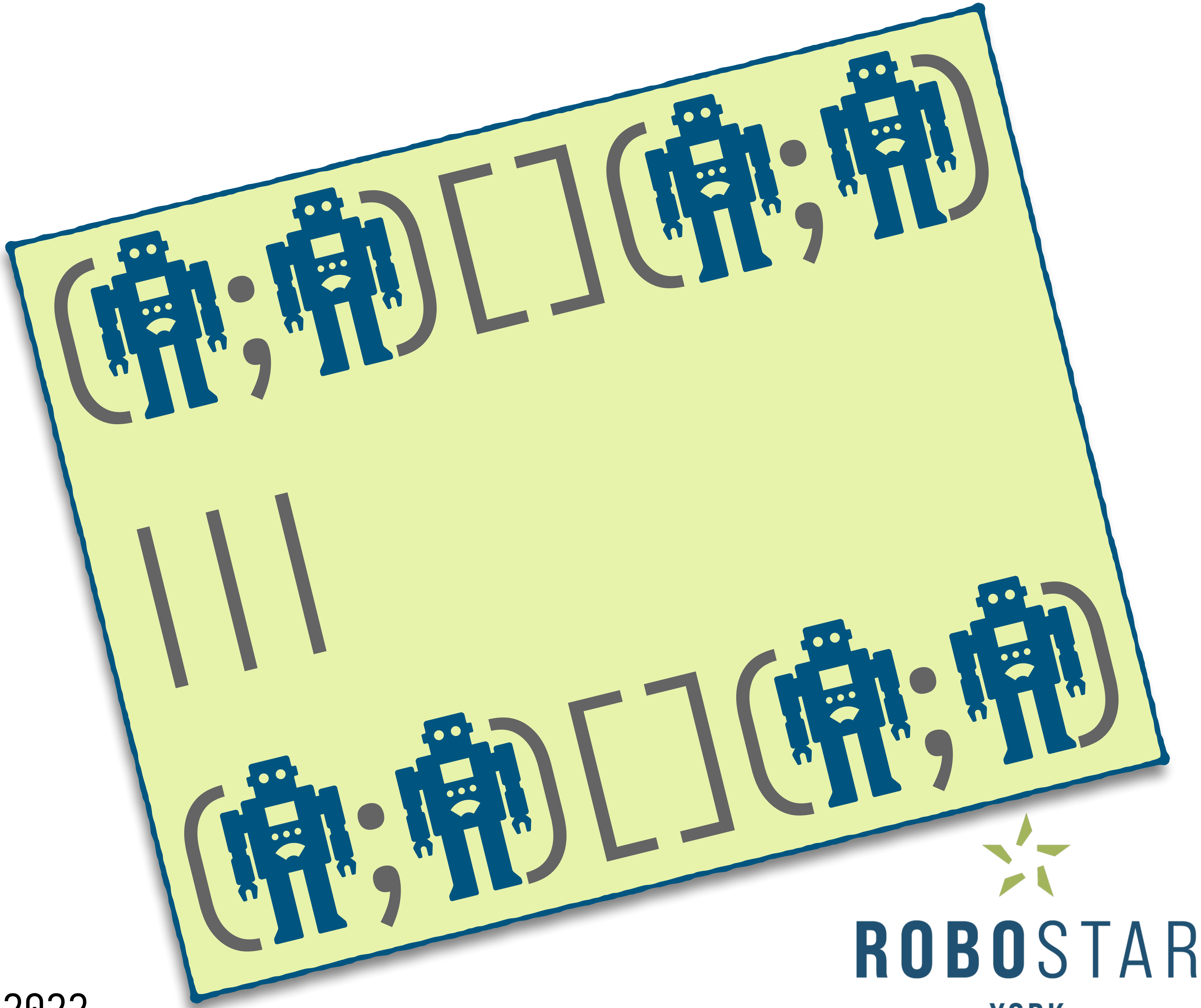
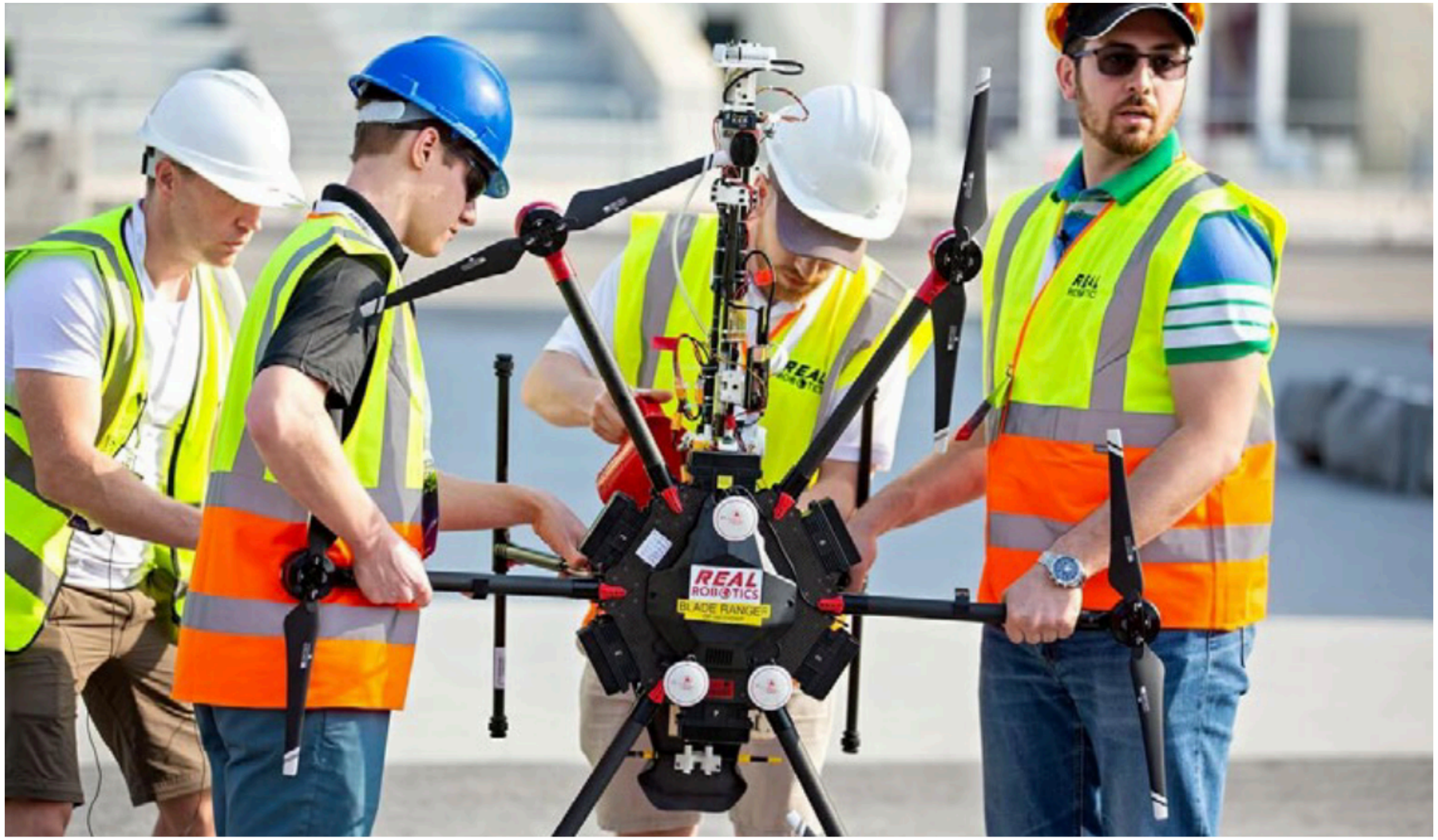
CSP in Practice

Timed Verification *of* Robot Software

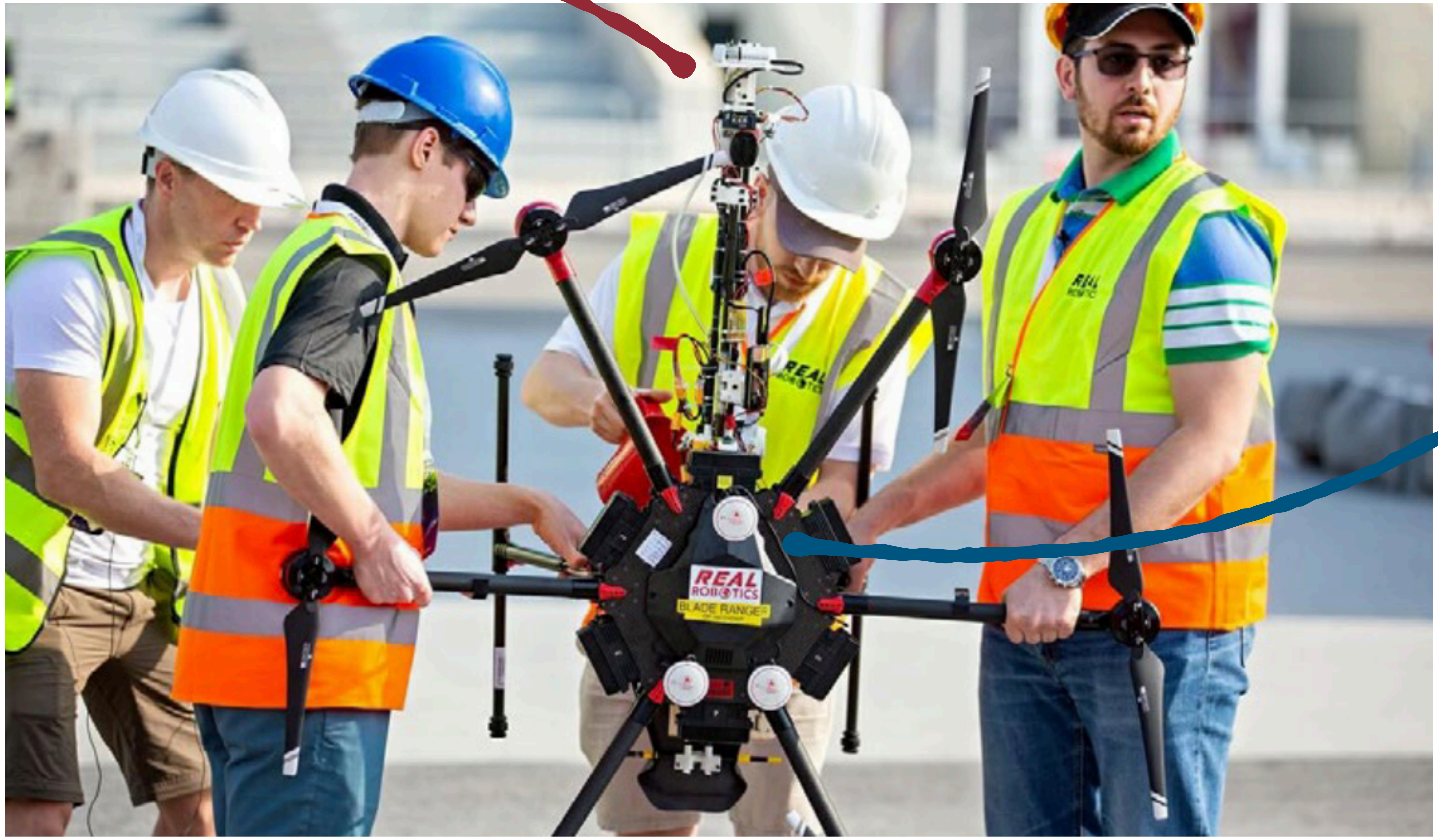


Matt Windsor (they/he), Concurrency Workshop 2022

ROBOSTAR
YORK



REAL ROBOTICS
BLADE RANGER



nozzle controller

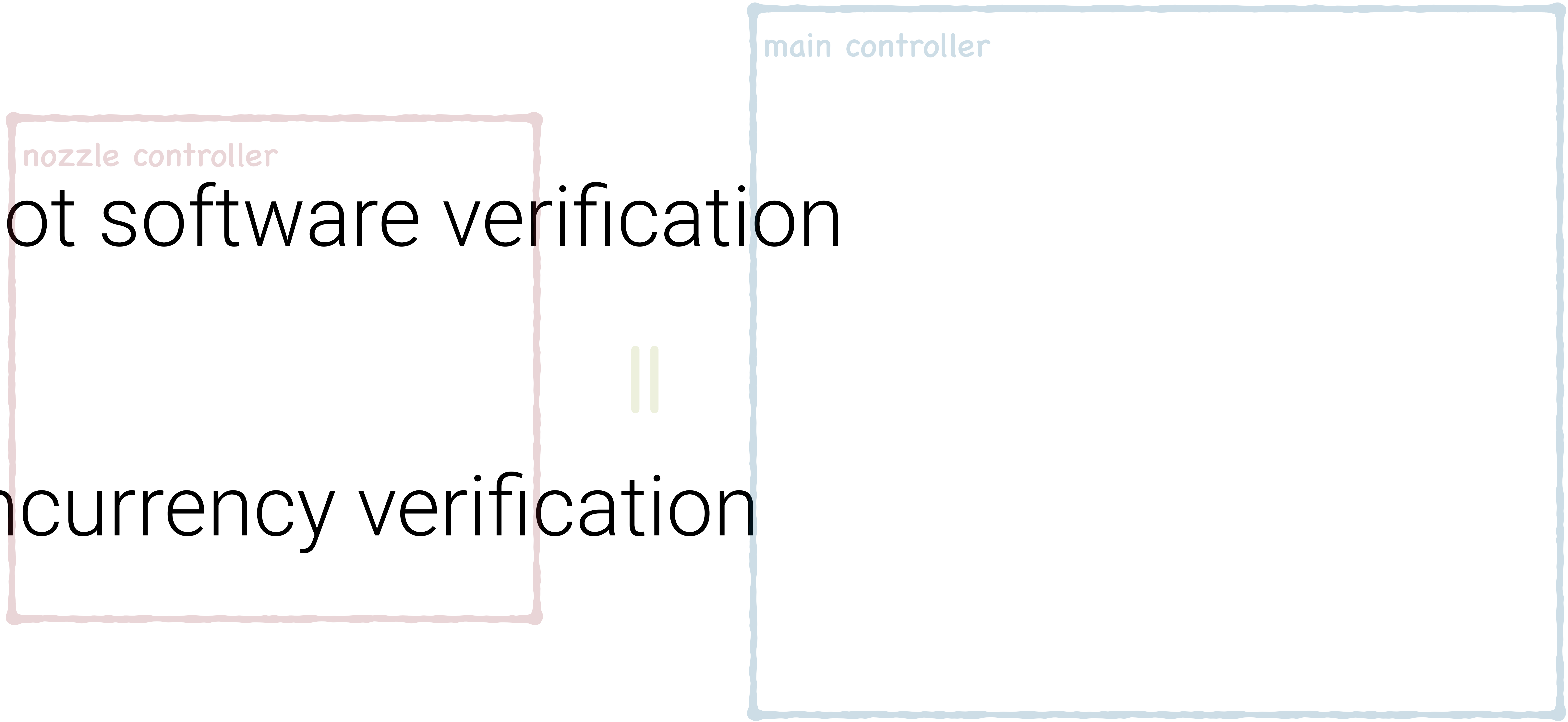
main controller

ROBOSTAR
YORK

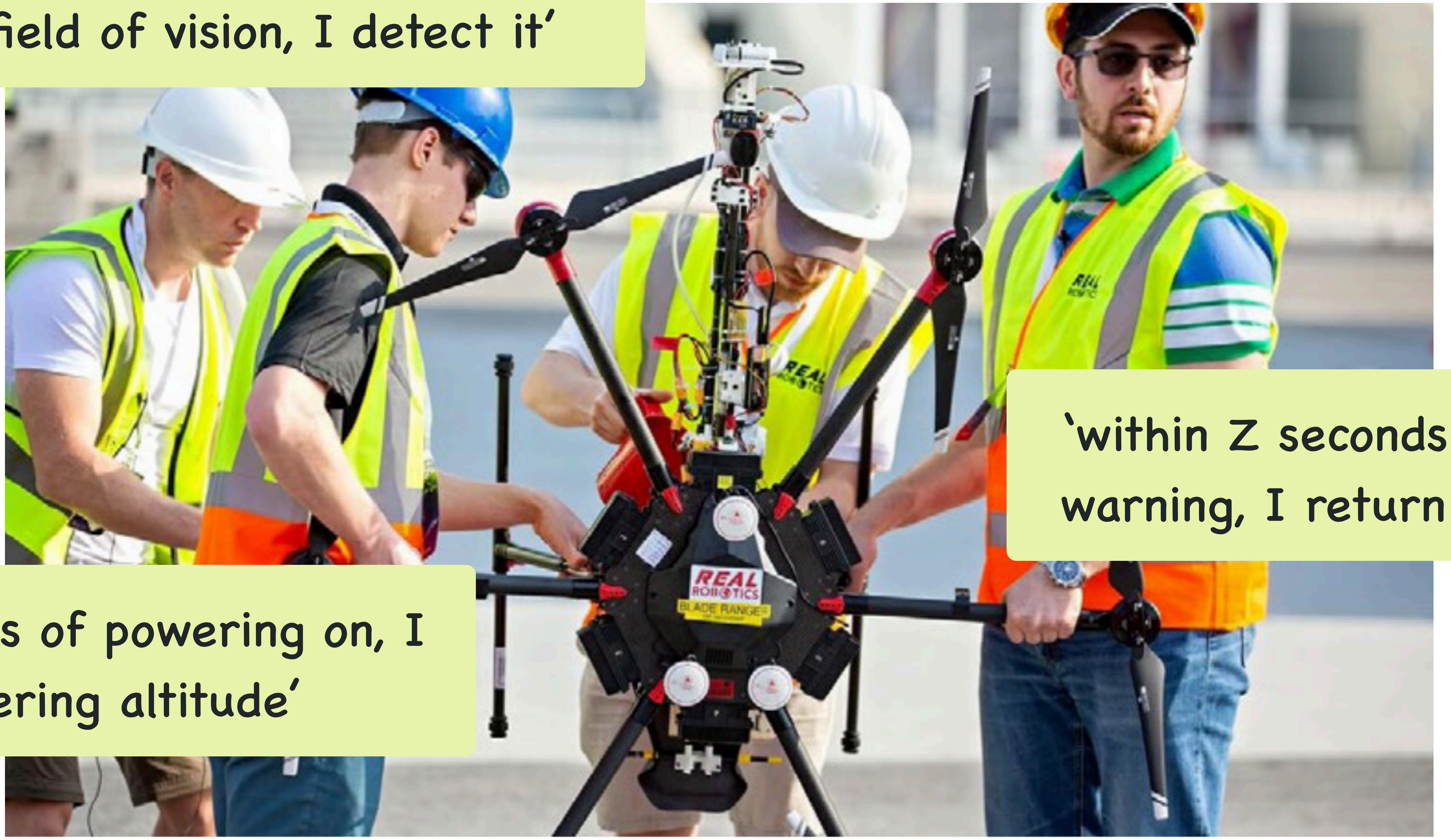
robot software verification

=

concurrency verification



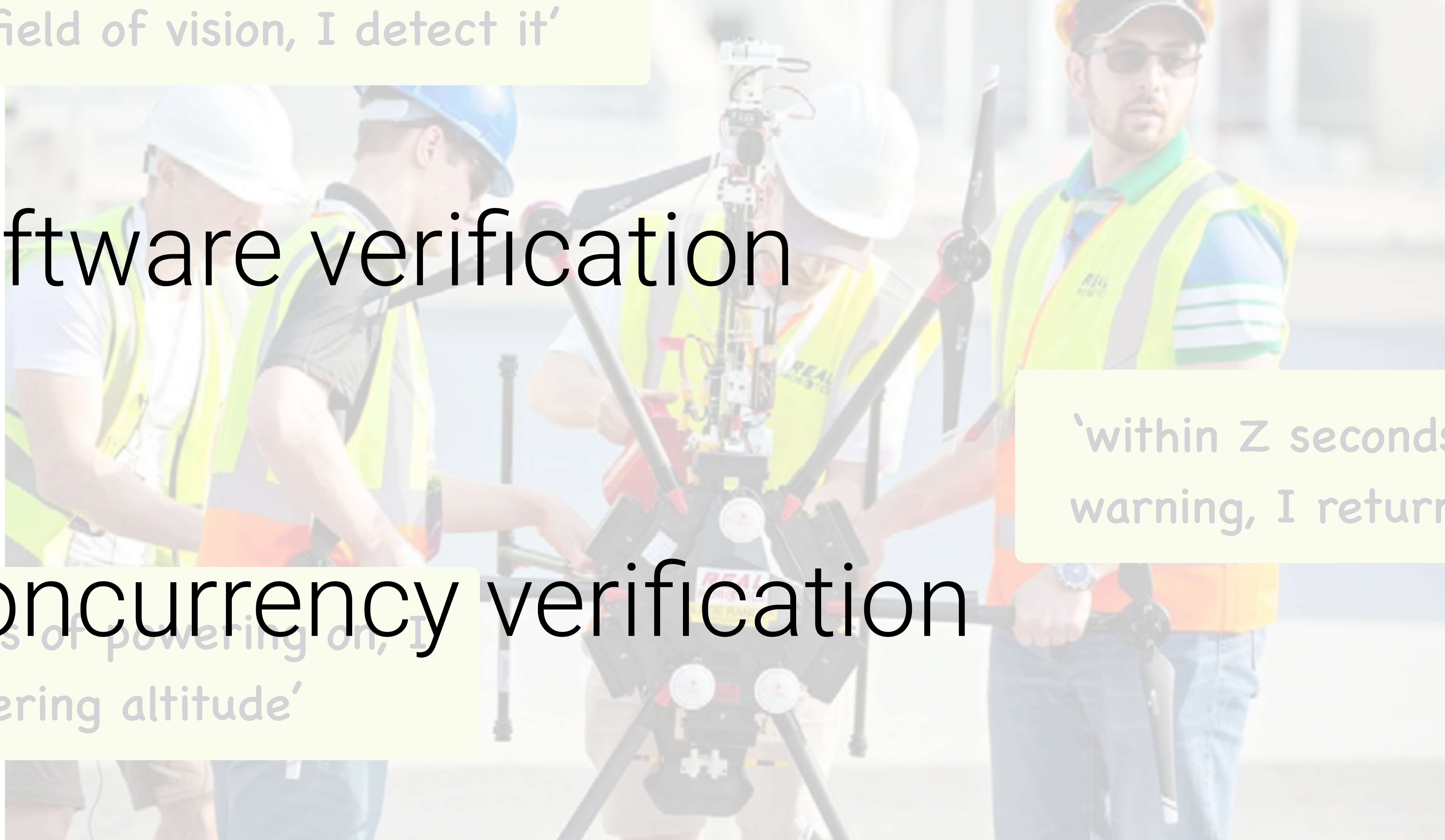
‘within X seconds of a fire appearing in my field of vision, I detect it’



‘within Z seconds of a battery low warning, I return to home position’

‘within Y seconds of powering on, I am at hovering altitude’

‘within X seconds of a fire appearing in my field of vision, I detect it’



robot software verification

=

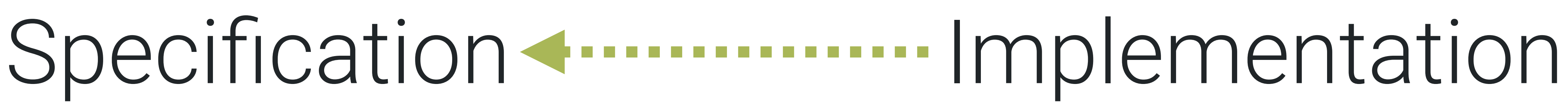
timed concurrency verification

‘within Z seconds of a battery low warning, I return to home position’

‘within Y seconds of powering on, I am at hovering altitude’

Verification approach

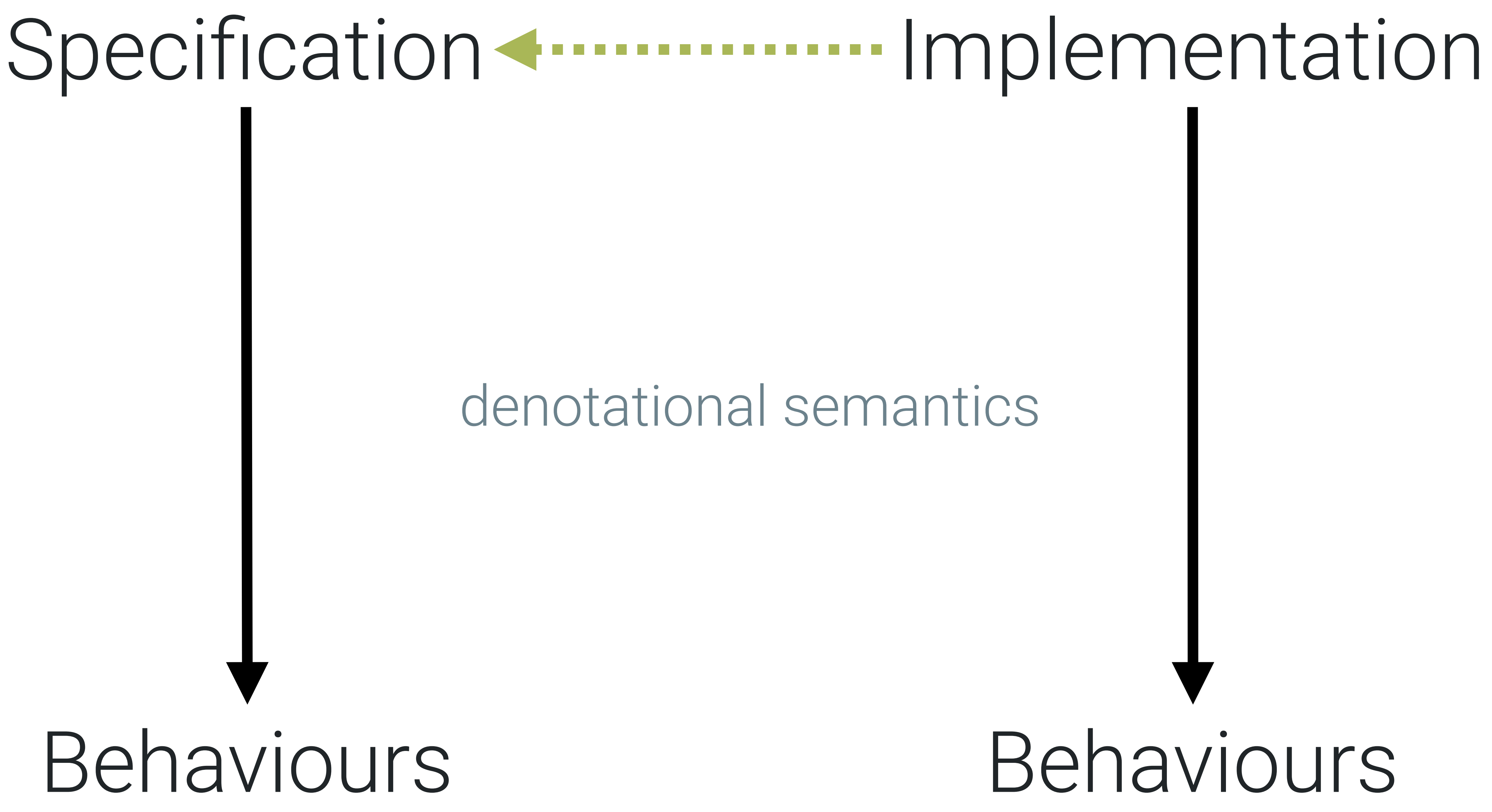
Refinement



Specification

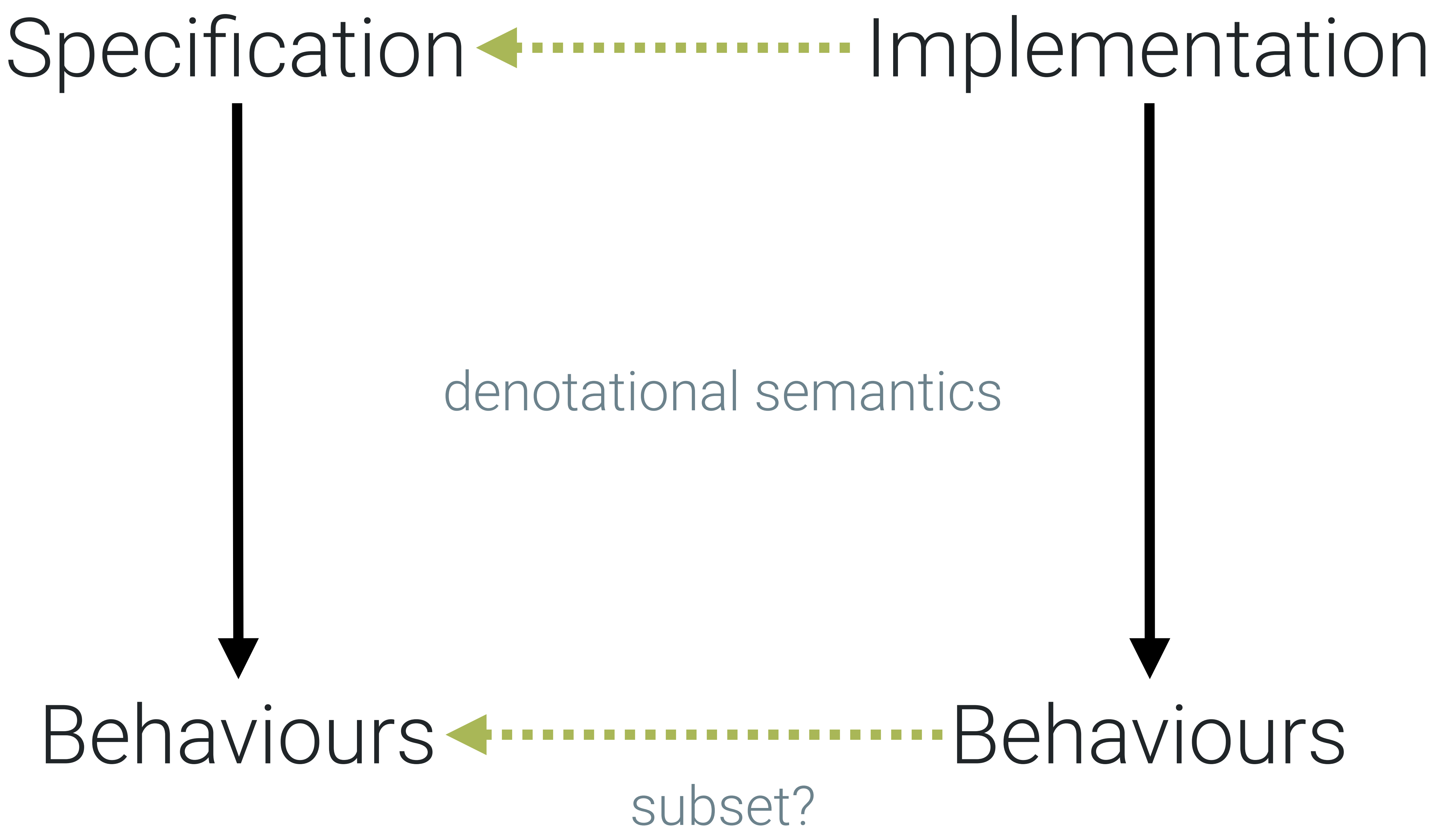
Implementation

Verification approach
Refinement



Verification approach

Refinement



Verification approach

Refinement: transitivity

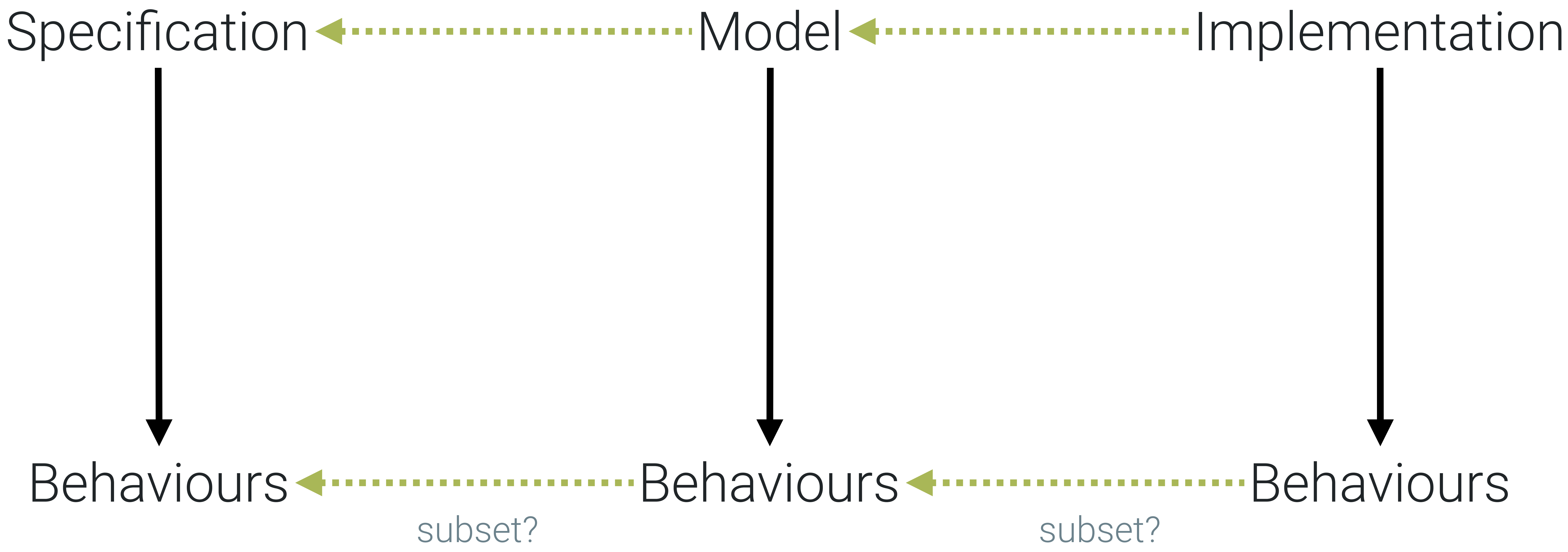
Specification

Behaviours

Implementation

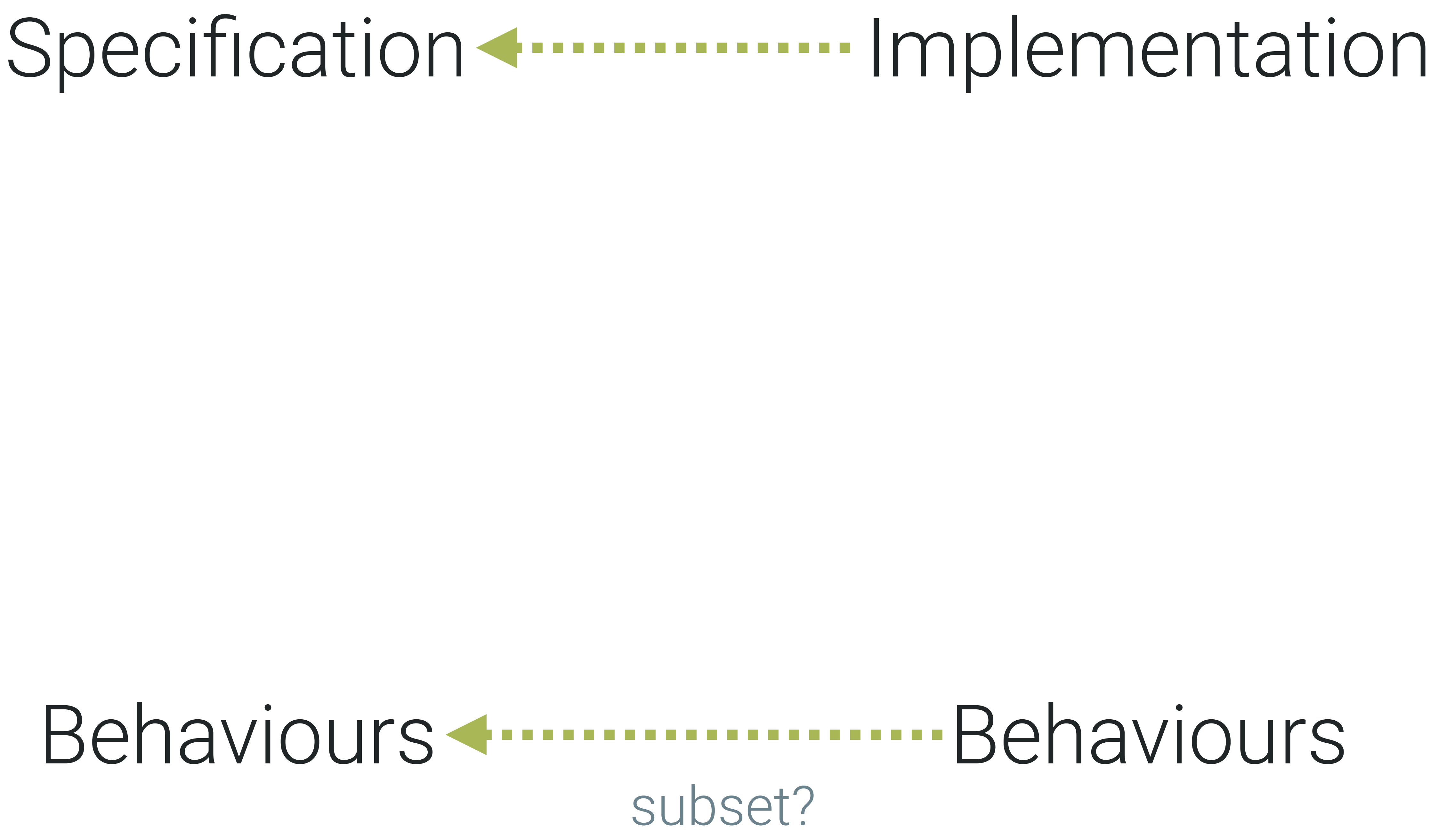
Behaviours

Verification approach
Refinement: transitivity



Verification approach

Refinement: transitivity, again



Specification

Implementation

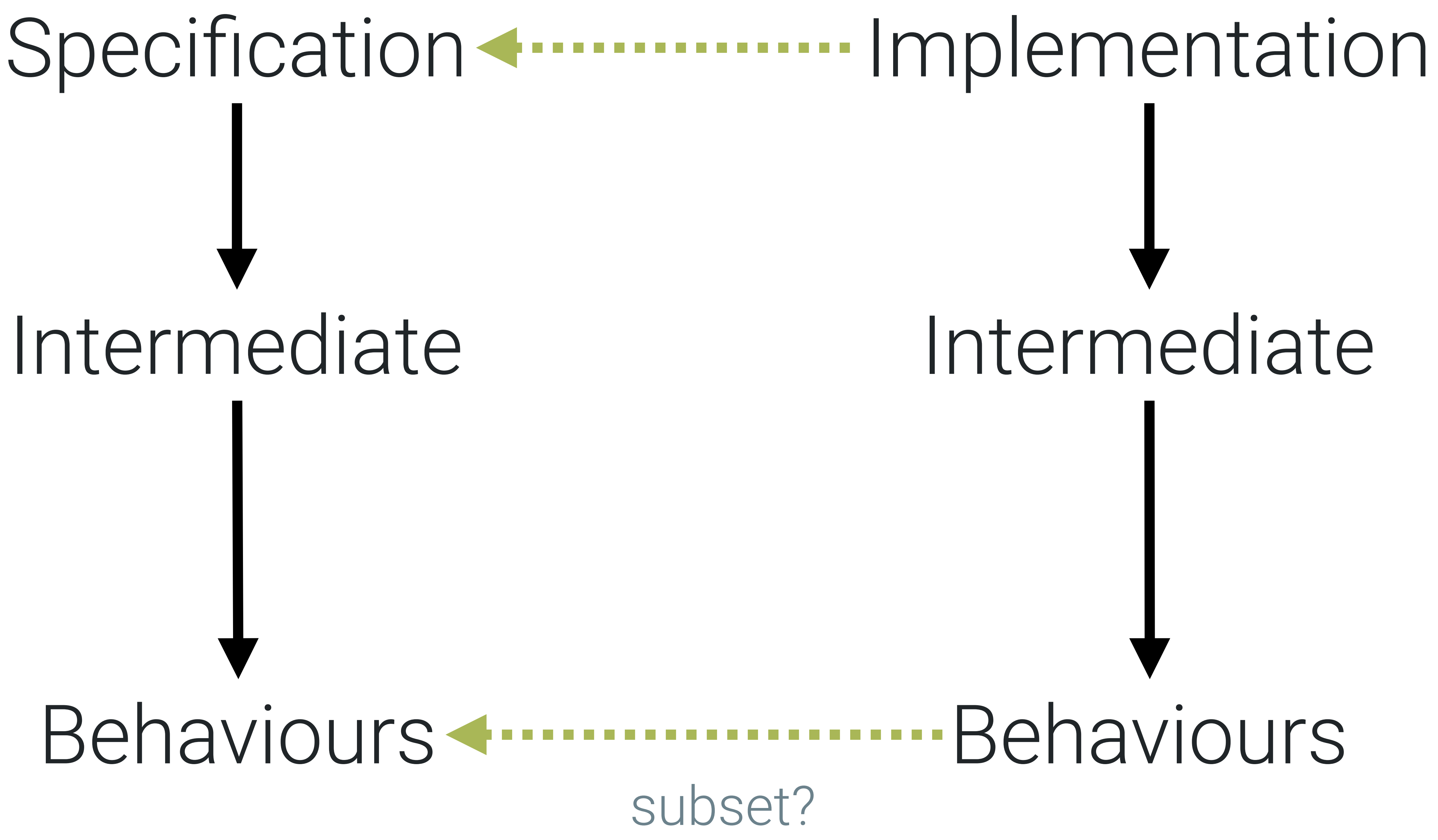
Behaviours

Behaviours

subset?

Verification approach

Refinement: transitivity, again



Specification

Implementation

Intermediate

Intermediate

Behaviours

Behaviours

subset?

CSP

Process algebra for refinement

STOP	deadlock	P [] Q	external choice	P;Q	sequential composition
SKIP	termination	P|~|Q	internal choice	μ x. P(x)	recursion
a → P	event	P|||Q	interleaving	P∧Q	interrupt
		P⟦A⟧Q	alphabetised parallel	…and many others	

CSP
Refinement

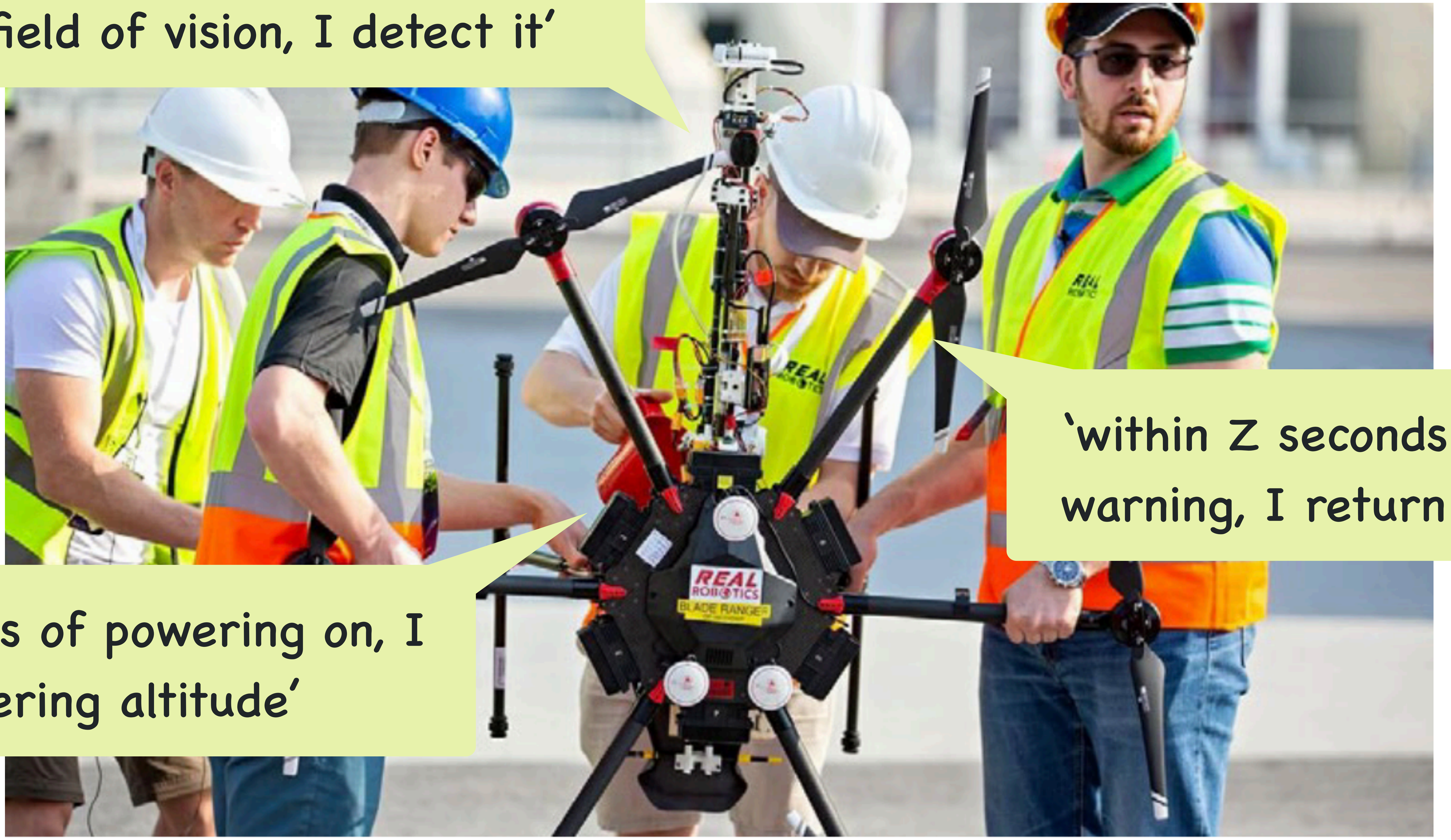
$$\textbf{Spec} \sqsubseteq_{M} \textbf{Impl}$$

all behaviours of process **Impl**

according to *semantic model* **M**

are behaviours of process **Spec**

according to **M**



‘within X seconds of a fire appearing in my field of vision, I detect it’

‘within Z seconds of a battery low warning, I return to home position’

‘within Y seconds of powering on, I am at hovering altitude’

tock-CSP
Discrete-time process algebra

tock

event representing
passage of one time unit

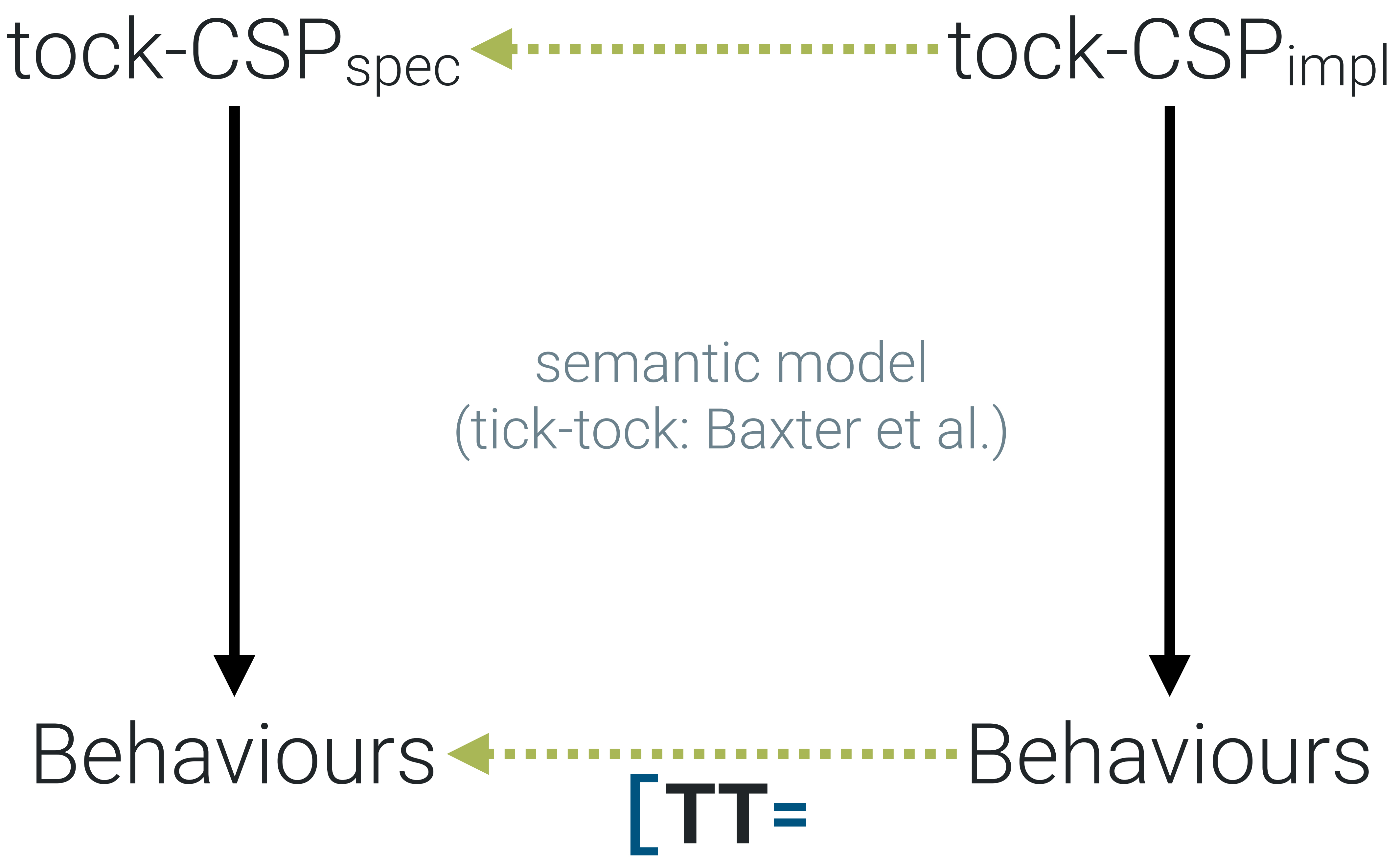
tock-CSP
Developments: FDR timed sections

`[]`, `/\` etc. **lifted** to accommodate **`tock`**

distinction between **`STOP`** (time may progress) and **`USTOP`** (timelock)

tock-CSP

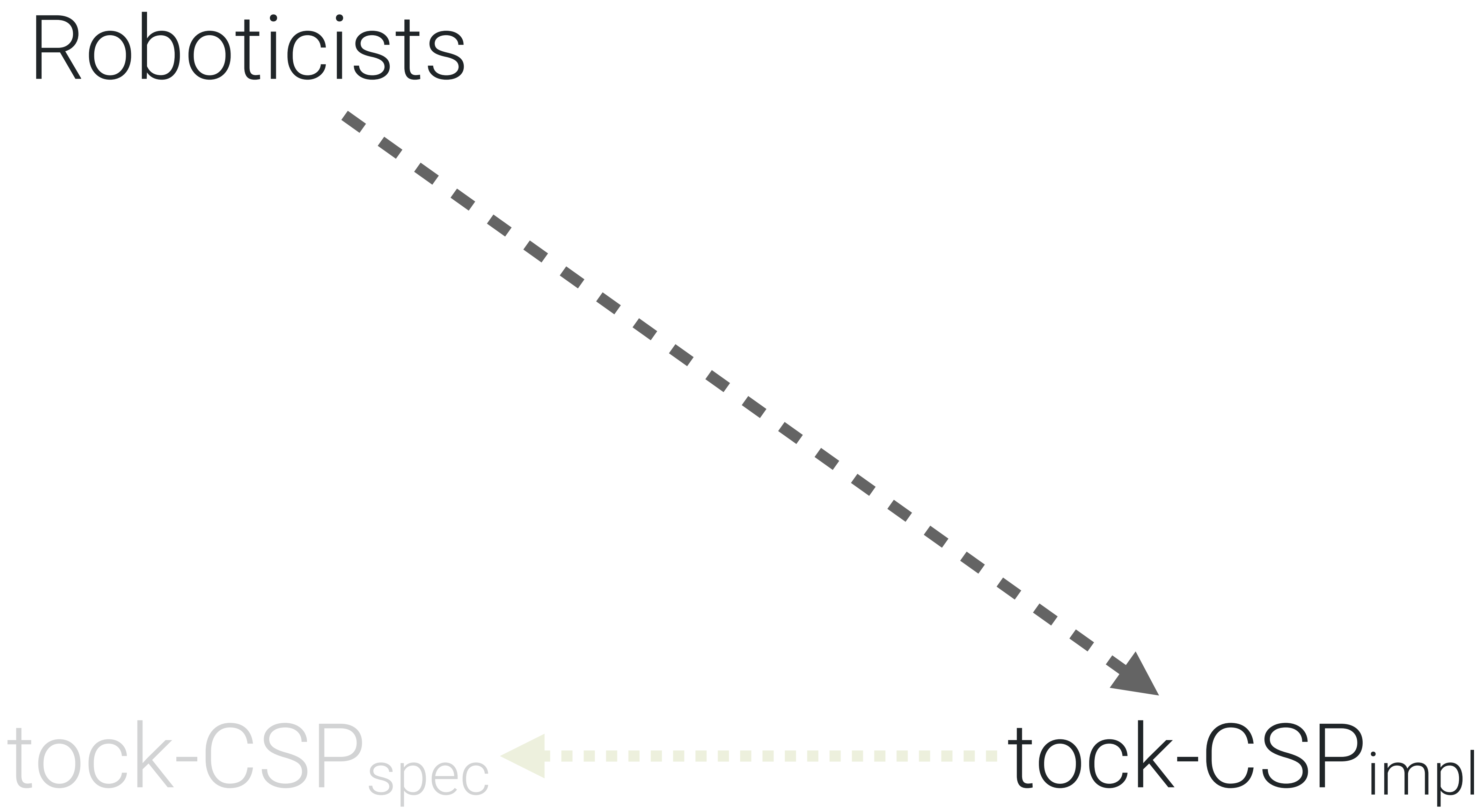
Developments: tick-tock semantic model



model-shifting into traces model for FDR etc.

tock-CSP$_{\text{spec}}$

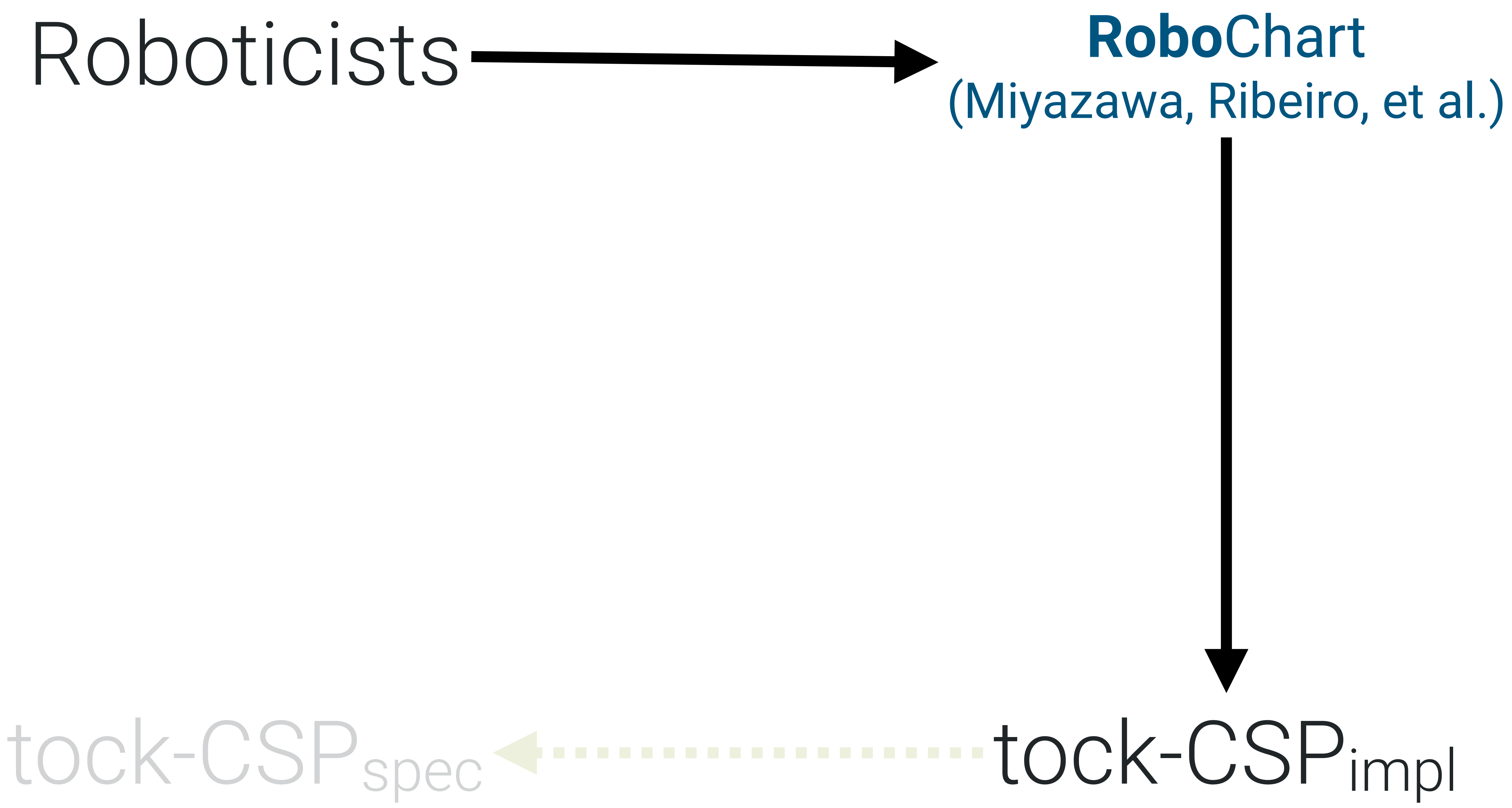
tock-CSP$_{\text{impl}}$



Roboticists

tock-CSP$_{\text{spec}}$

tock-CSP$_{\text{impl}}$



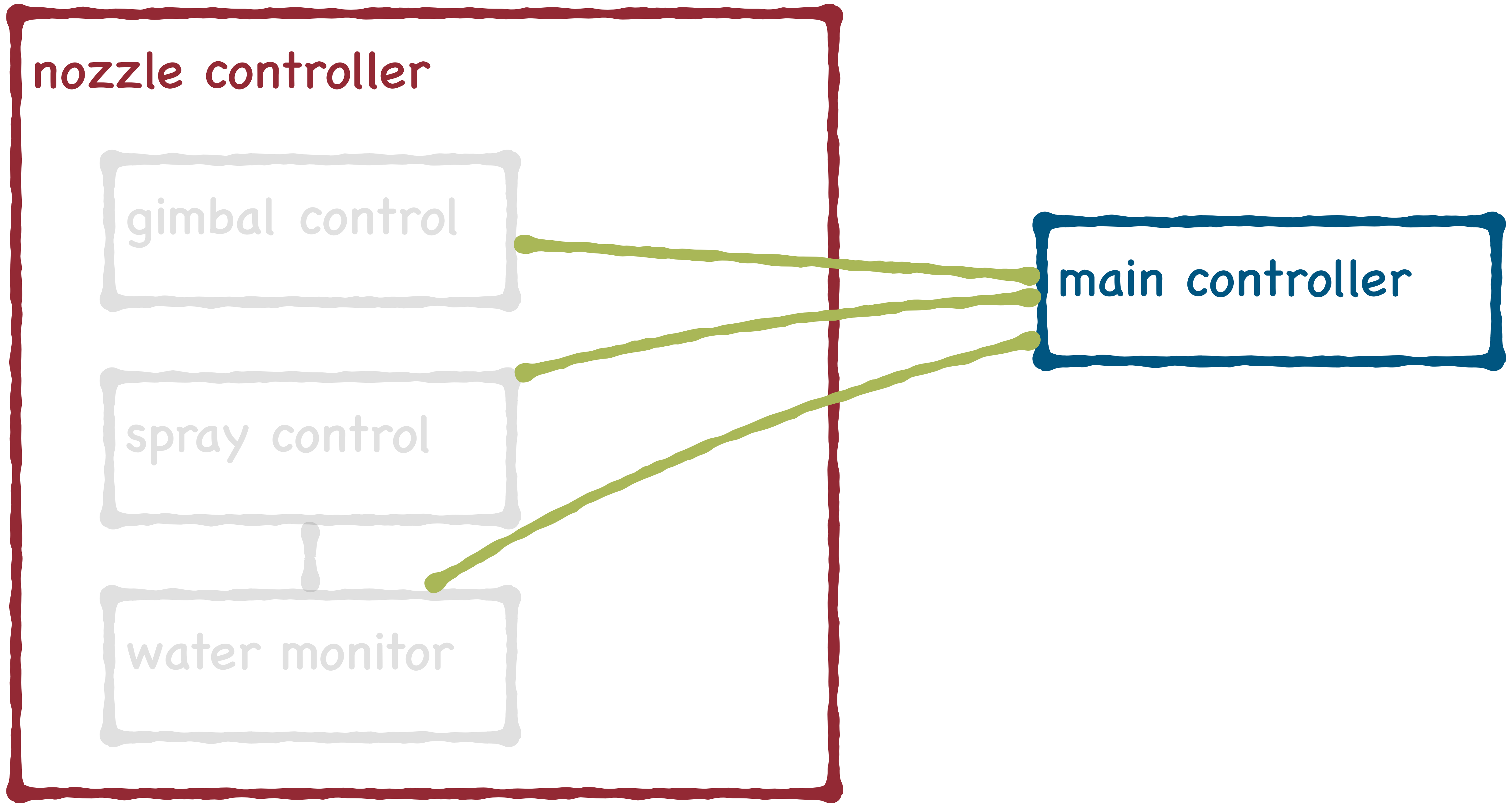
Roboticists

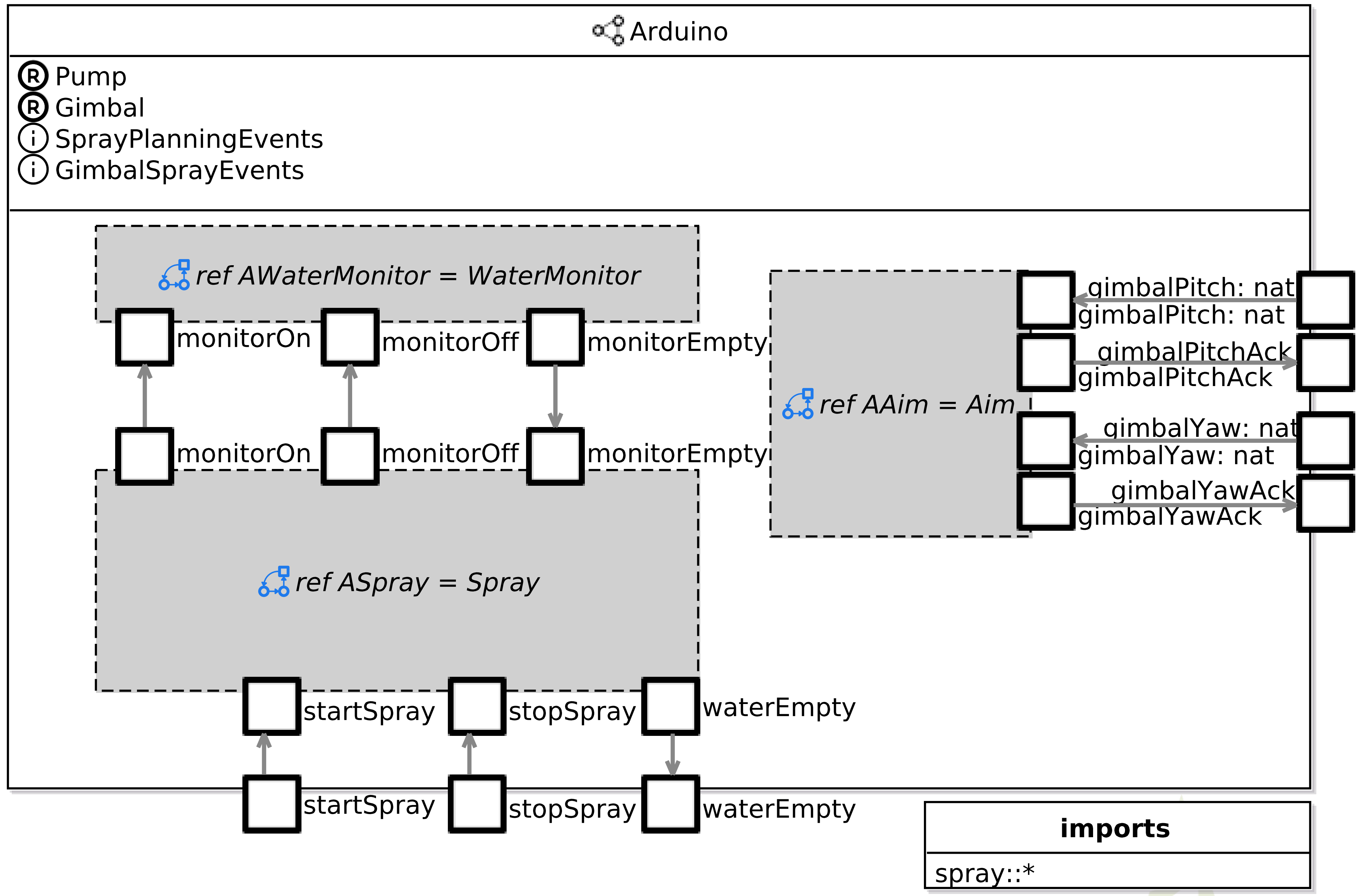
RoboChart
(Miyazawa, Ribeiro, et al.)

tock-CSP$_{\text{impl}}$

tock-CSP$_{\text{spec}}$

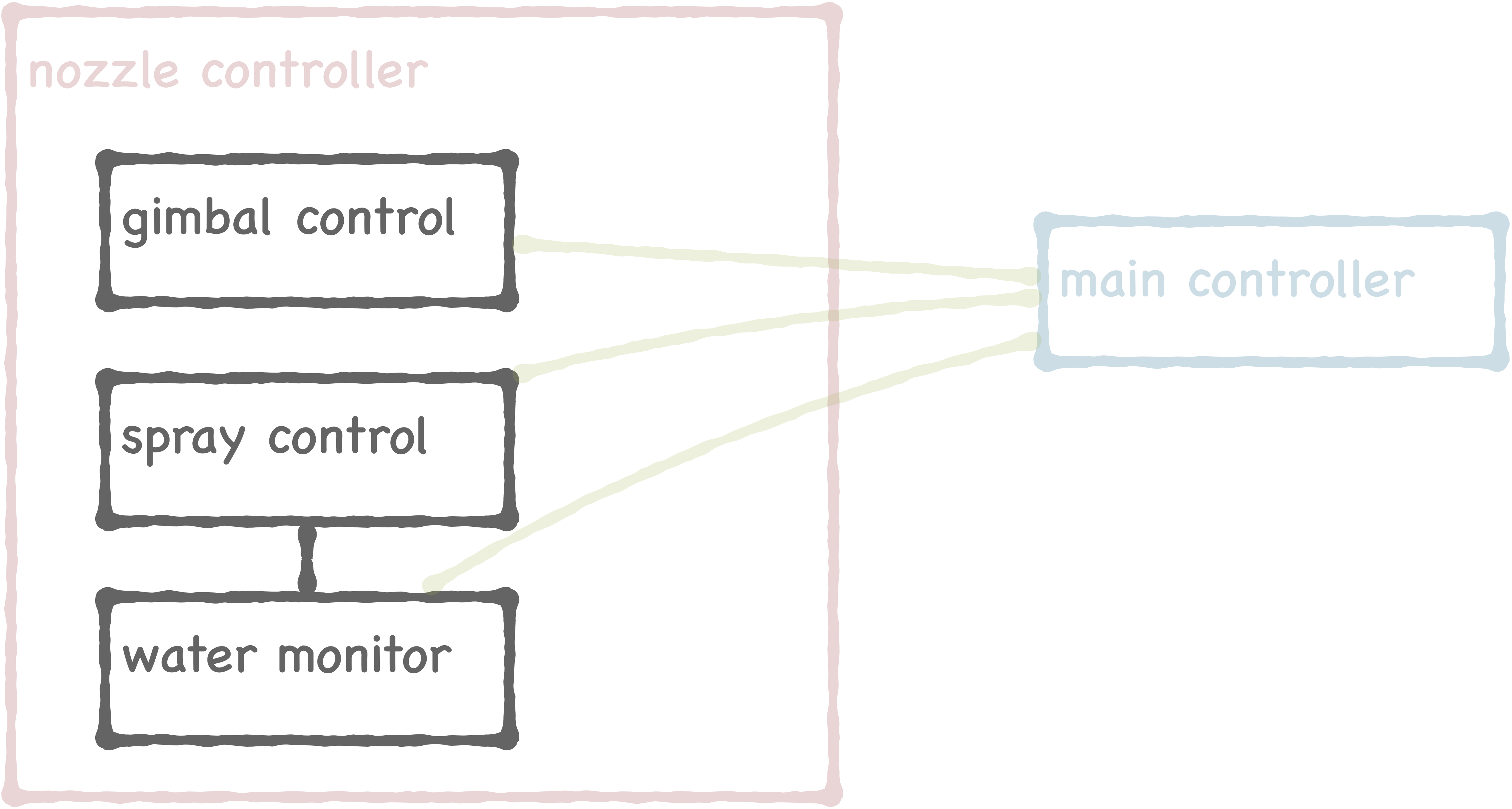
RoboChart
Graphical notation for software design models

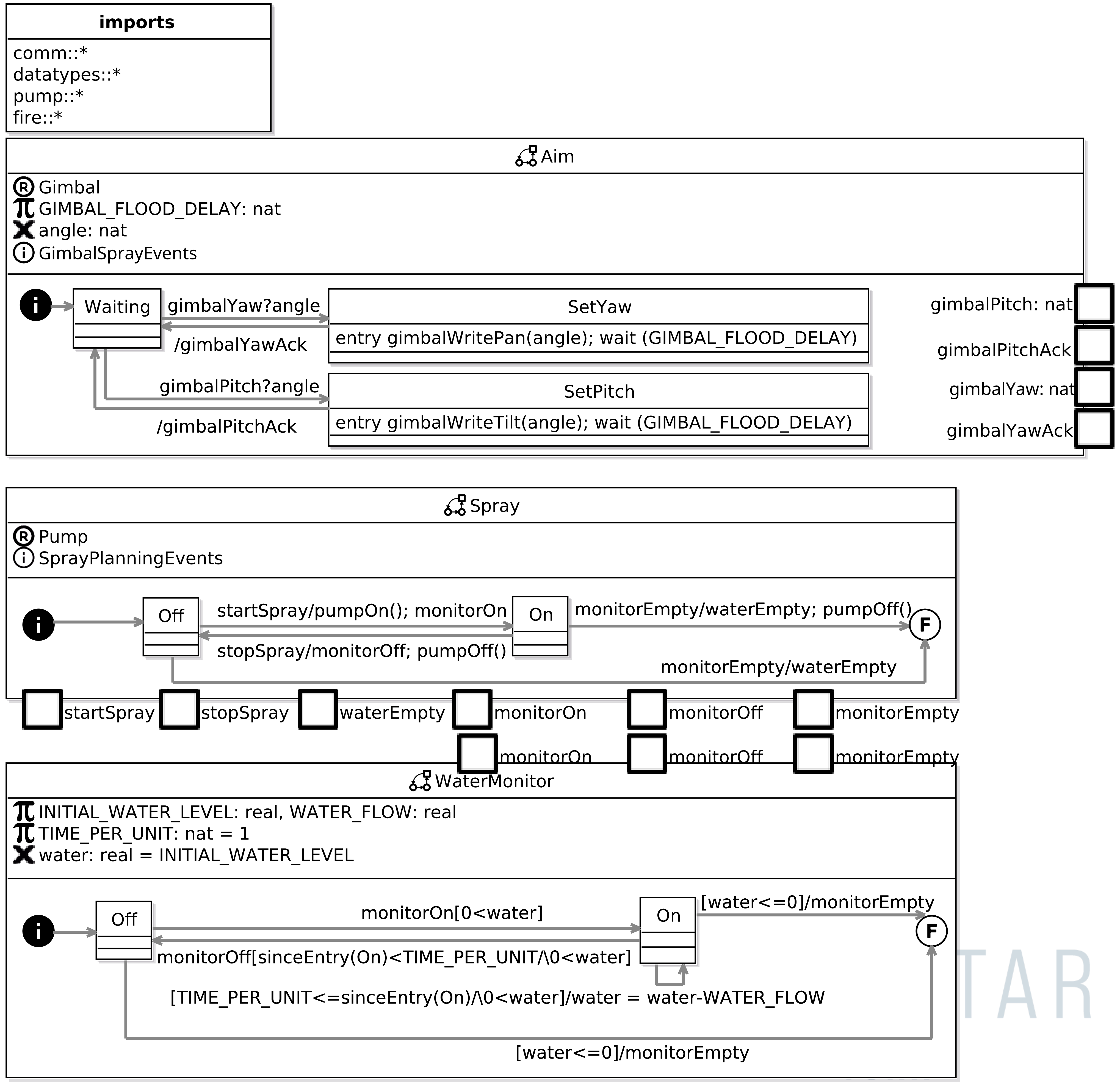




RoboChart

Graphical notation for software design models





imports

comm::*
datatypes::*
pump::*
fire::*

Aim

Ⓡ Gimbal
π GIMBAL_FLOOD_DELAY: nat
angle: nat
ⓘ GimbalSprayEvents

Waiting

gimbalYaw?angle

/gimbalYawAck

SetYaw

entry gimbalWritePan(angle); wait (GIMBAL_FLOOD_DELAY)

gimbalPitch?angle

/gimbalPitchAck

SetPitch

entry gimbalWriteTilt(angle); wait (GIMBAL_FLOOD_DELAY)

gimbalPitch: nat
gimbalPitchAck
gimbalYaw: nat
gimbalYawAck

Spray

Ⓡ Pump
ⓘ SprayPlanningEvents

Off

startSpray/pumpOn(); monitorOn

On

stopSpray/monitorOff; pumpOff()

monitorEmpty/waterEmpty; pumpOff()

monitorEmpty/waterEmpty

F

startSpray
stopSpray
waterEmpty
monitorOn
monitorOff
monitorEmpty

monitorOn
monitorOff
monitorEmpty

WaterMonitor

π INITIAL_WATER_LEVEL: real, WATER_FLOW: real
π TIME_PER_UNIT: nat = 1
water: real = INITIAL_WATER_LEVEL

Off

monitorOn[0<water]

On

monitorOff[sinceEntry(On)<TIME_PER_UNIT/\0<water]

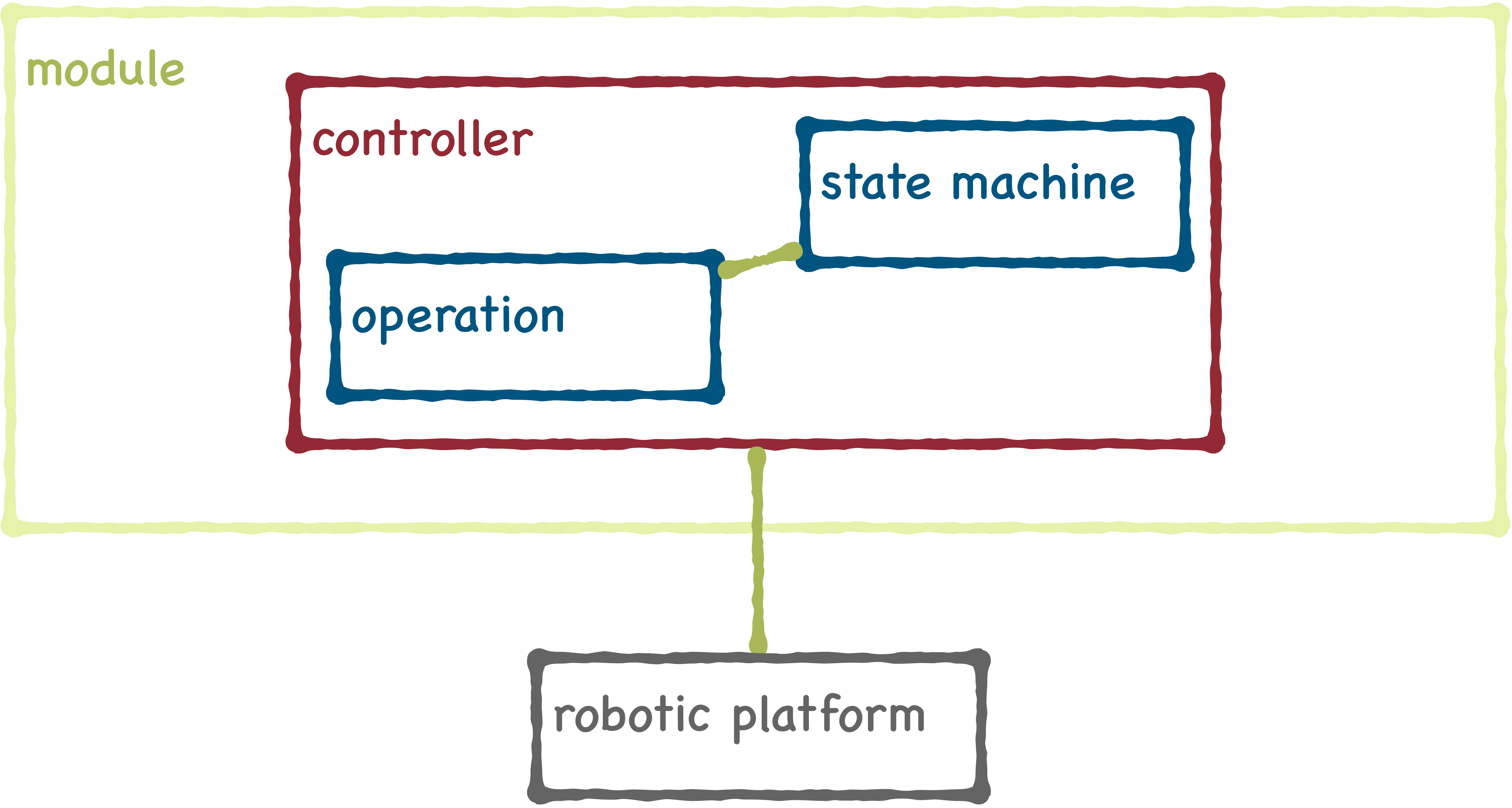
[water<=0]/monitorEmpty

[TIME_PER_UNIT<=sinceEntry(On)/\0<water]/water = water-WATER_FLOW

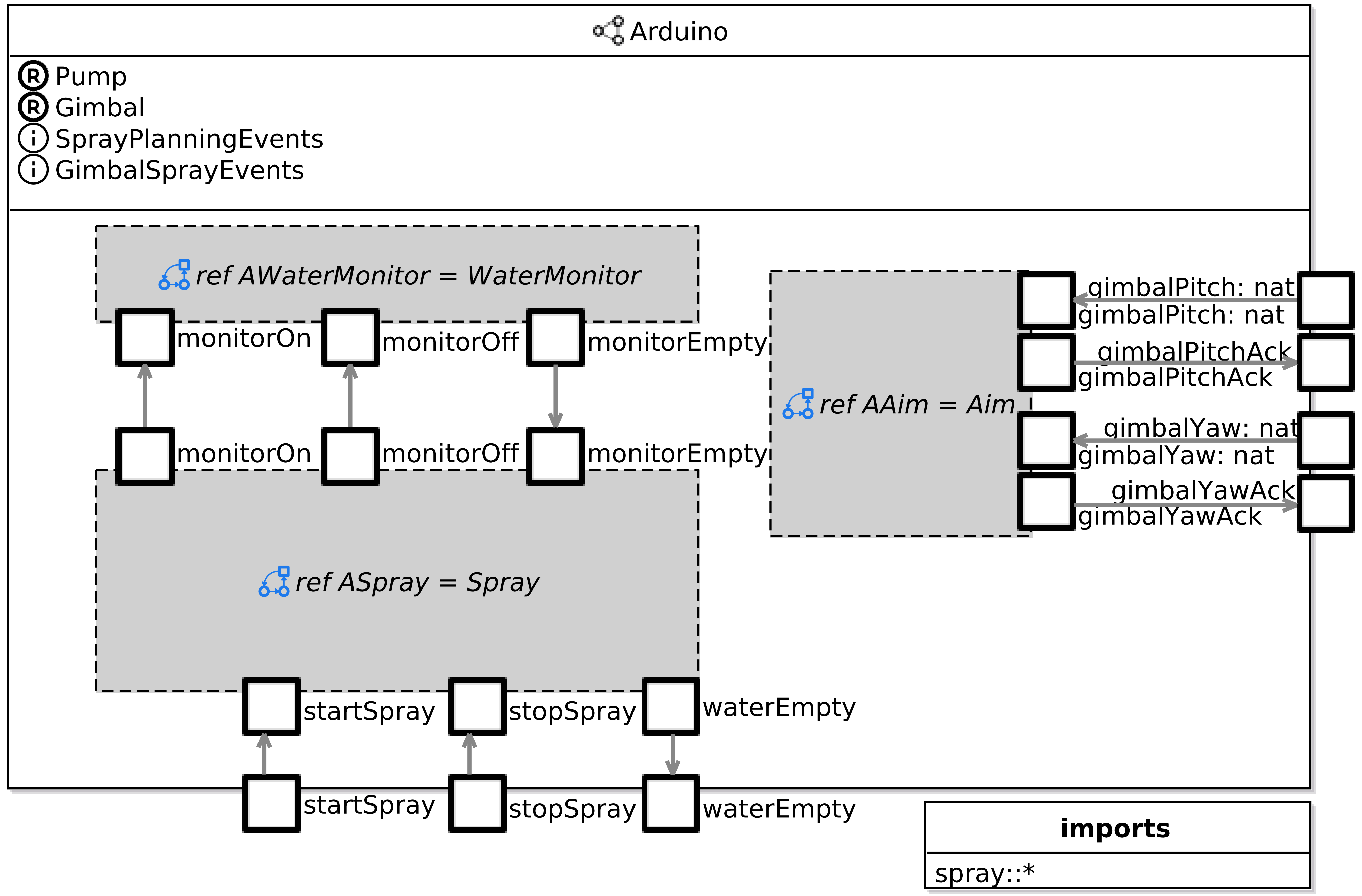
[water<=0]/monitorEmpty

F

RoboChart
Component model



RoboChart
Translation of controller to CSP

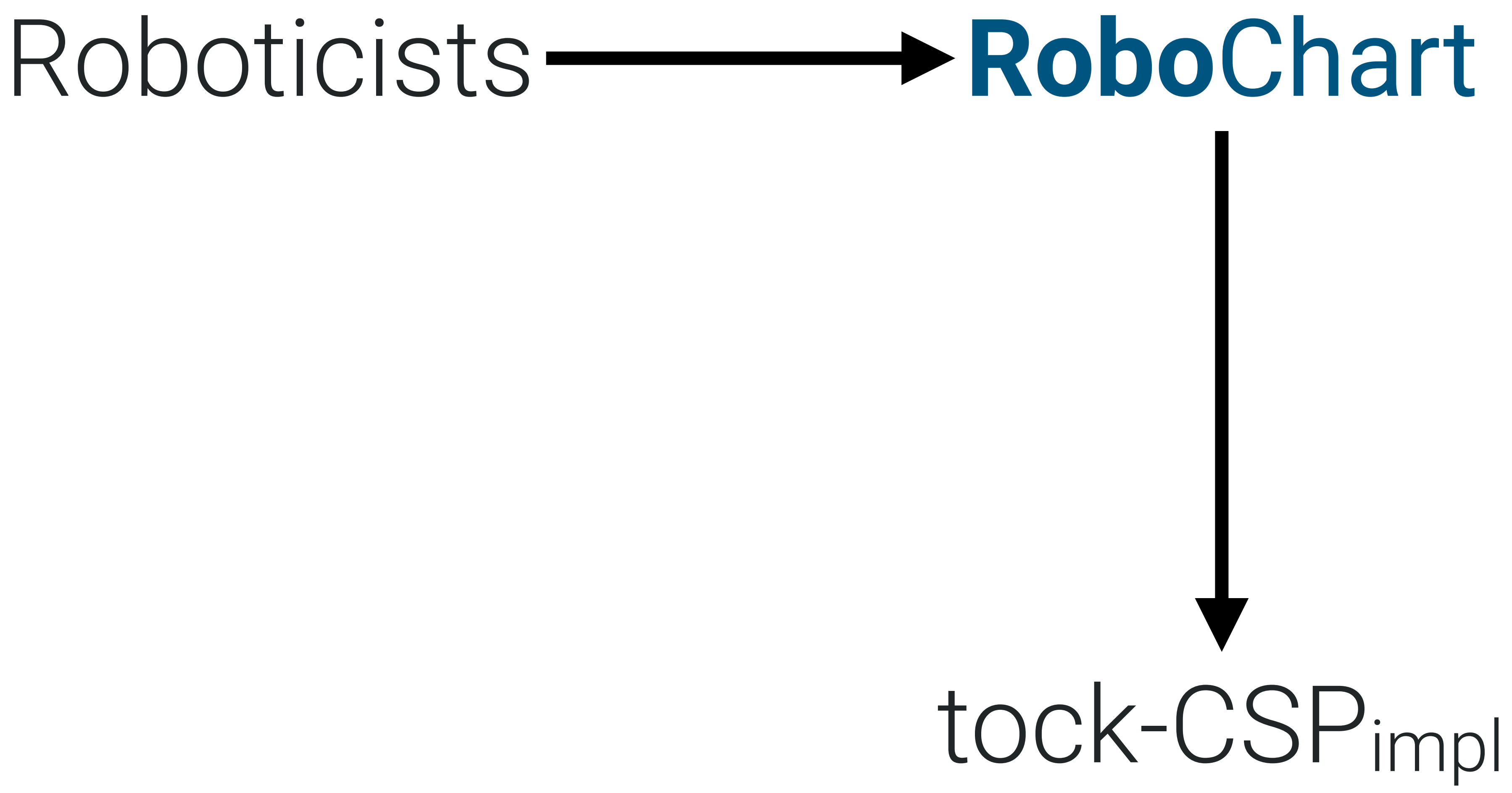


State machines become **processes**

parallelised by shared events

eg

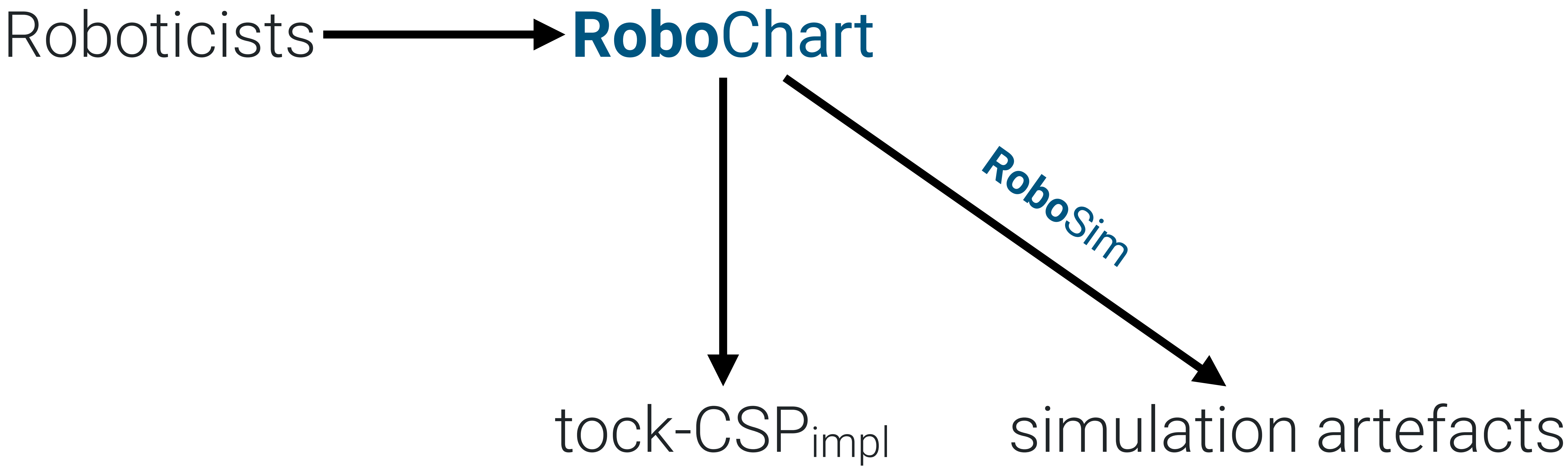
```
AWaterMonitor
⟦ { monitorOn, monitorOff, monitorEmpty } ⟧
( ASpray ||| AAim )
```



Roboticists

RoboChart

tock-CSP$_{impl}$



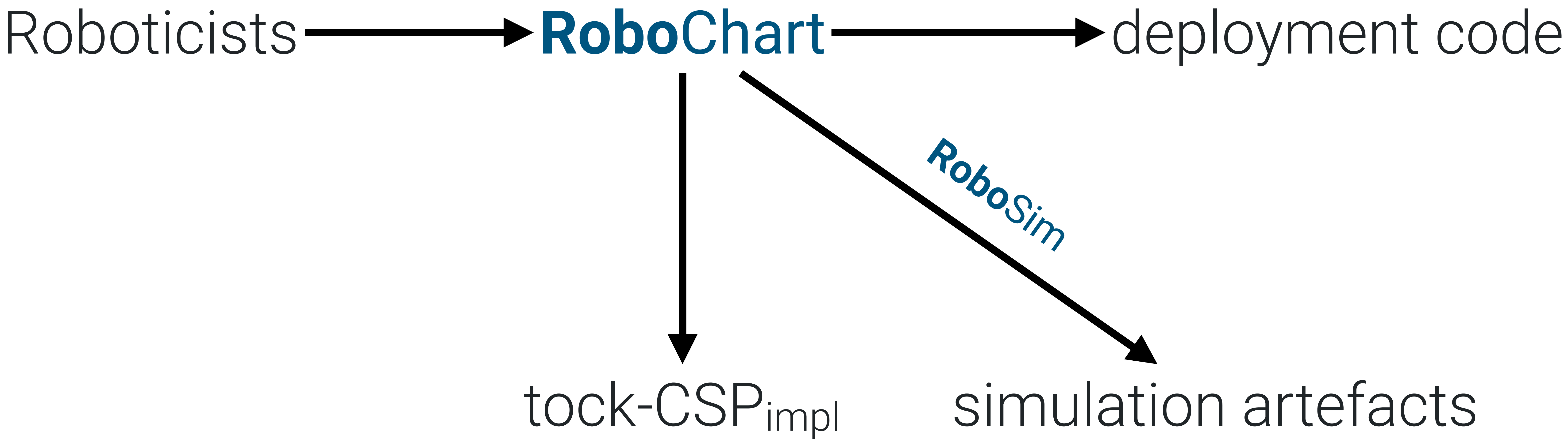
Roboticists

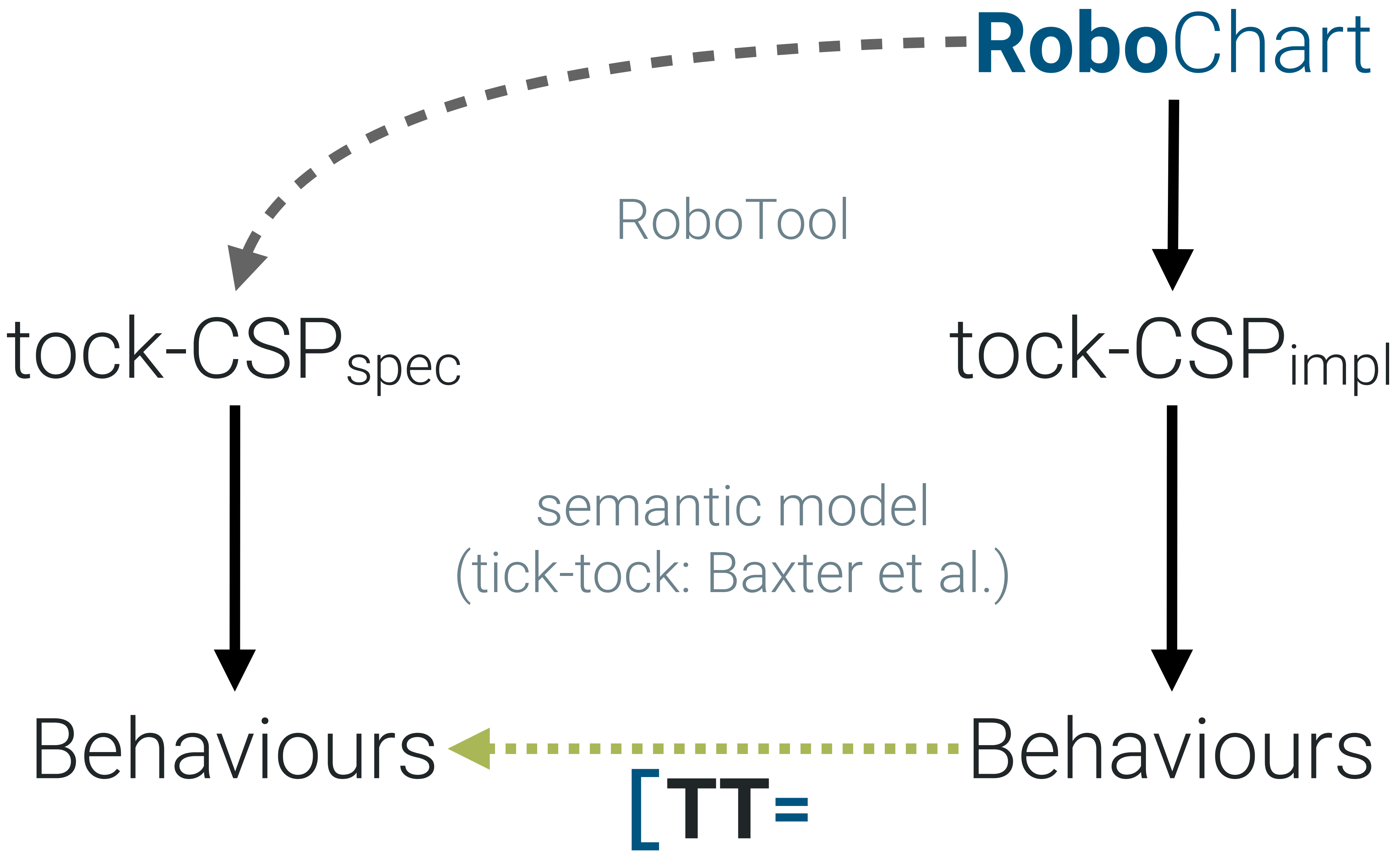
RoboChart

RoboSim

tock-CSP$_{\text{impl}}$

simulation artefacts


Roboticists
RoboChart
deployment code
RoboSim
tock-CSP$_{impl}$
simulation artefacts



RoboChart

RoboTool

tock-CSP$_{spec}$

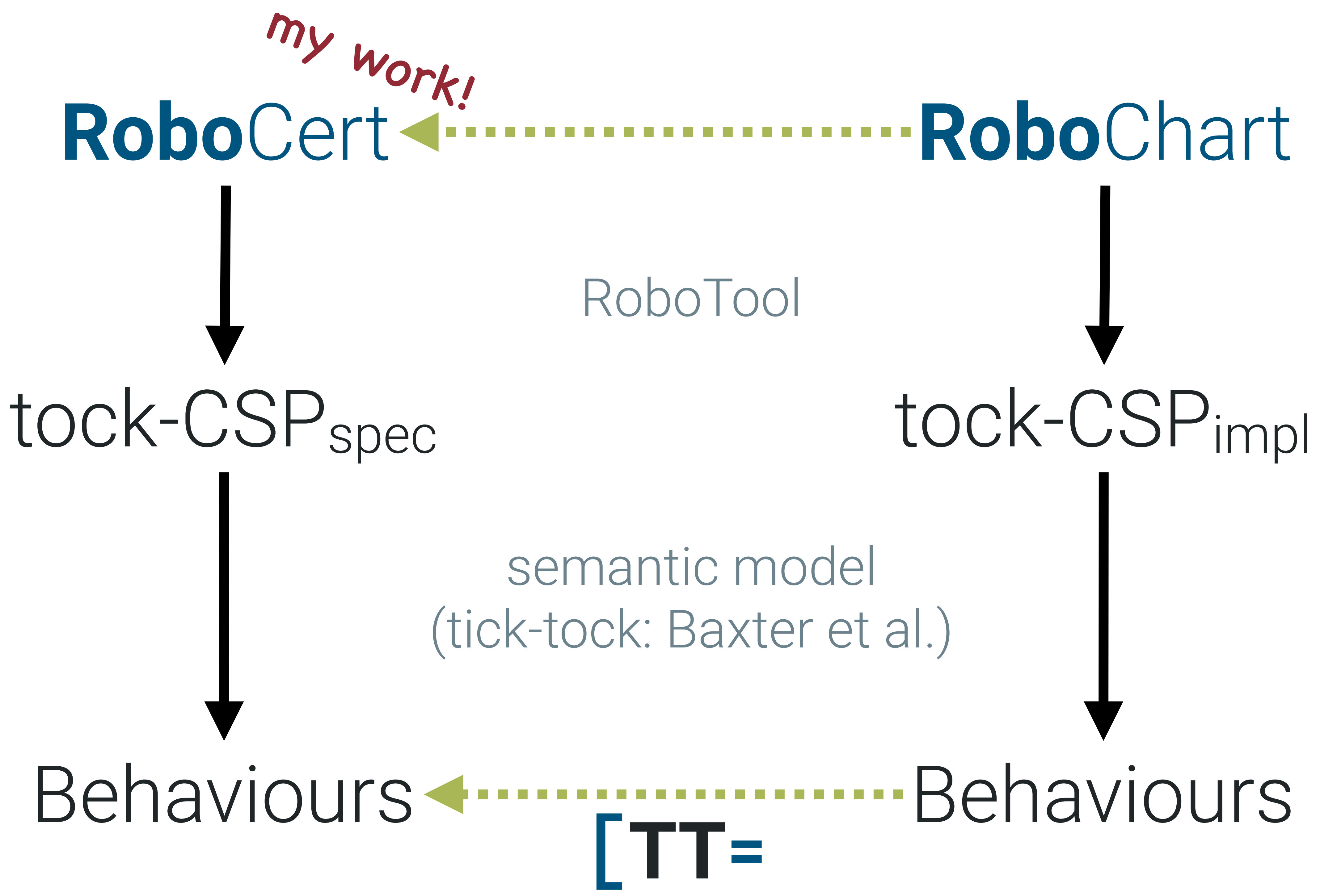
tock-CSP$_{impl}$

semantic model
(tick-tock: Baxter et al.)

Behaviours

Behaviours

[TT=



my work!

RoboCert ◀······ **Robo**Chart

RoboTool

tock-CSP$_{spec}$

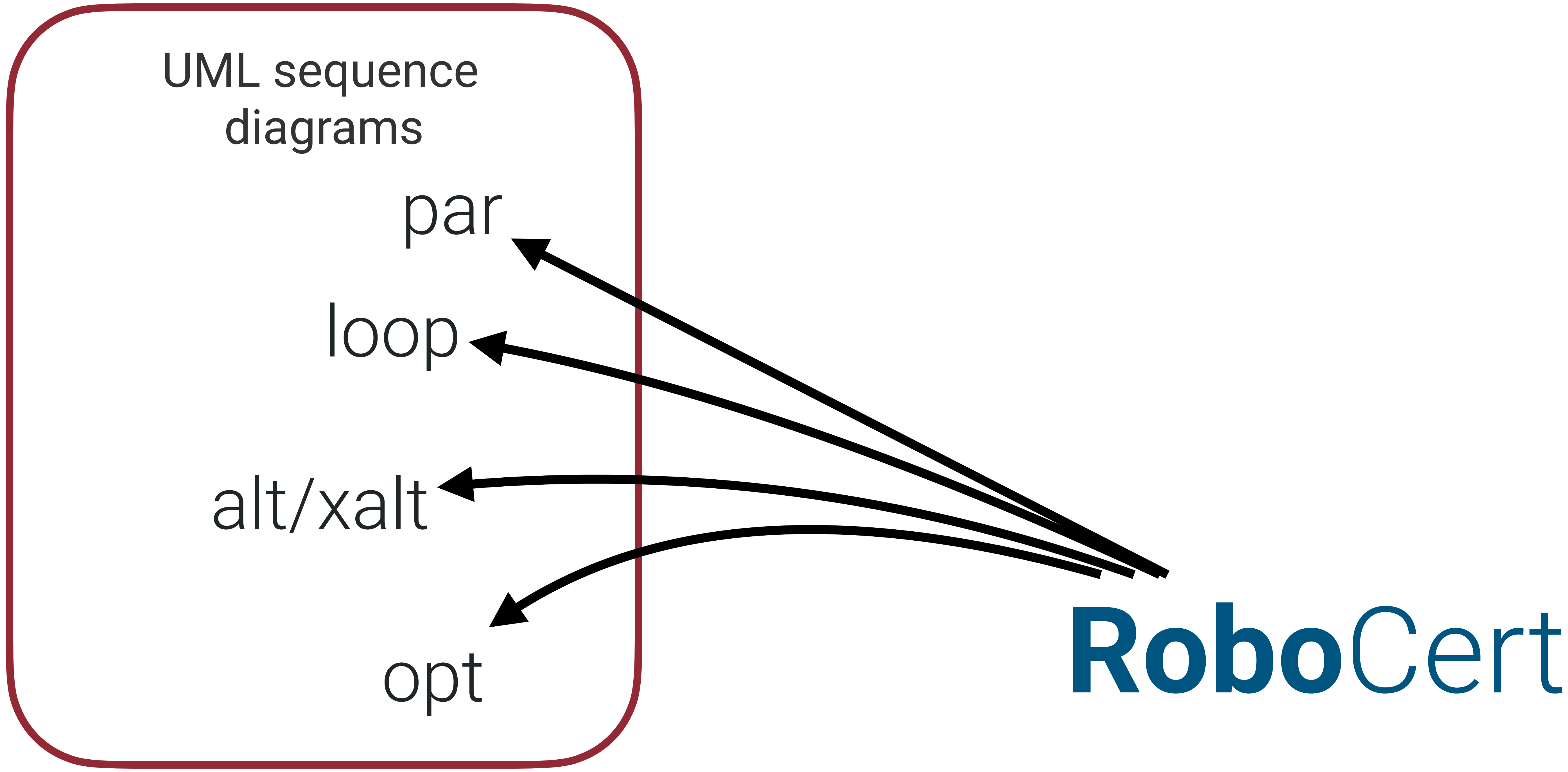
tock-CSP$_{impl}$

semantic model
(tick-tock: Baxter et al.)

Behaviours ◀······ Behaviours

$[TT=$

RoboCert



UML sequence diagrams

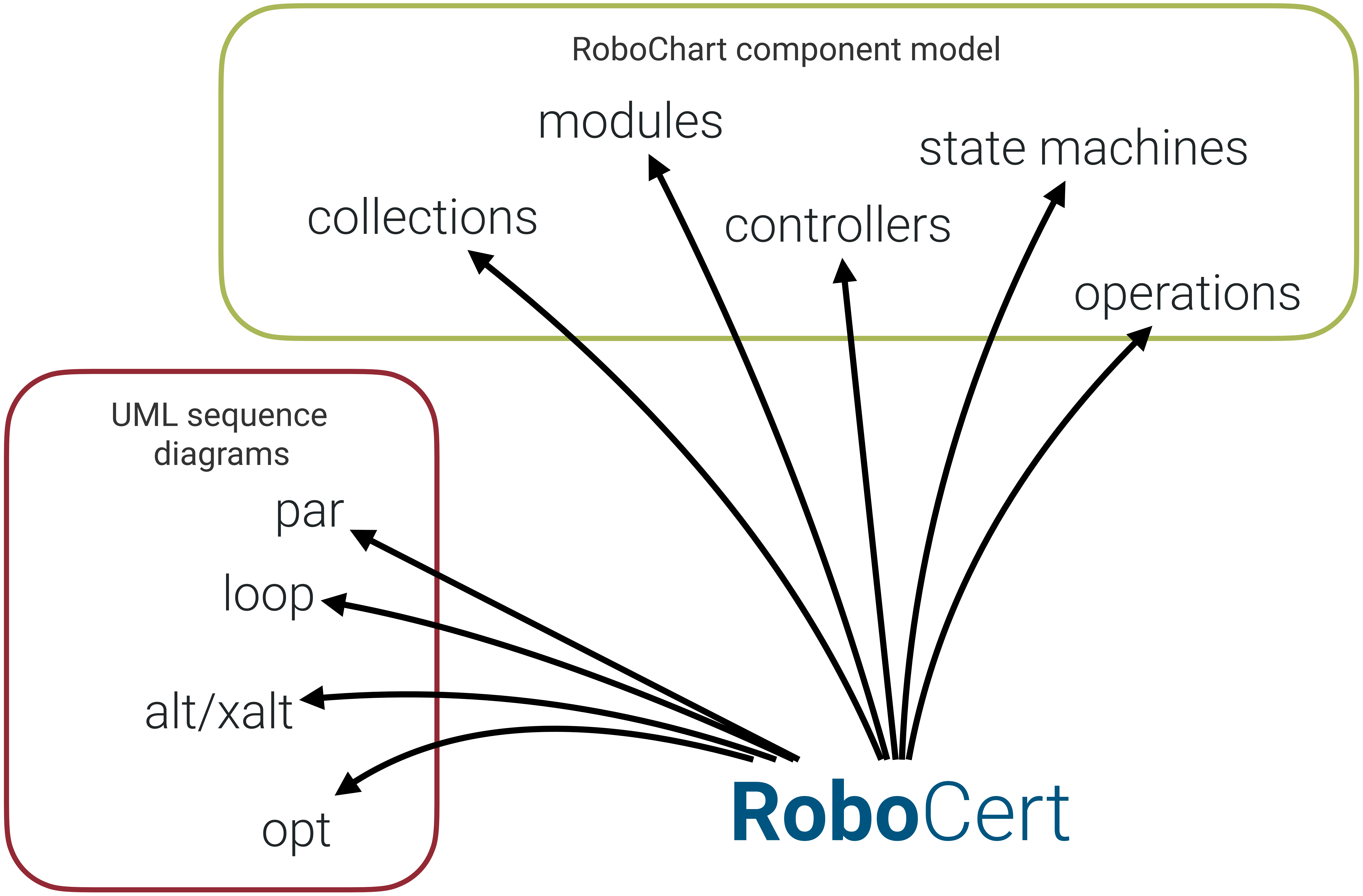
par

loop

alt/xalt

opt

RoboCert



RoboChart component model

modules

state machines

collections

controllers

operations

UML sequence diagrams

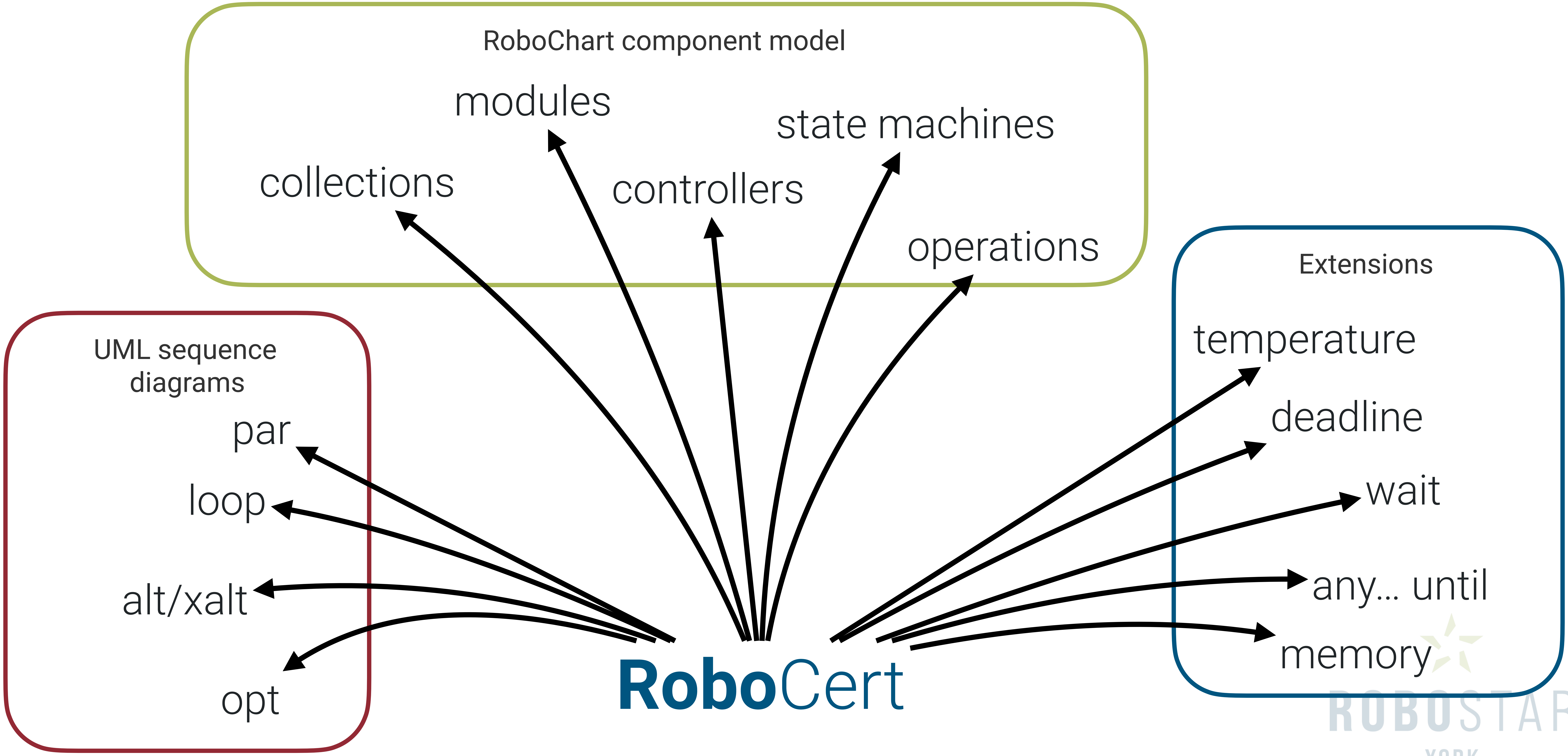
par

loop

alt/xalt

opt

RoboCert



RoboChart component model

modules

state machines

collections

controllers

operations

Extensions

temperature

deadline

wait

any… until

memory

UML sequence diagrams

par

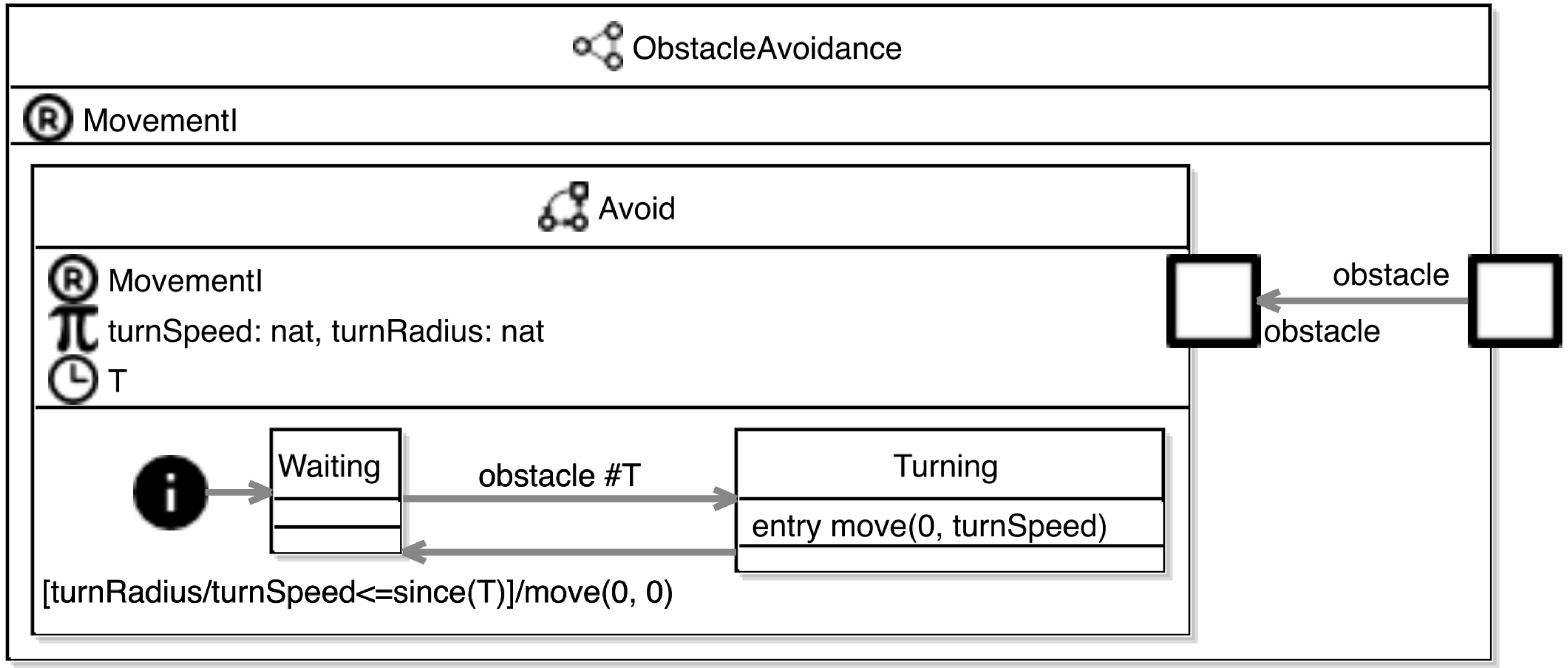
loop

alt/xalt

opt

RoboCert

RoboCert
Another motivating example



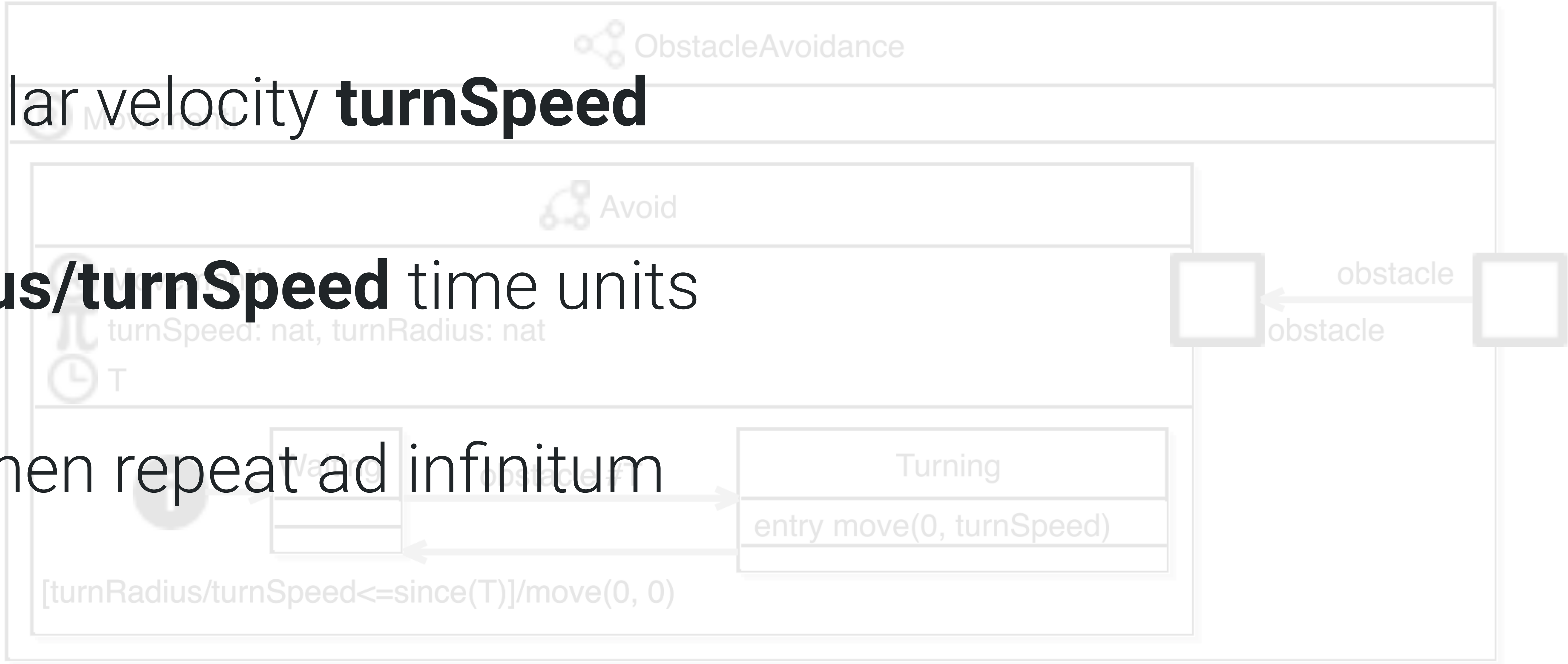
RoboCert
Another motivating example

wait for warning of obstacle

turn with angular velocity **turnSpeed**

wait **turnRadius/turnSpeed** time units

stop turning, then repeat ad infinitum



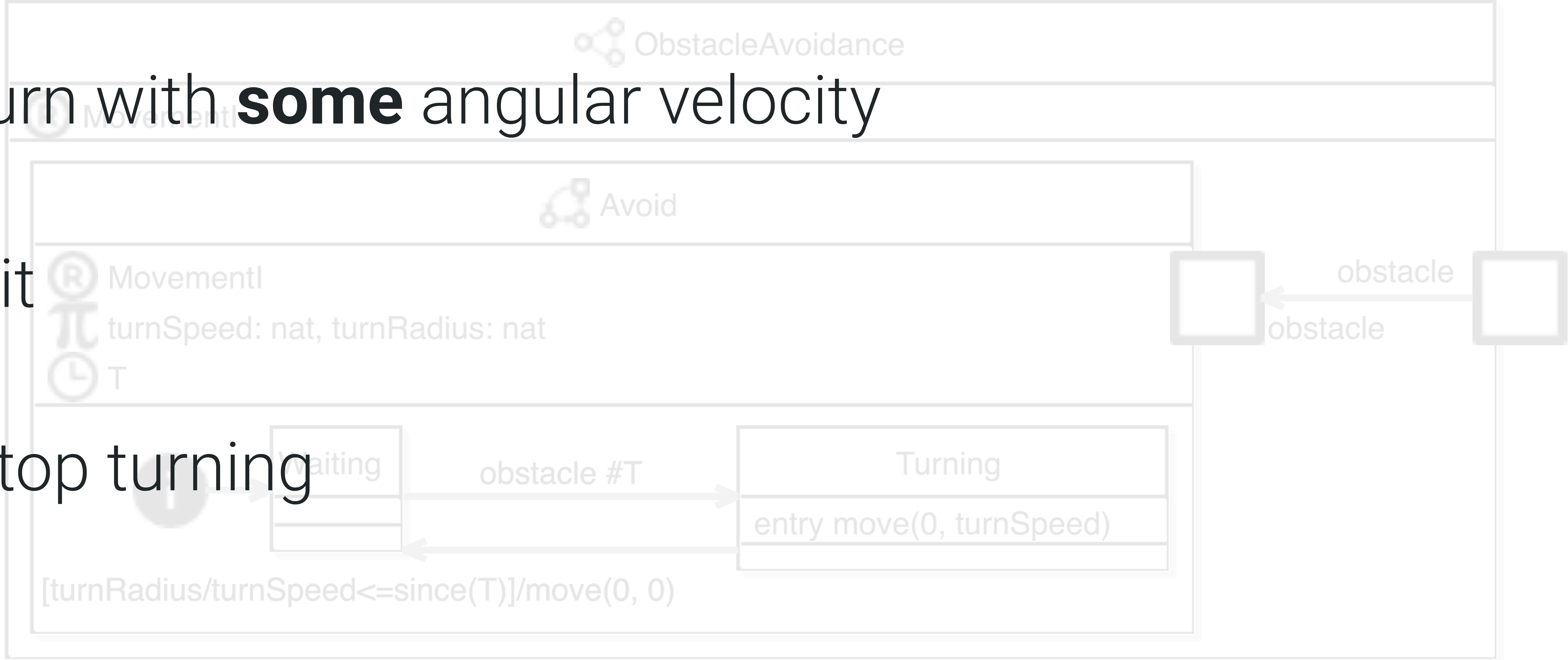
RoboCert
Behaviour set, in English

receive warning of obstacle

immediately turn with **some** angular velocity

wait **1** time unit

immediately stop turning



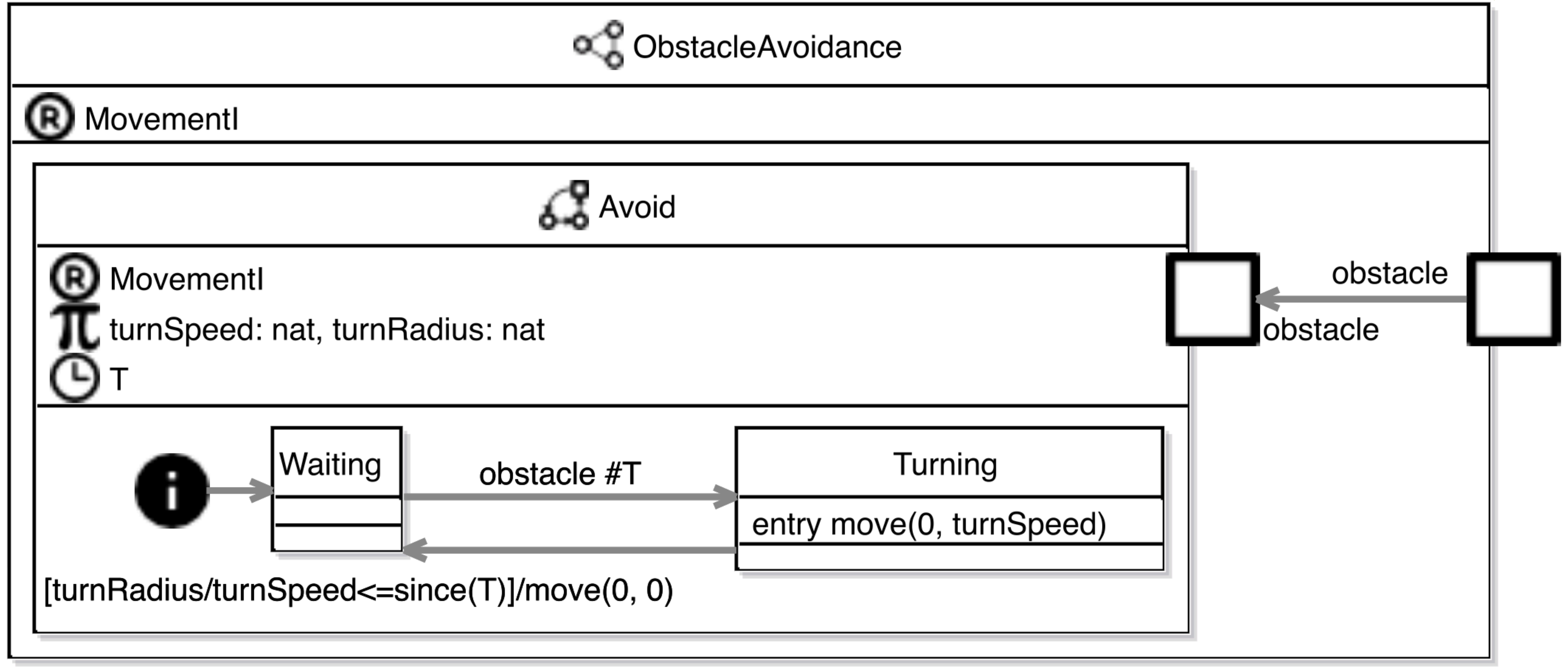
RoboCert
Behaviour set, as a sequence diagram

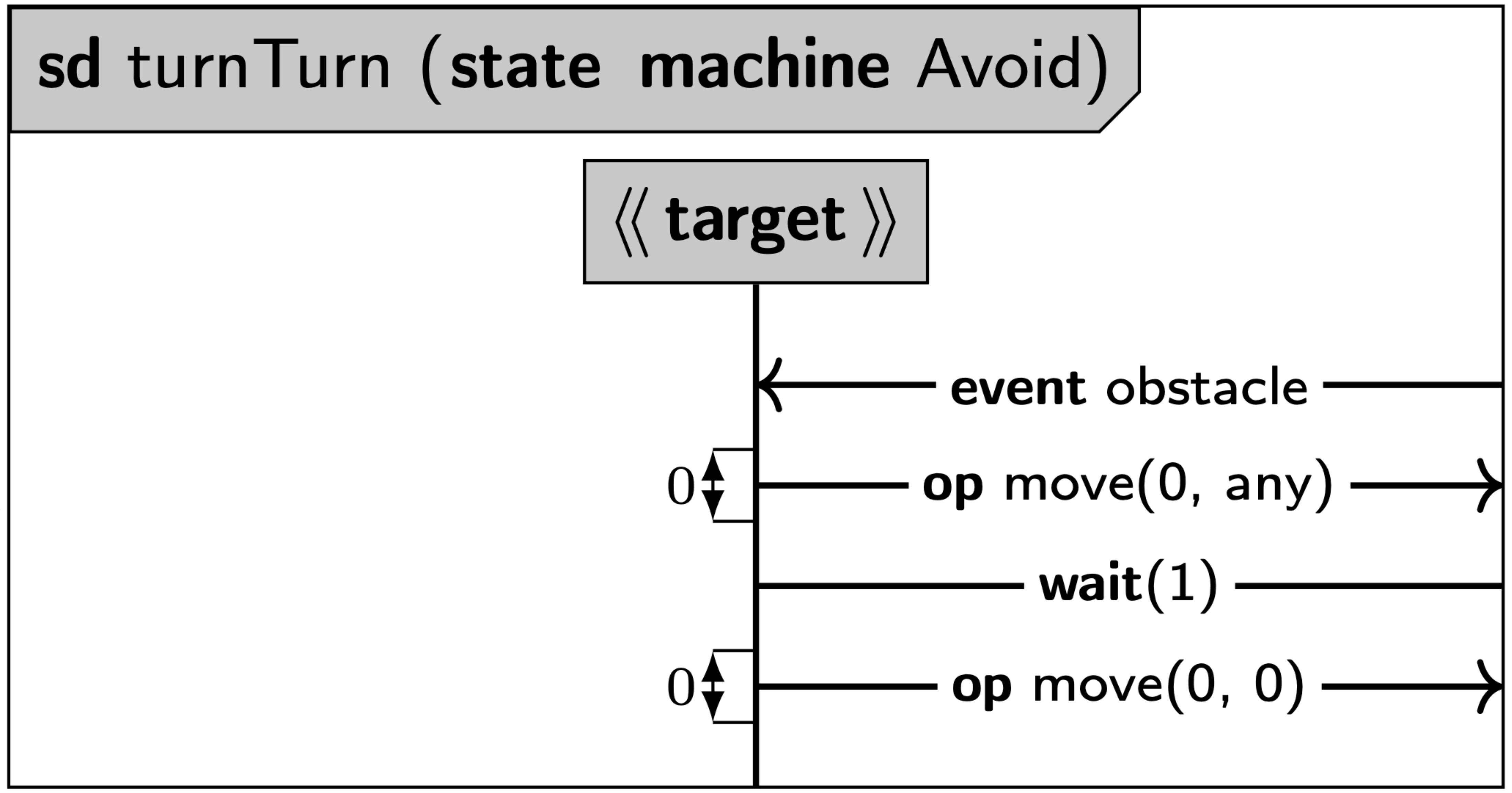
receive warning of obstacle

immediately turn with **some** angular velocity

wait **1** time unit

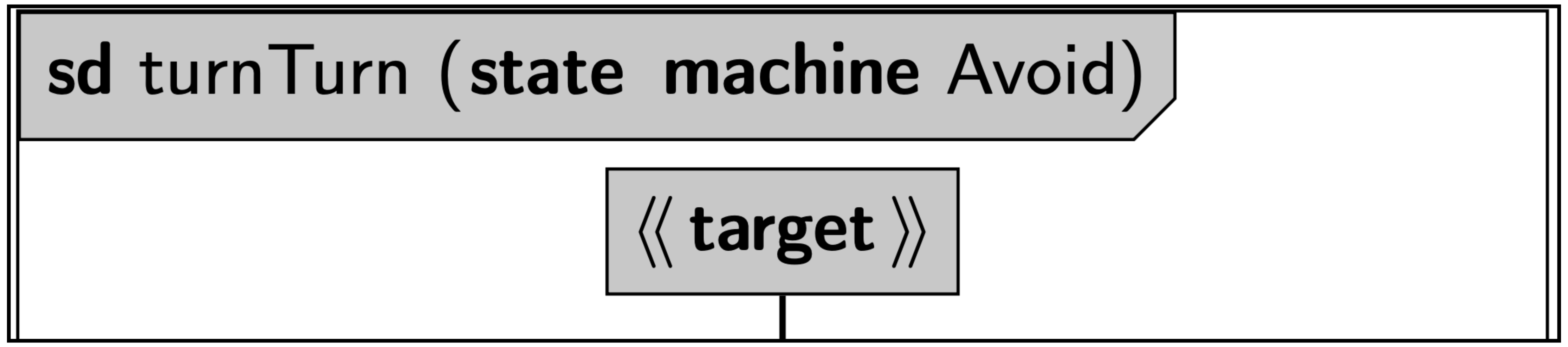
immediately stop turning





RoboCert

Interface with RoboChart component model

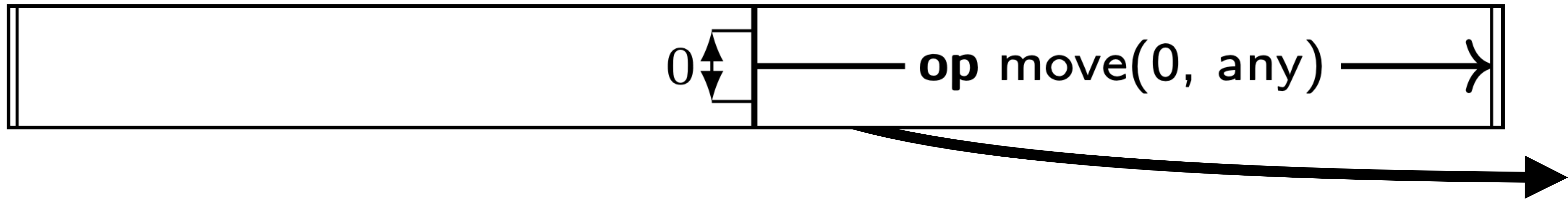


targets: **Robo**Chart components **or** ‘subcomponents of’ said components

lifelines: targets or subcomponents of targets

diagram edge corresponds to ‘world’: component(s) outside the target

RoboCert
Deadlines



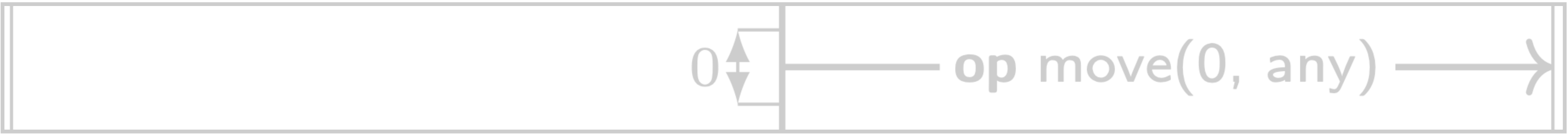
```
DeadlineF(0)( Cold( moveCall.0?_ → SKIP ) )

DeadlineF(T)(P) = P ⋀ WAIT(T); USTOP
```

Hybrid of UML notation and RoboChart discrete time semantics

Use timestop to encode deadlines

RoboCert
Deadlines as a use of timestop



0 op move(0, any)

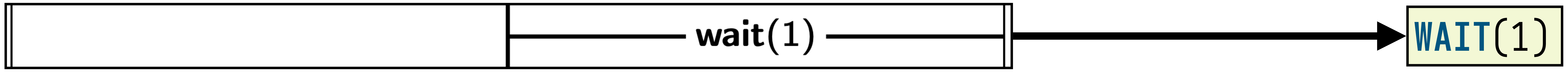
```
P ∧ WAIT(T); USTOP
```

both sides of interrupt must synchronise on tocks

after T units, rhs no longer permits time, so no more time can pass in P

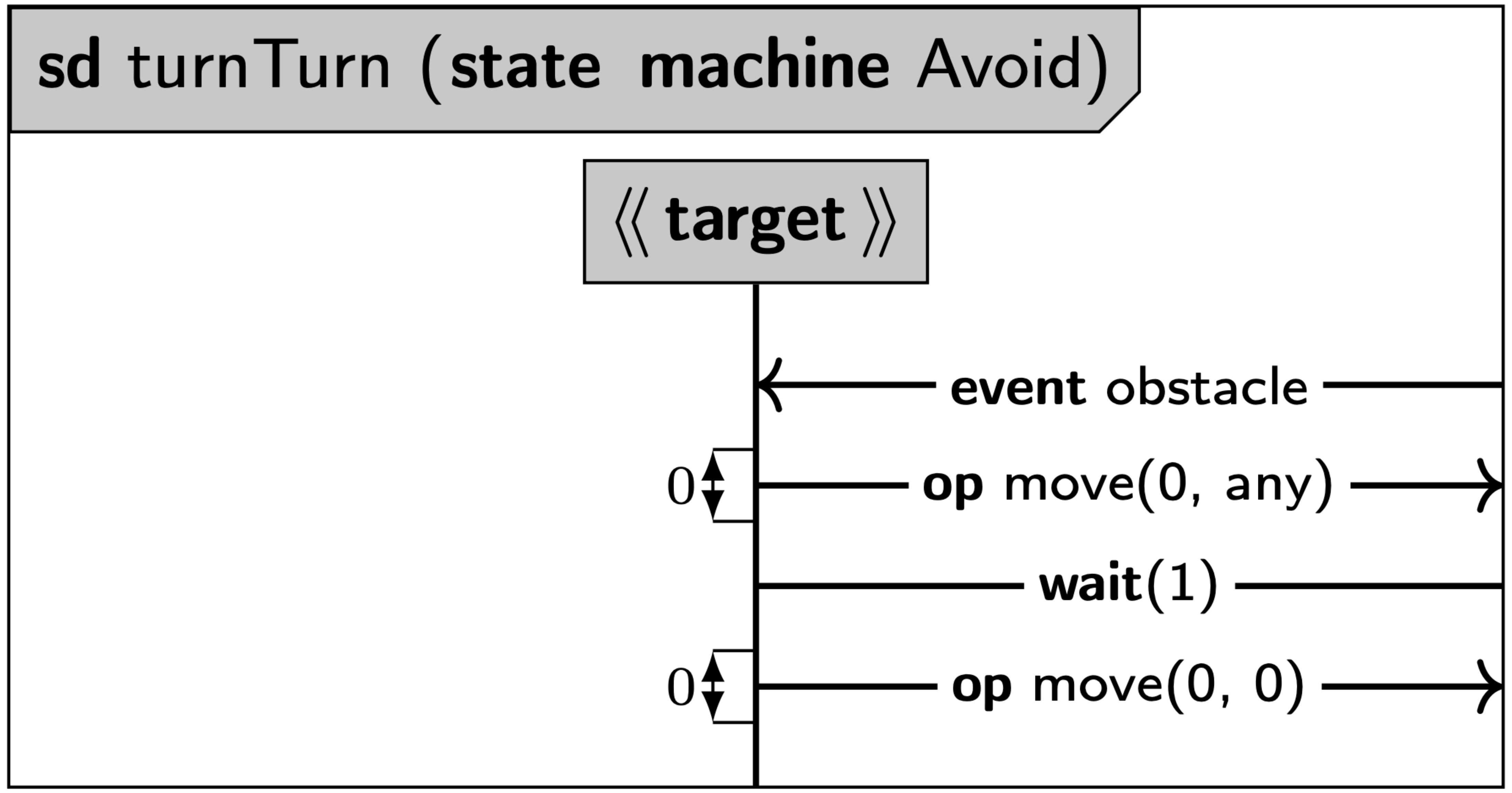
RoboCert

Waits



Waiting for *n* units = sequence of *n* **tock**s

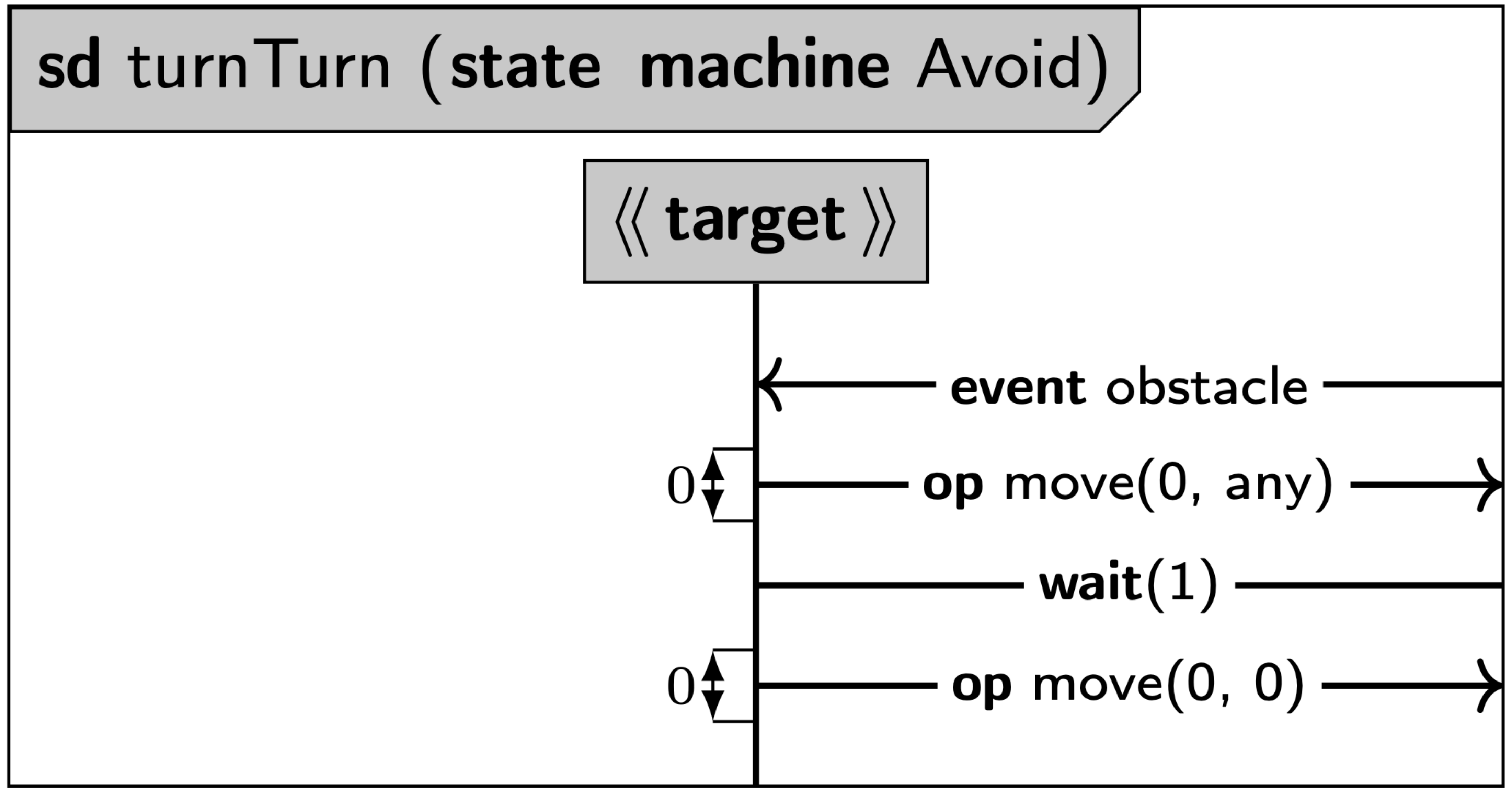
RoboCert
CSP of diagram



```
turnTurn = (
  Cold( obstacle.in → SKIP );
  DeadlineF(0)( Cold( moveCall.0?_ → SKIP ) );
  WAIT(1);
  DeadlineF(0)( Cold( moveCall.0.0 → SKIP ) )
)


Cold(P)         = P |~| tock → P
DeadlineF(T)(P) = P ⋀ WAIT(T); USTOP
```

RoboCert
Assertions



```
assertion A: turnTurn is      observed in the traces model
                   ... is not observed         timed
                   ... holds
                   ... does not hold
```

RoboCert
CSP of assertions

```
assertion A: turnTurn is     observed in the traces model
                  ... is not observed       timed
                  ... holds
                  ... does not hold
```

```
    ( Avoid    [T= turnTurn ; STOP )
not ( Avoid    [T= turnTurn ; STOP )
    ( turnTurn [T= Avoid           )
not ( turnTurn [T= Avoid           )
```

in **`timed`** model, we use semantic model **TT**

Concluding…

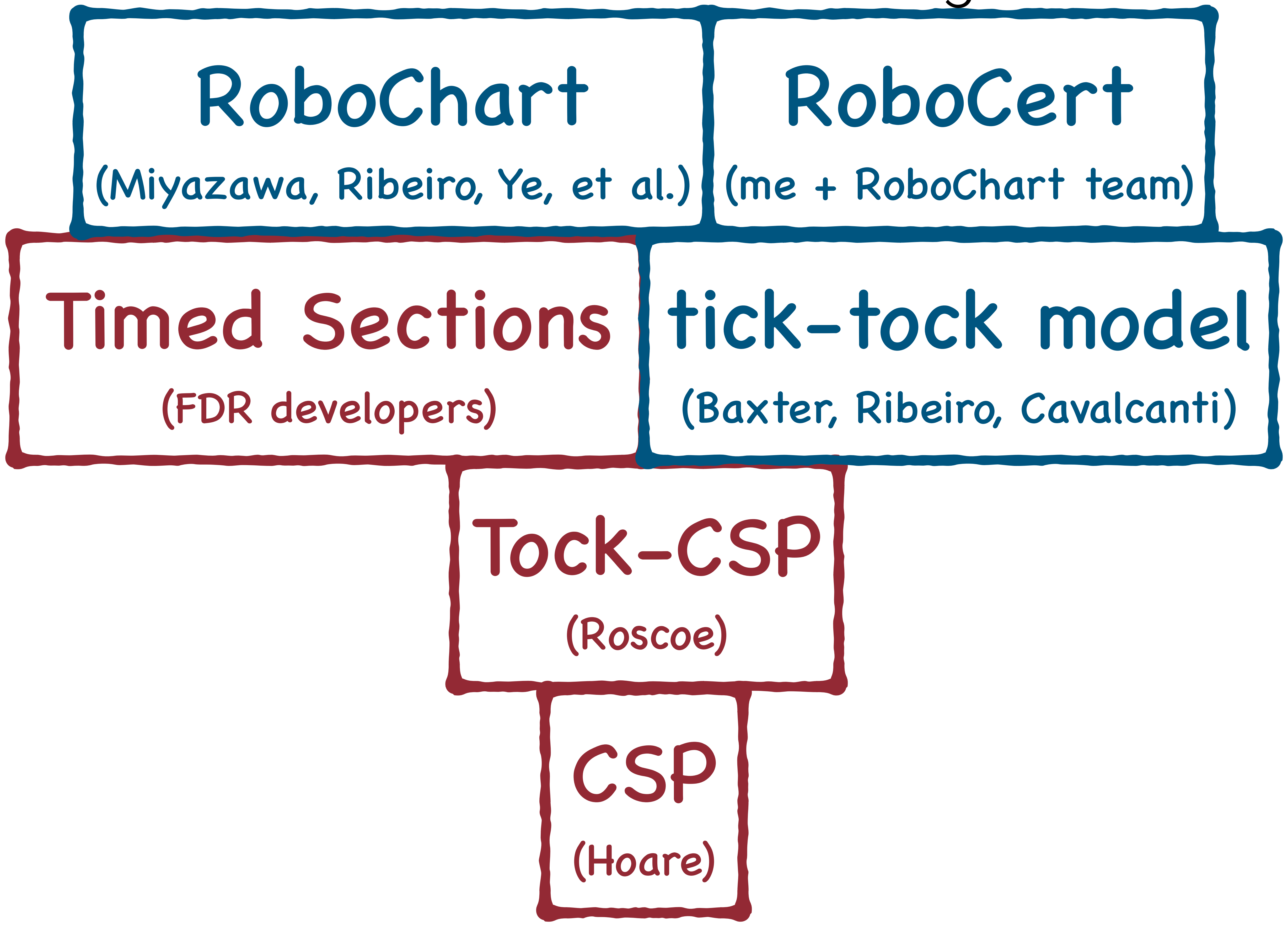
Building blocks towards reasoning

timed robotic reasoning

fundamentals of concurrent reasoning

Concluding...
Building blocks towards reasoning

timed robotic reasoning



fundamentals of concurrent reasoning

Concluding…
Further info

RoboChart manual	https://robostar.cs.york.ac.uk/publications/techreports/reports/robochart-reference.pdf
RoboCert manual	https://robostar.cs.york.ac.uk/publications/reports/robocert.pdf
Tool	https://robostar.cs.york.ac.uk/robotool/
Eclipse update site	https://robostar.cs.york.ac.uk/robotool/update/
RoboCert extras	https://github.com/UoY-RoboStar/robocert-evaluation