Hybrid UML and its Application to Specification and Test of Train Control Systems. Part 2

Kirsten Berkenkötter, Stefan Bisanz, Ulrich Hannemann and Jan Peleska

University of Bremen Universität Bremen , Germany
{kirsten,bisanz,ulrichh,jp}@informatik.uni-bremen.de

Outline

1. What is “HybridUML”?
2. A short reminder: Railway crossing case study
3. Applied HybridUML: Train controller

What is “HybridUML”?

- CHARON
- UML

⇒ UML 2.0 Profile: **HybridUML**

What is “HybridUML”?

- CHARON
  - \+ Hybrid automata
  - \+ Hierarchic modelling → “statecharts-like”
  - \+ Formal semantics
- UML

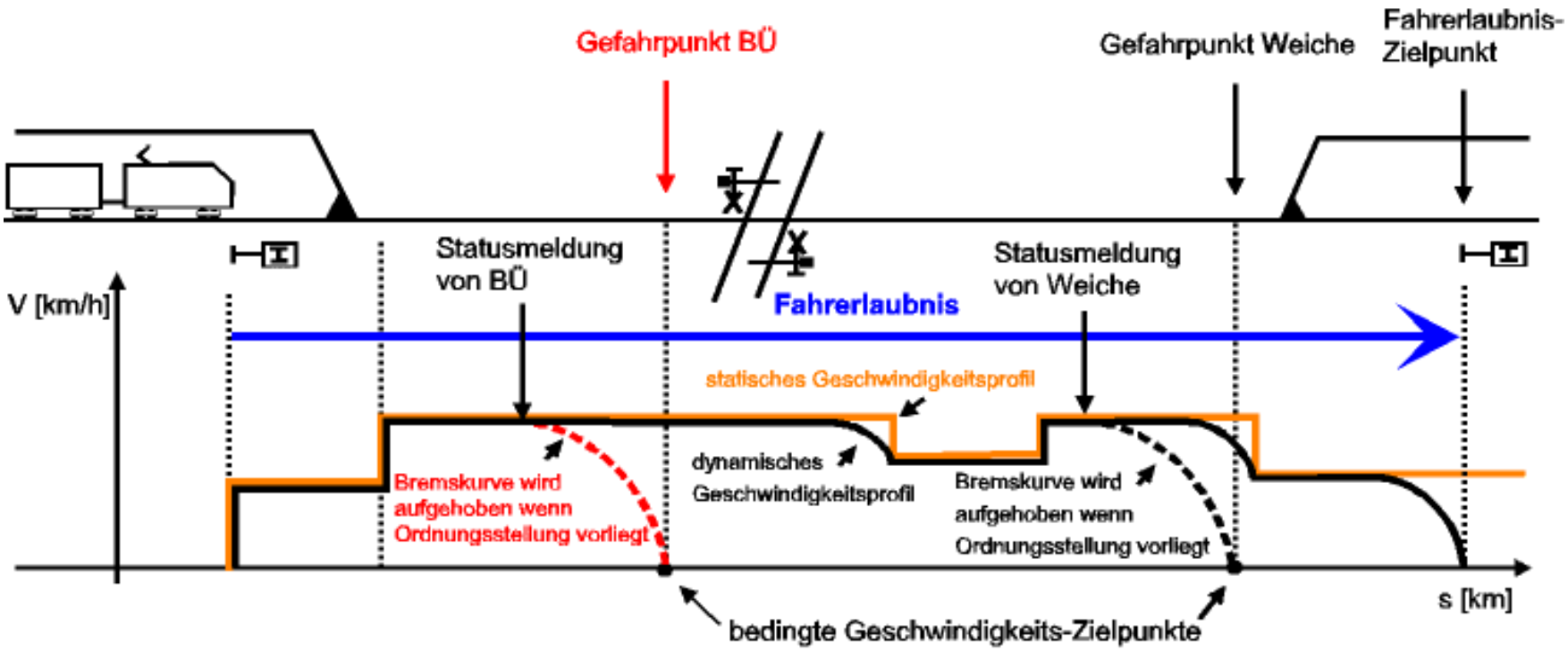
⇒ UML 2.0 Profile: **HybridUML**

What is “HybridUML”?

- CHARON
- UML 2.0
  - \+ well known, tool support . . .
  - − no sufficient support for real-time in UML 2.0
  - − no support for time-continuous modelling

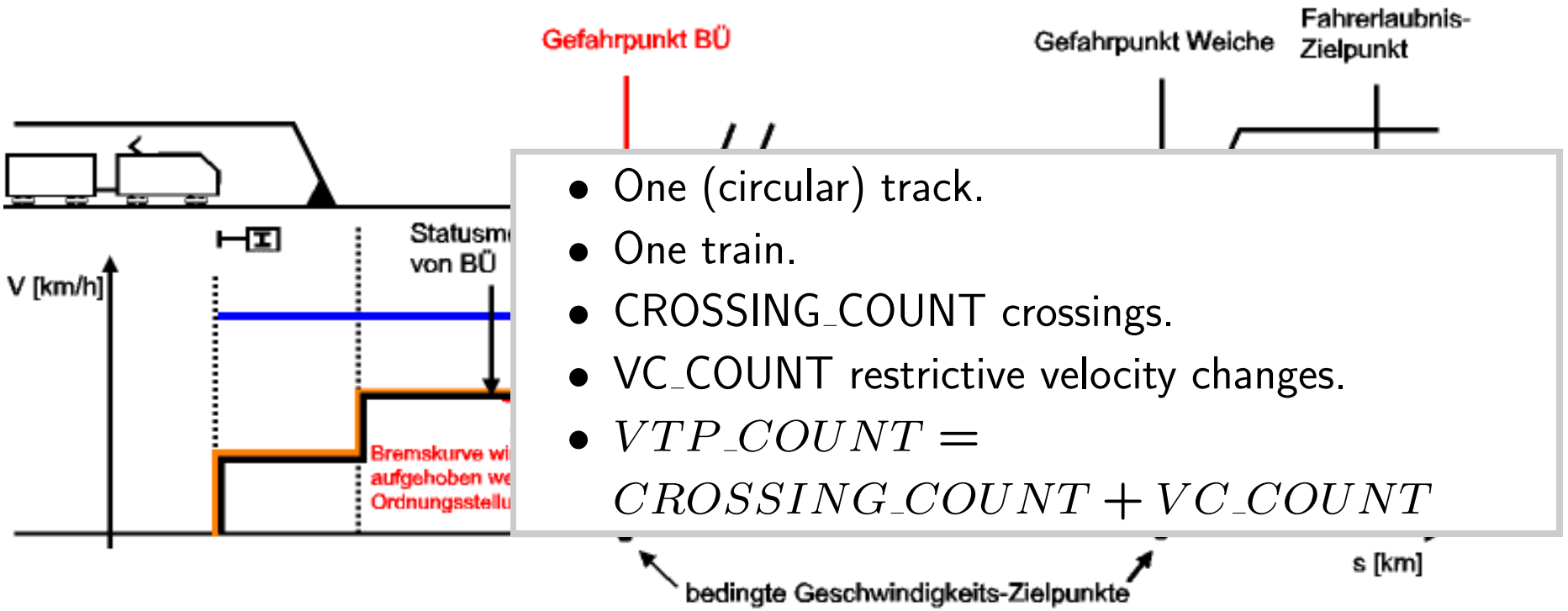
⇒ UML 2.0 Profile: **HybridUML**

A short reminder: Railway crossing case study



Scenario: Radio controlled train operation.

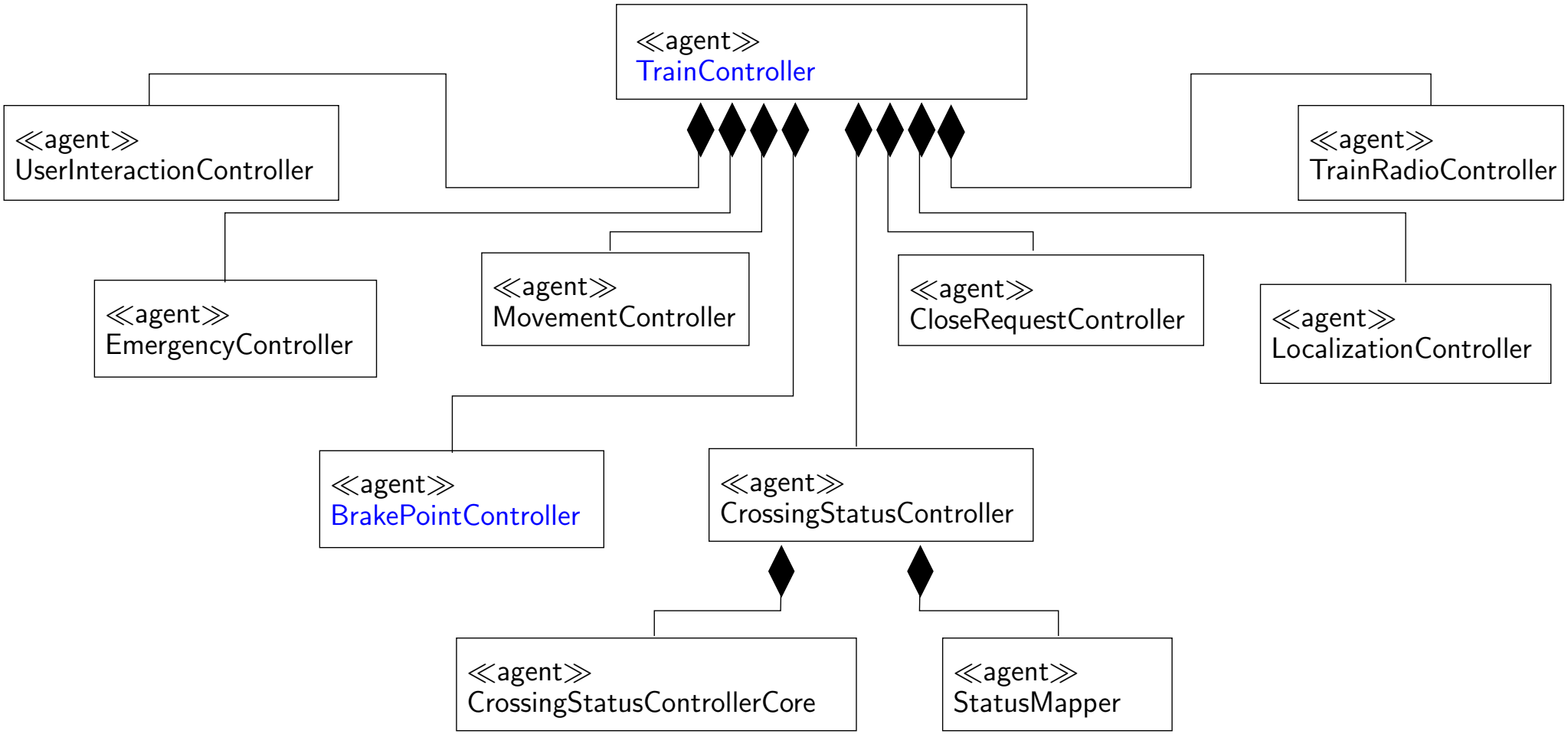
A short reminder: Railway crossing case study



- One (circular) track.
- One train.
- CROSSING_COUNT crossings.
- VC_COUNT restrictive velocity changes.
- $VTP\_COUNT = CROSSING\_COUNT + VC\_COUNT$

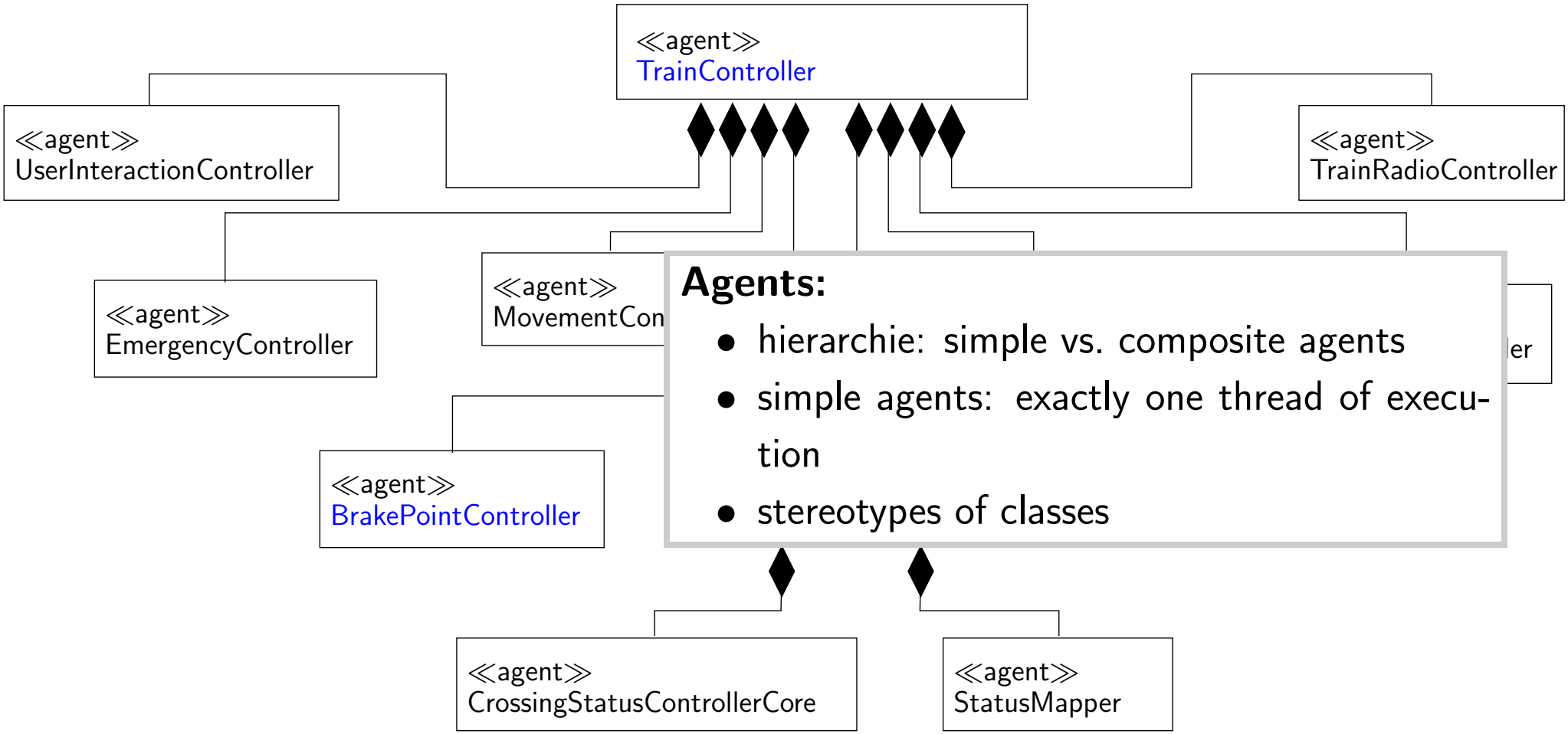
Scenario: Radio controlled train operation.

Applied HybridUML: Train controller



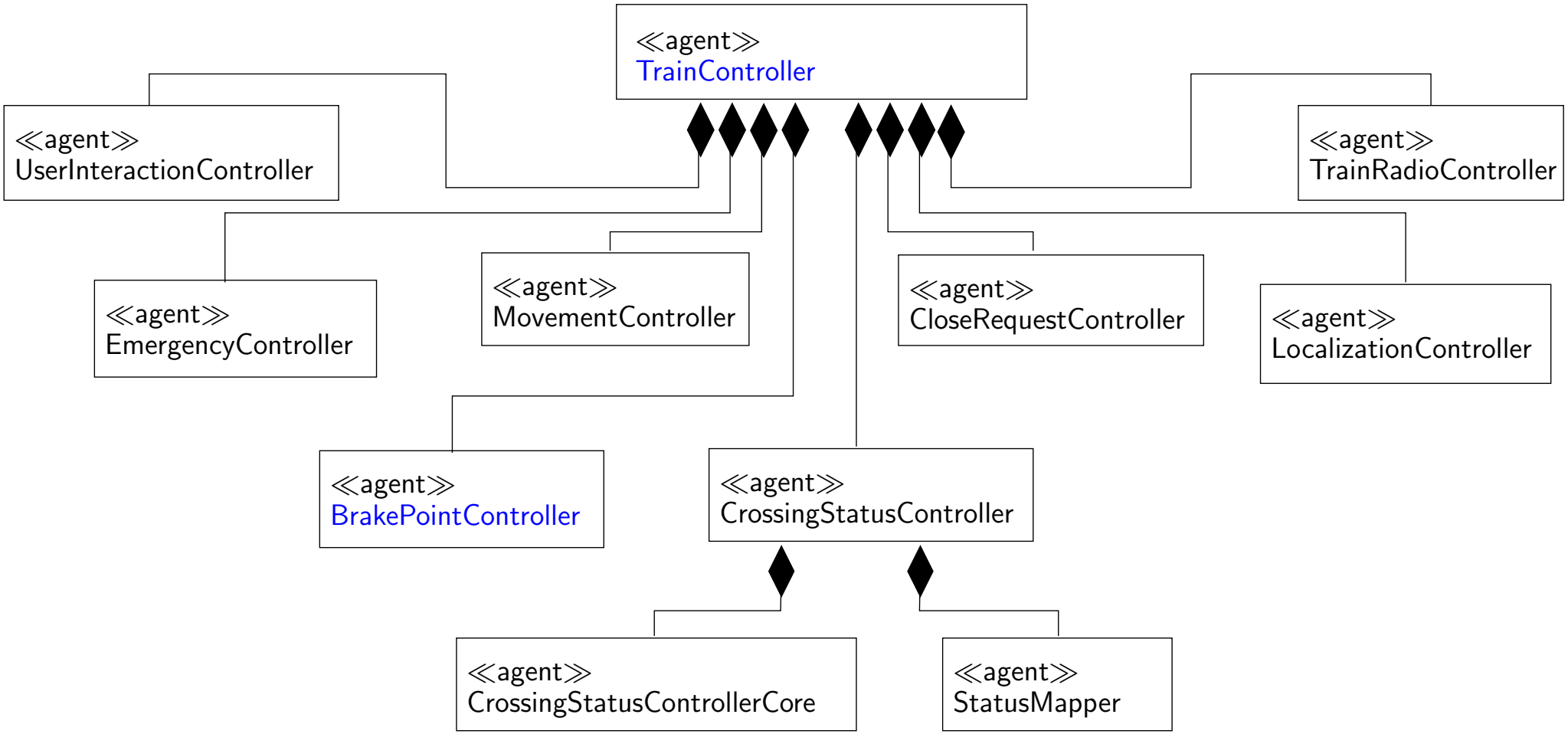
Subagents of TrainController

Applied HybridUML: Train controller

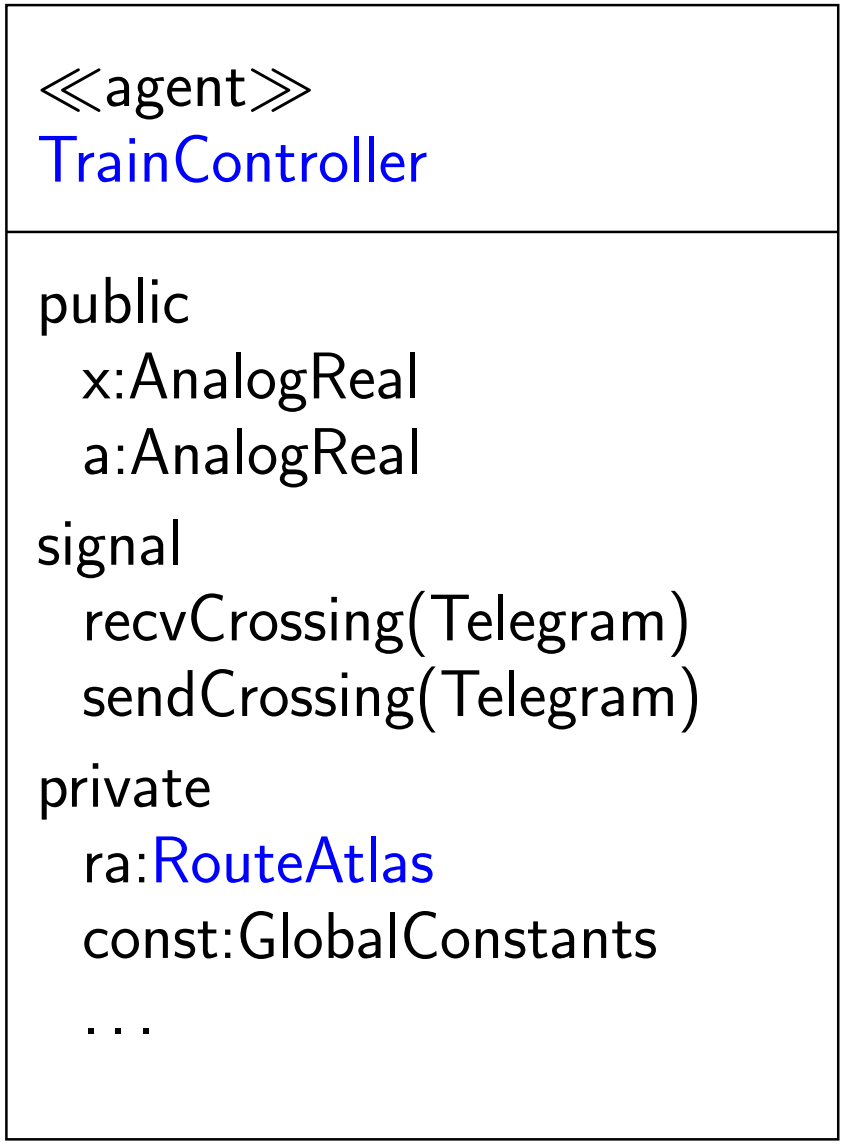


Subagents of TrainController

Applied HybridUML: Train controller



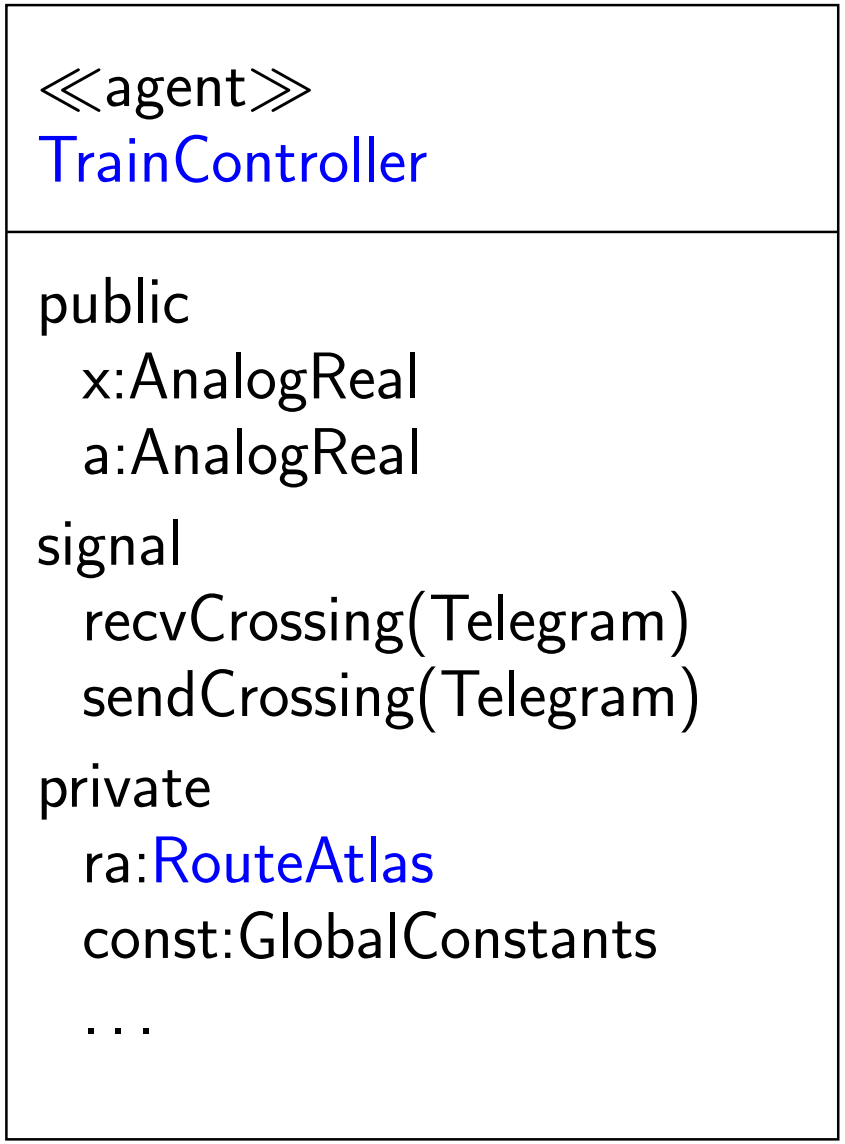
Subagents of TrainController



Class view of TrainController

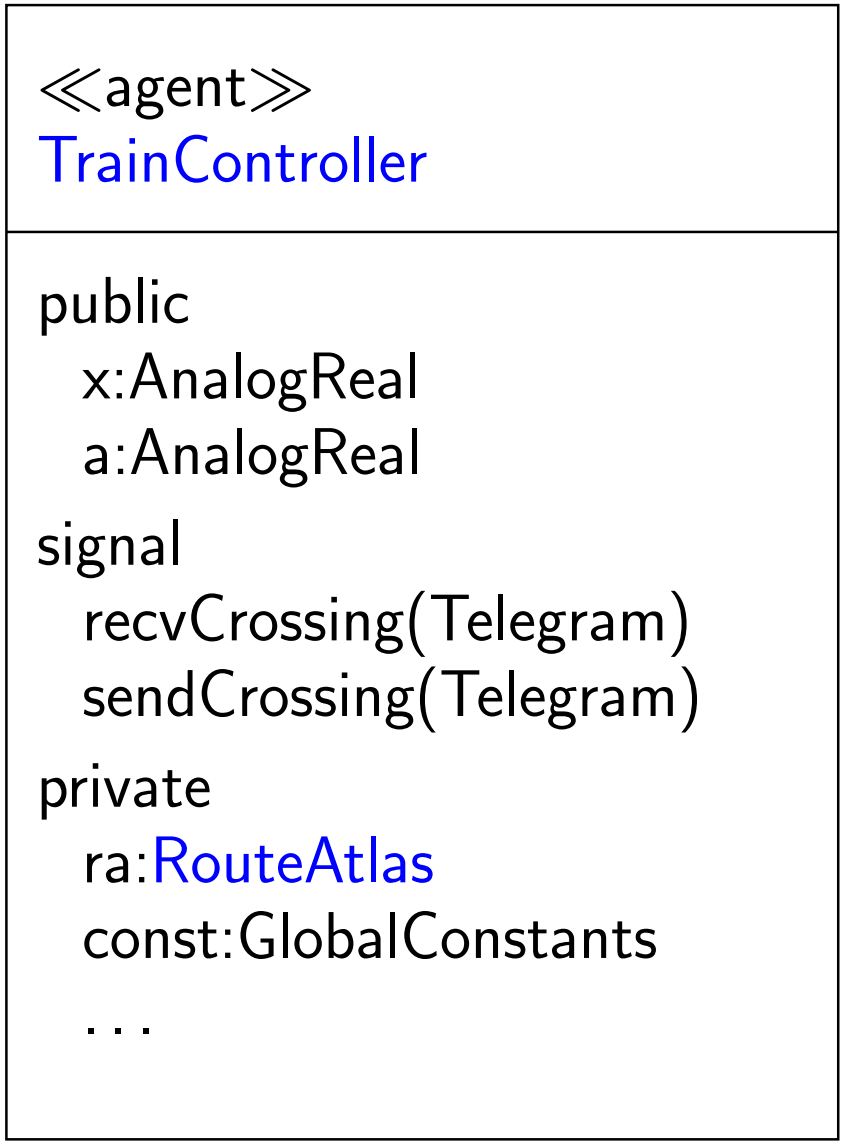
←Subagents of TrainController

- public and private variables
- events
- introduction of AnalogReal



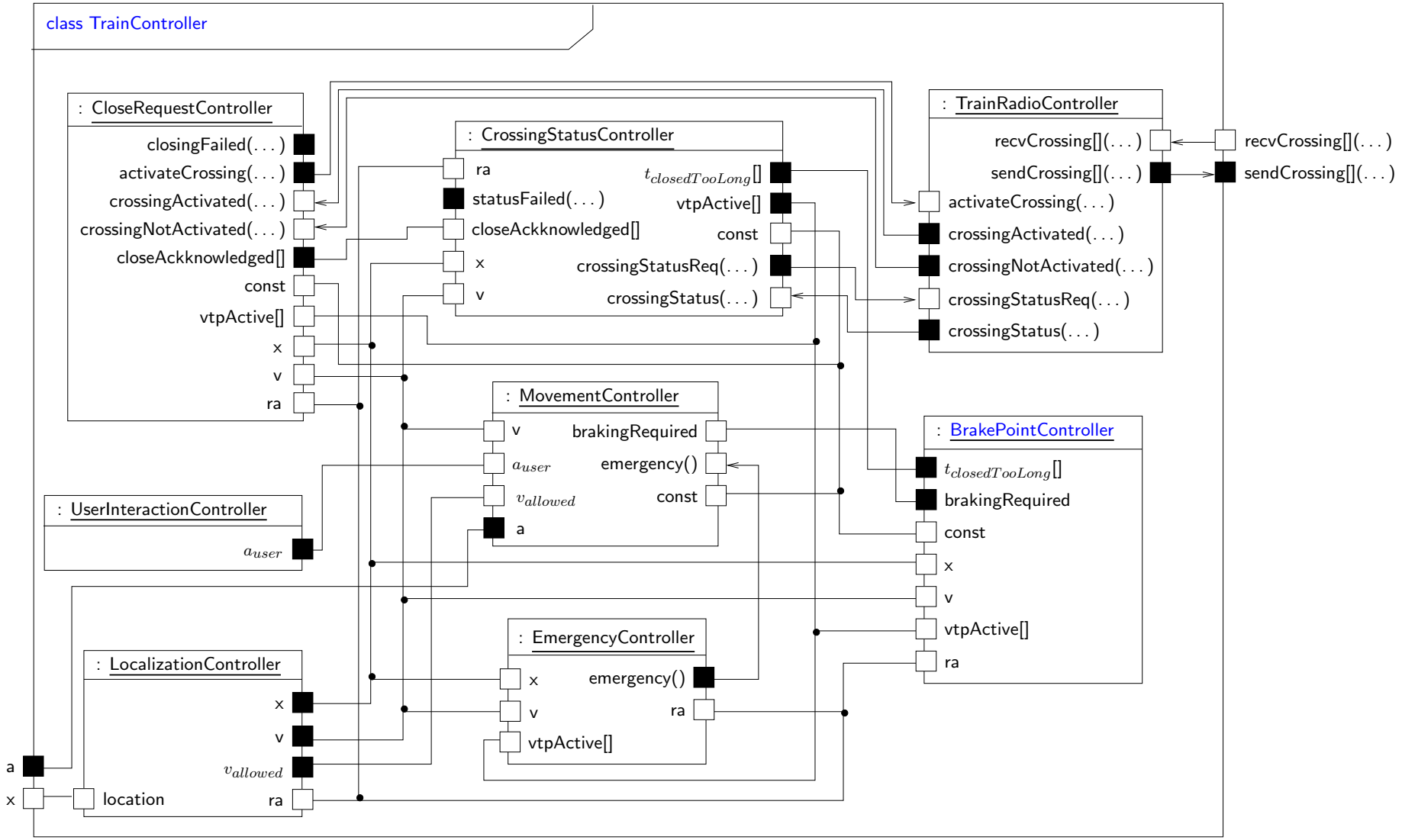
Class view of TrainController

←Subagents of TrainController

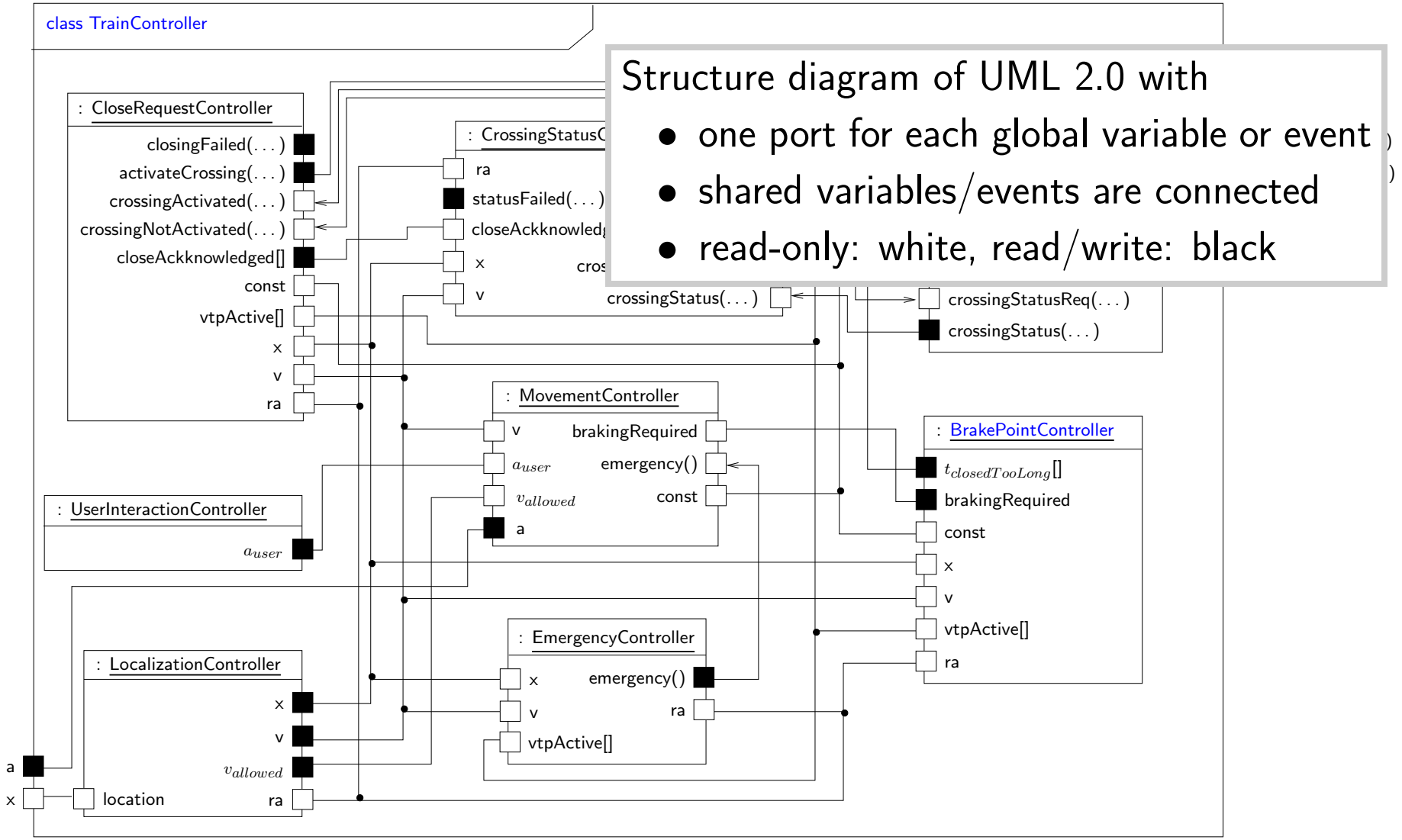


Class view of TrainController

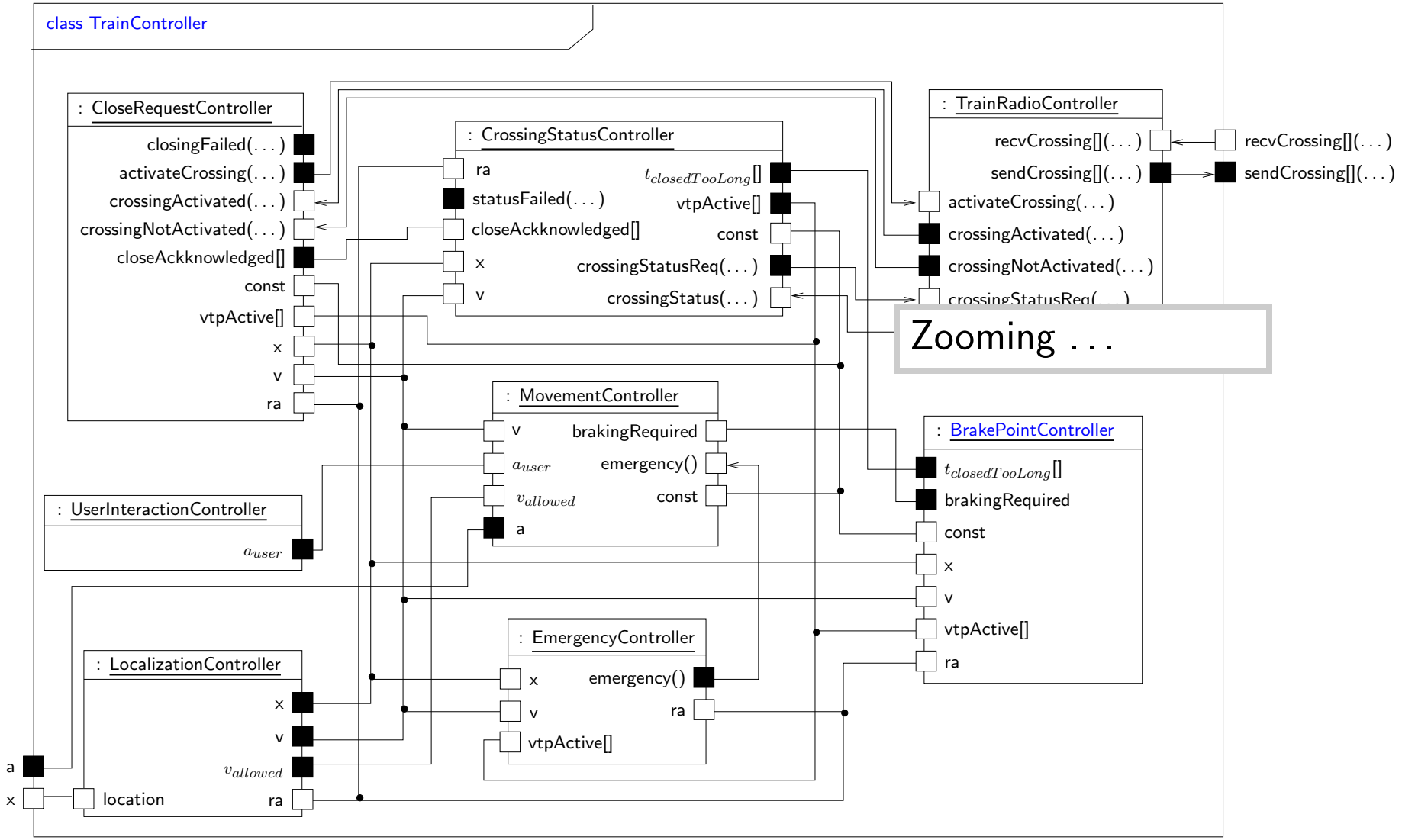
←Subagents of TrainController



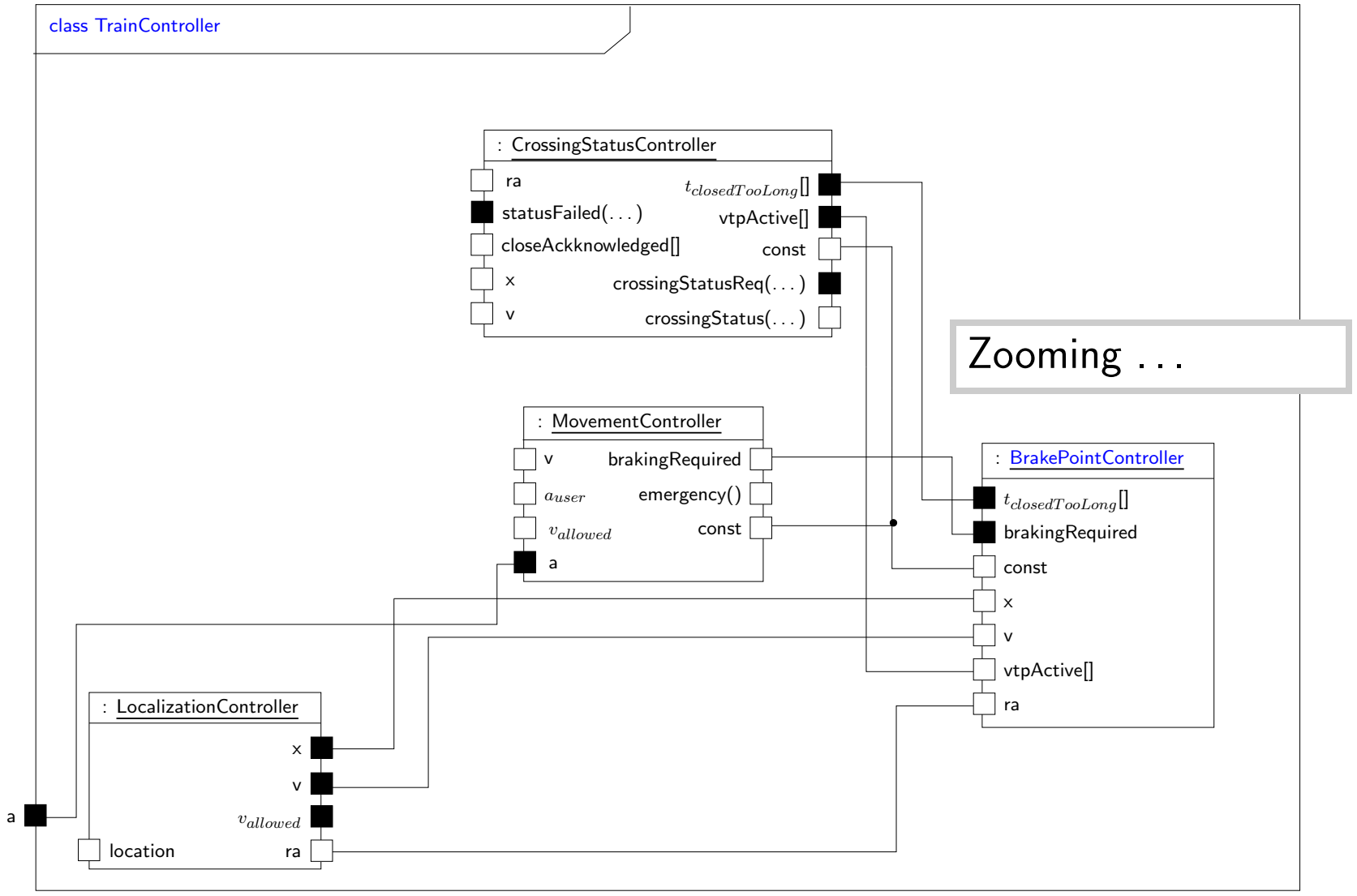
Structure diagram of TrainController



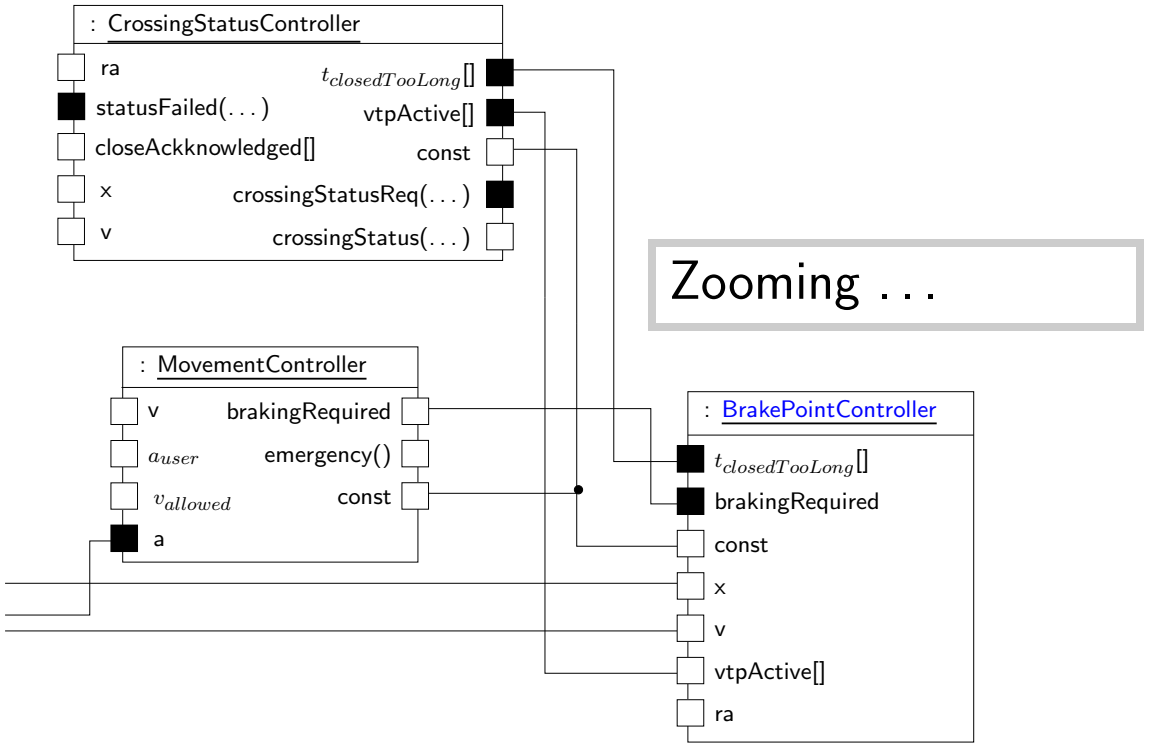
Structure diagram of TrainController



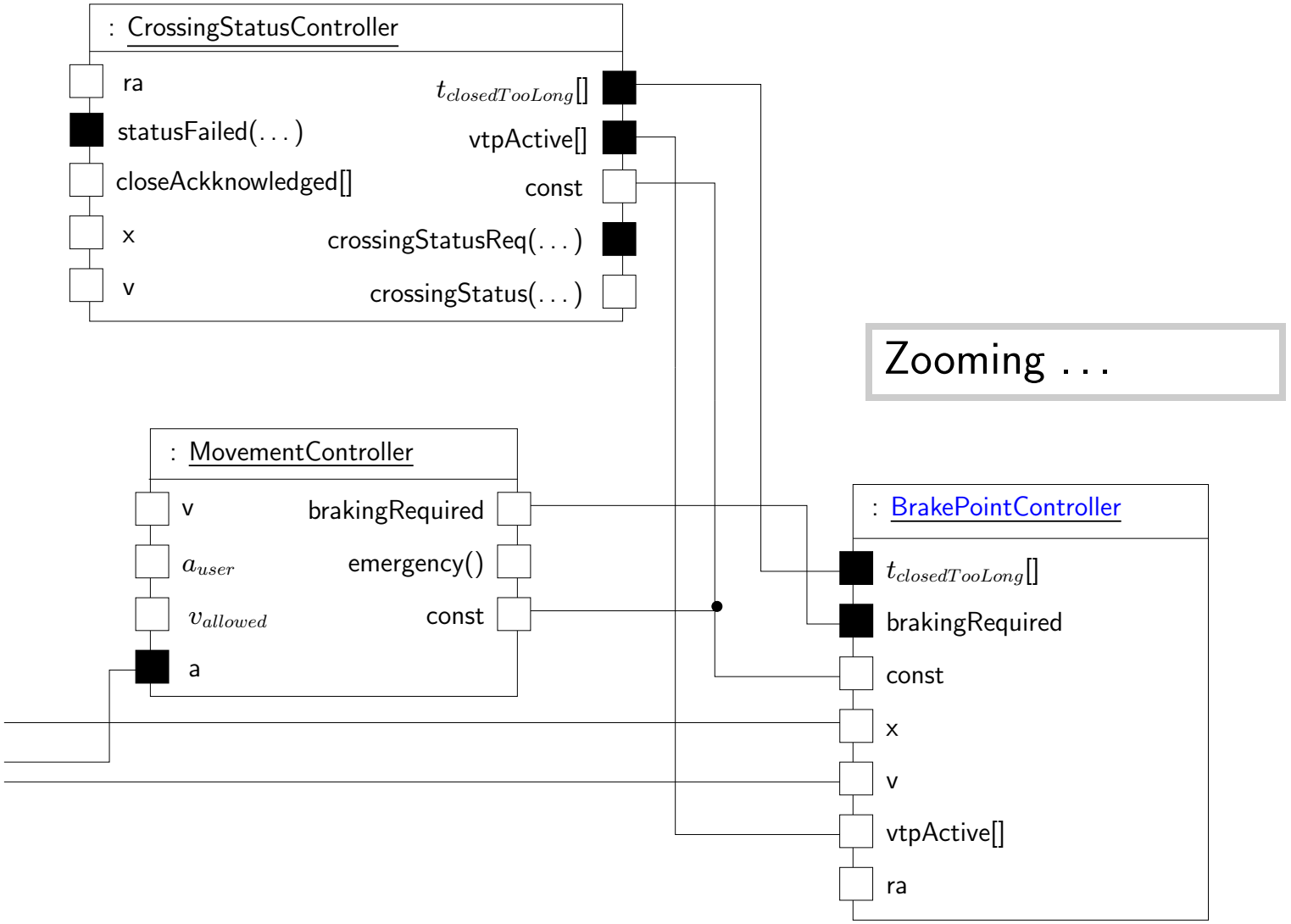
Structure diagram of TrainController



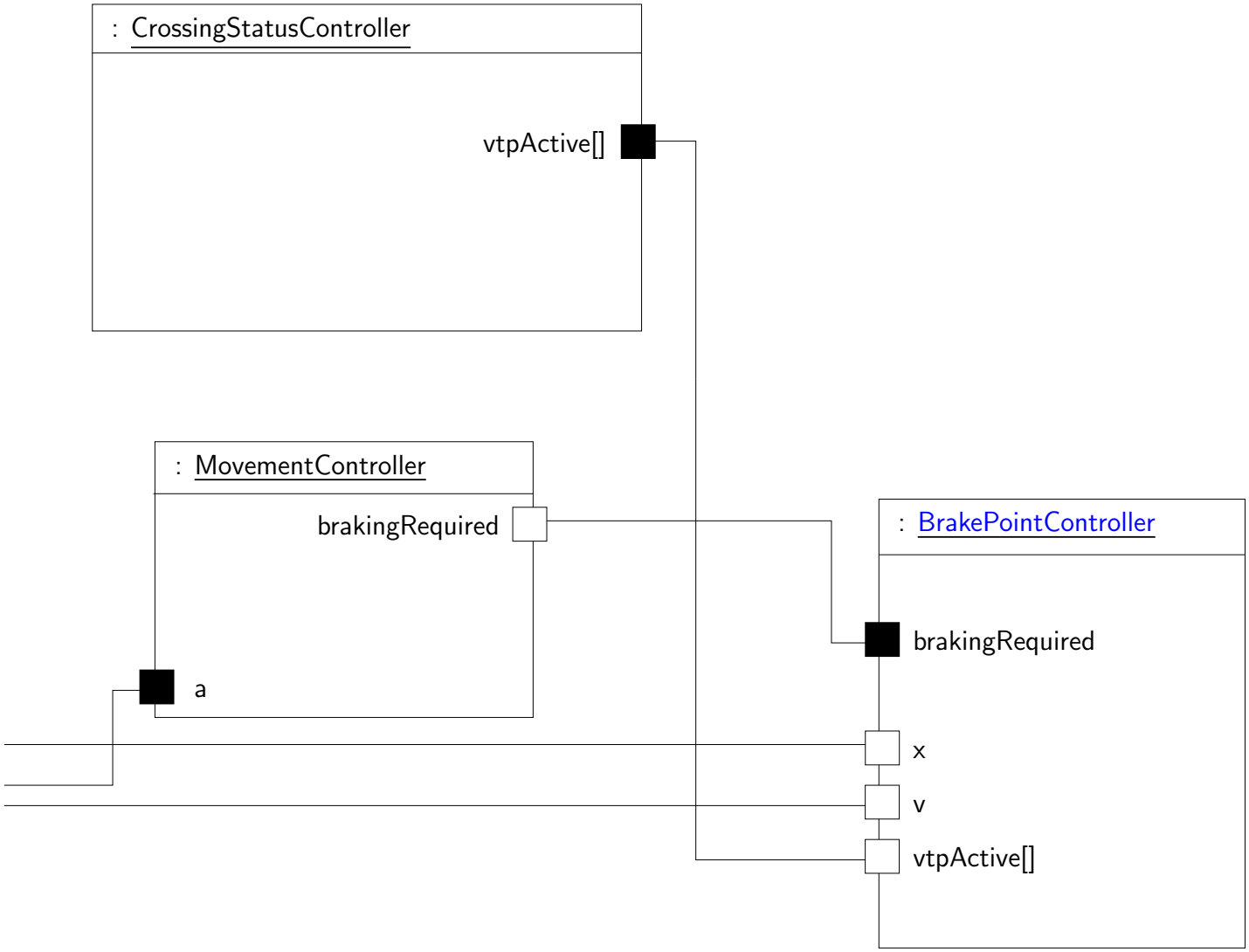
Focus on BrakePointController



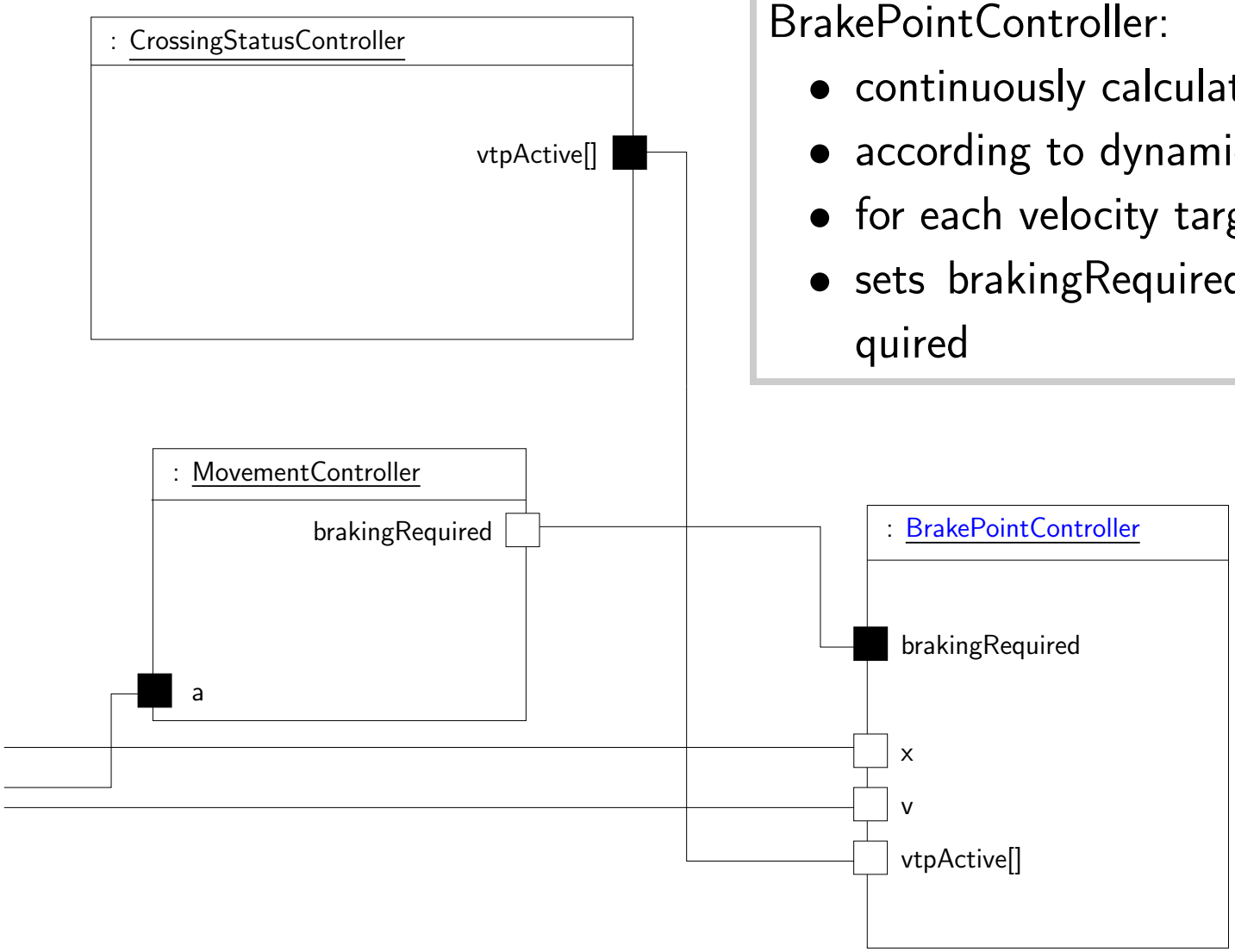
Focus on BrakePointController



Focus on BrakePointController



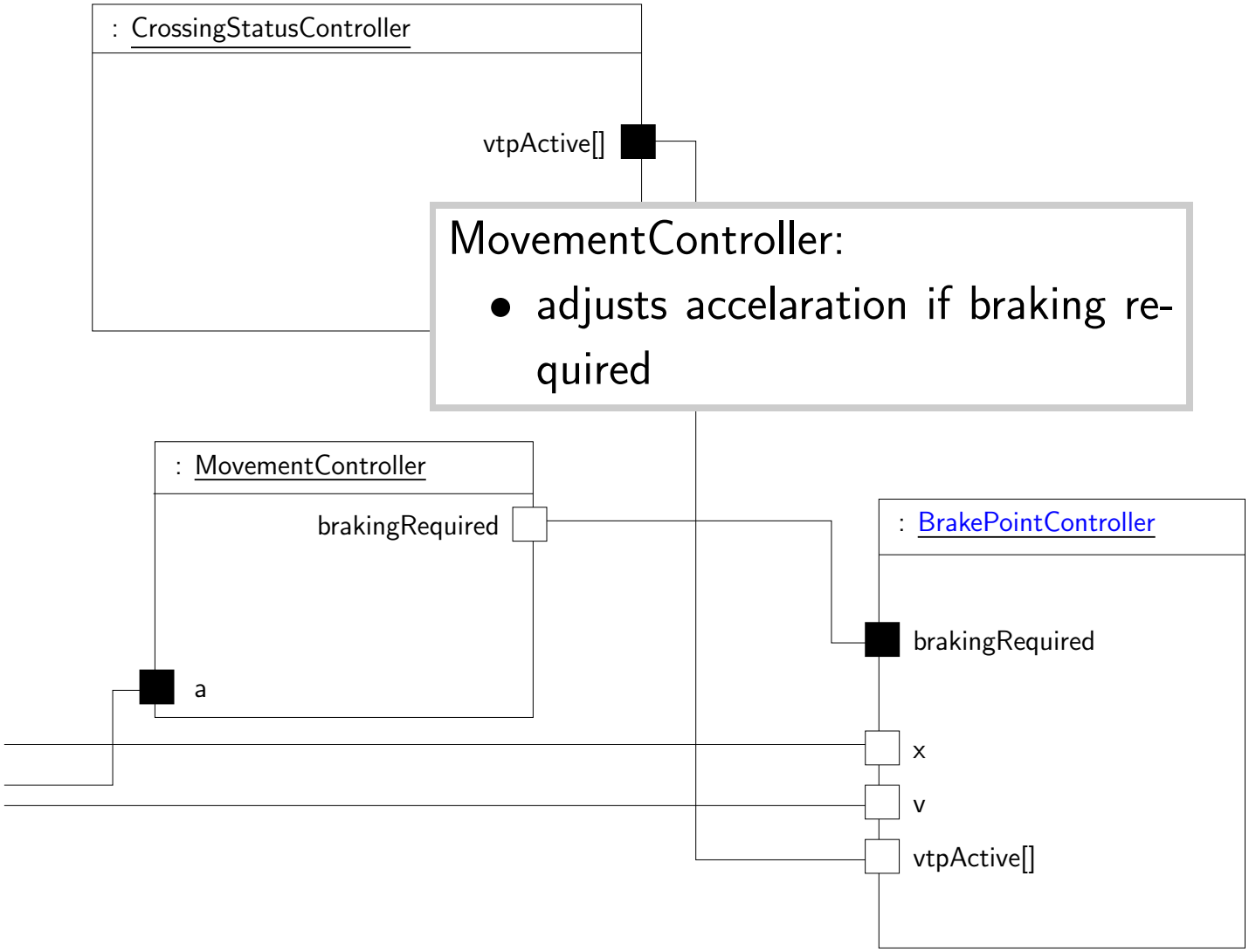
Focus on BrakePointController



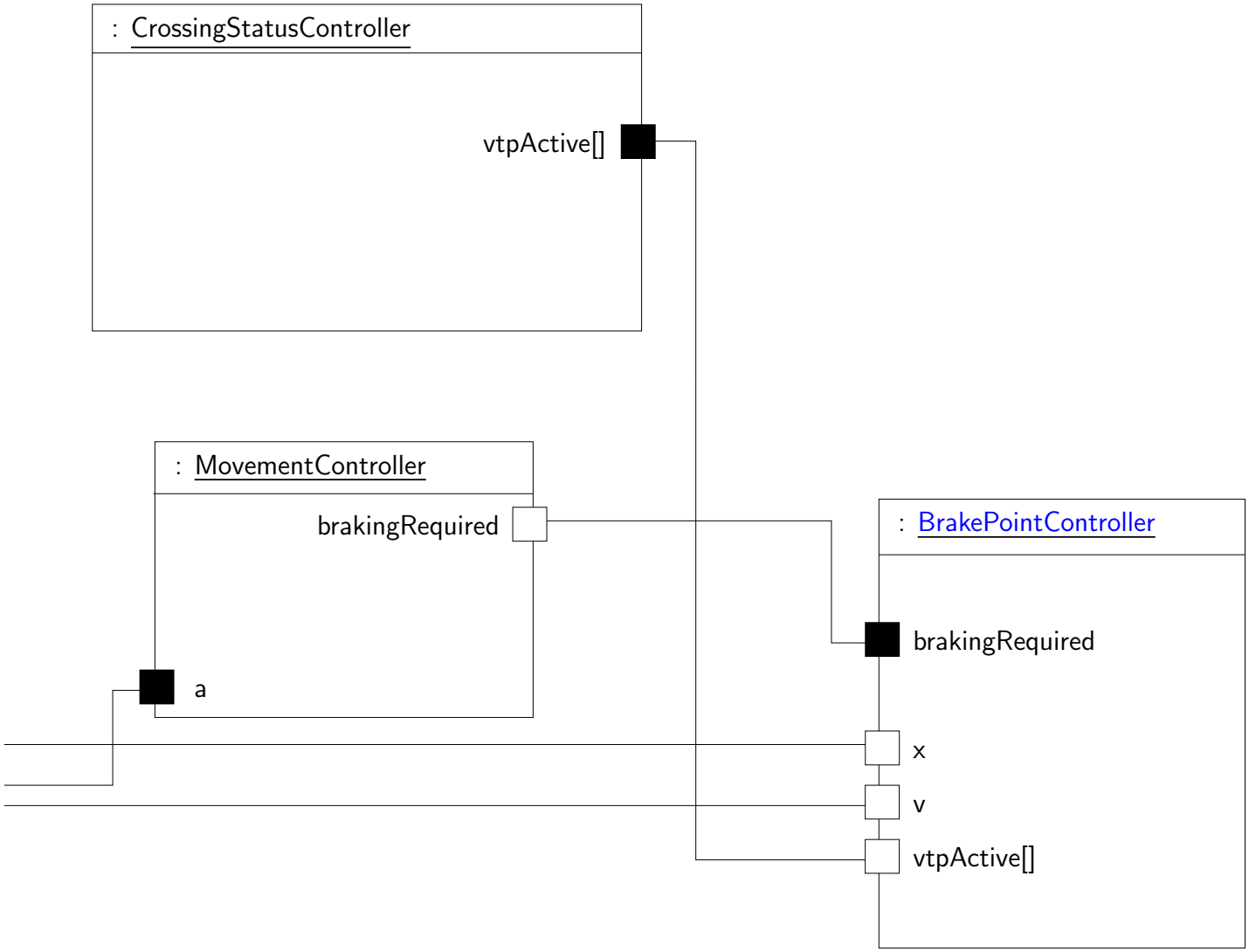
BrakePointController:

- continuously calculates brake points
- according to dynamic velocity profile
- for each velocity target point (VTP)
- sets brakingRequired iff braking is required

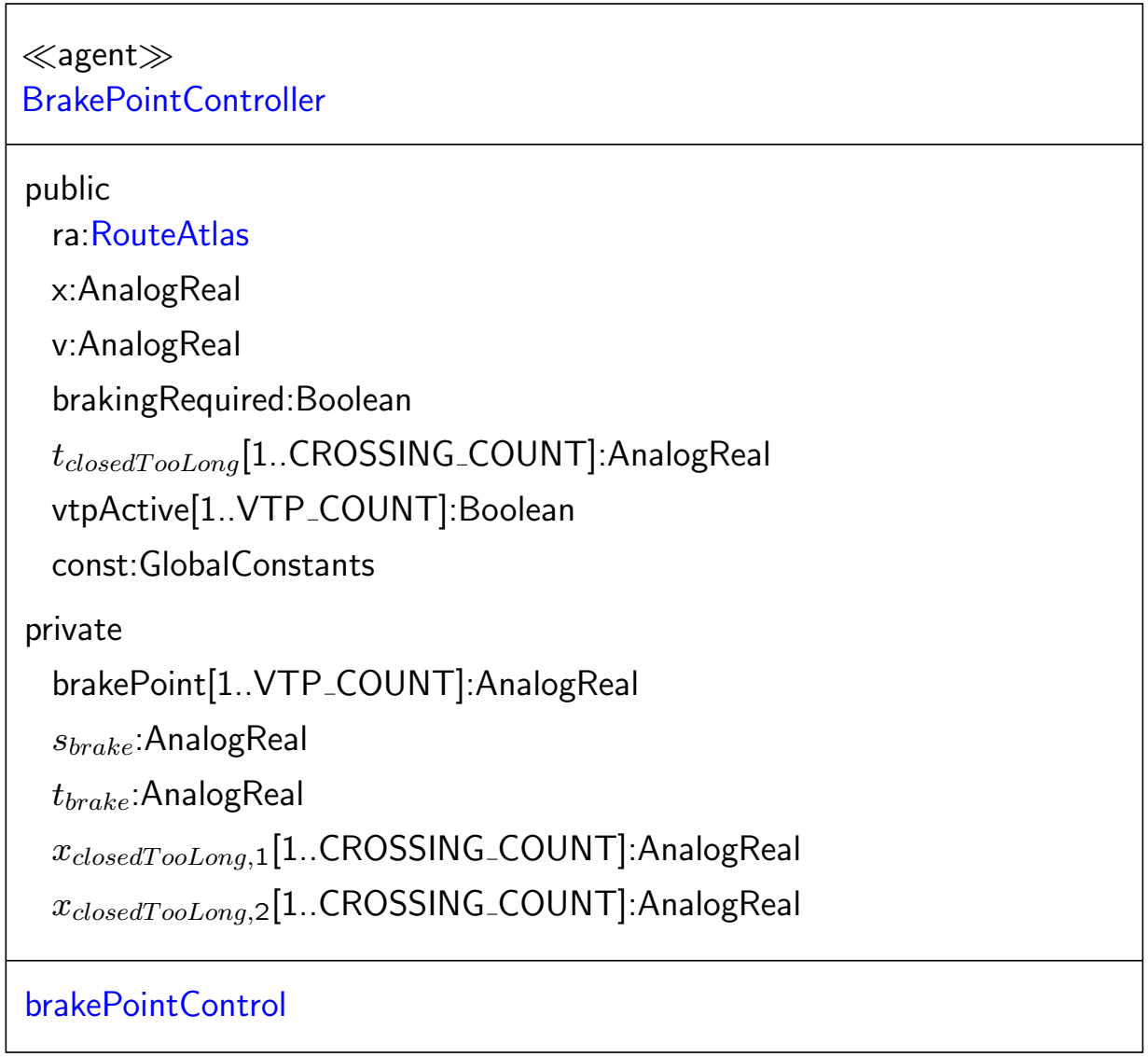
Focus on BrakePointController



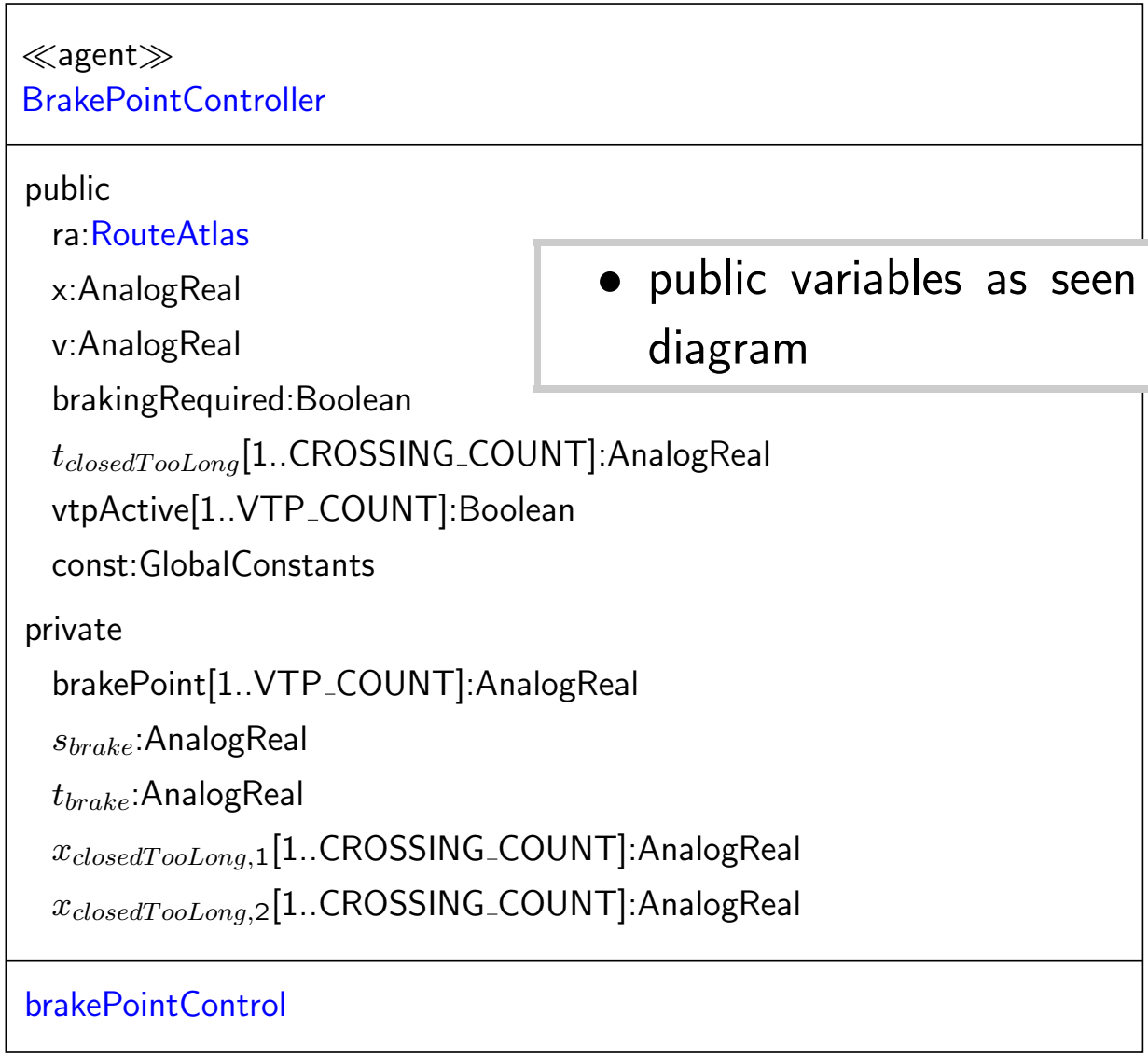
Focus on BrakePointController



Focus on BrakePointController

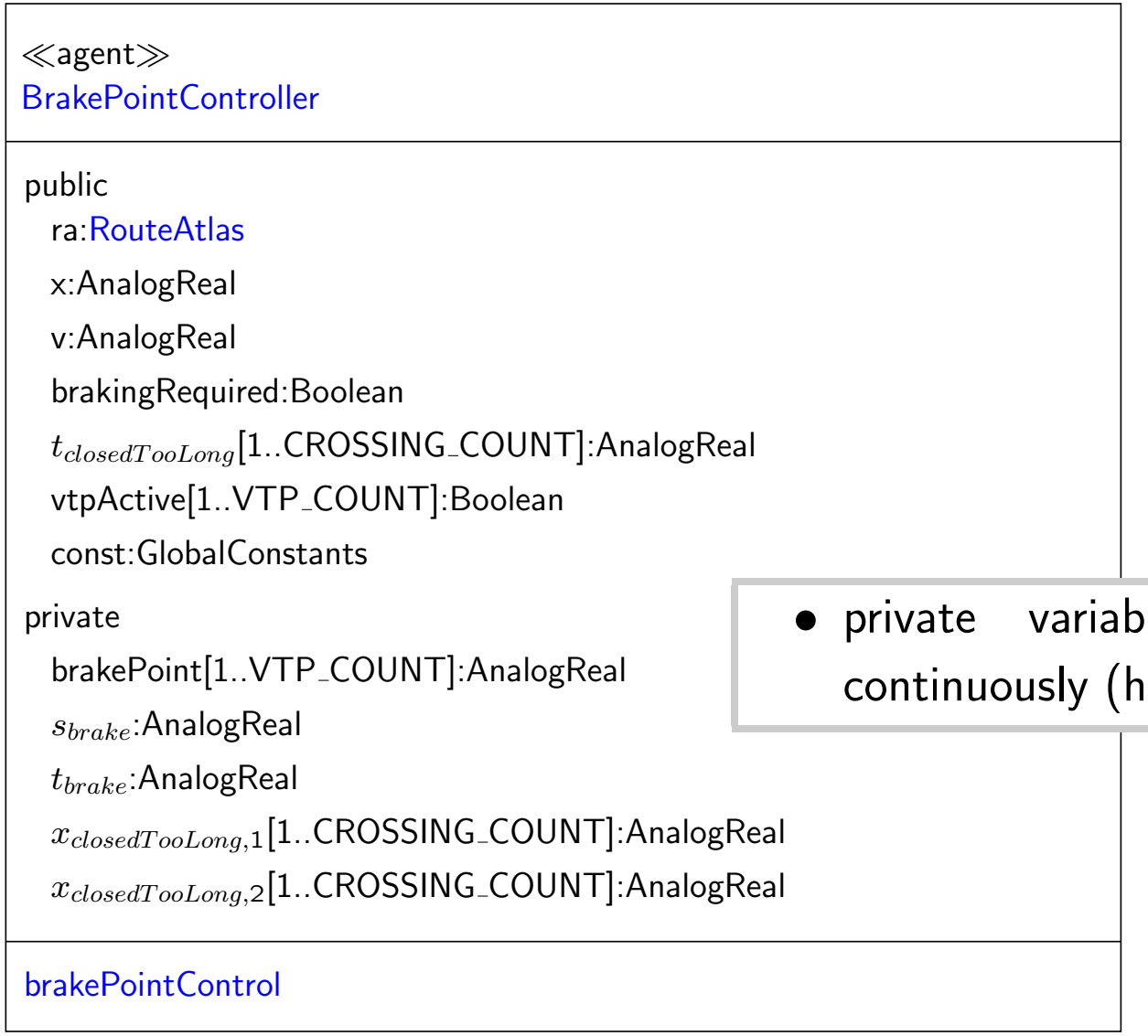


Class view of BrakePointController



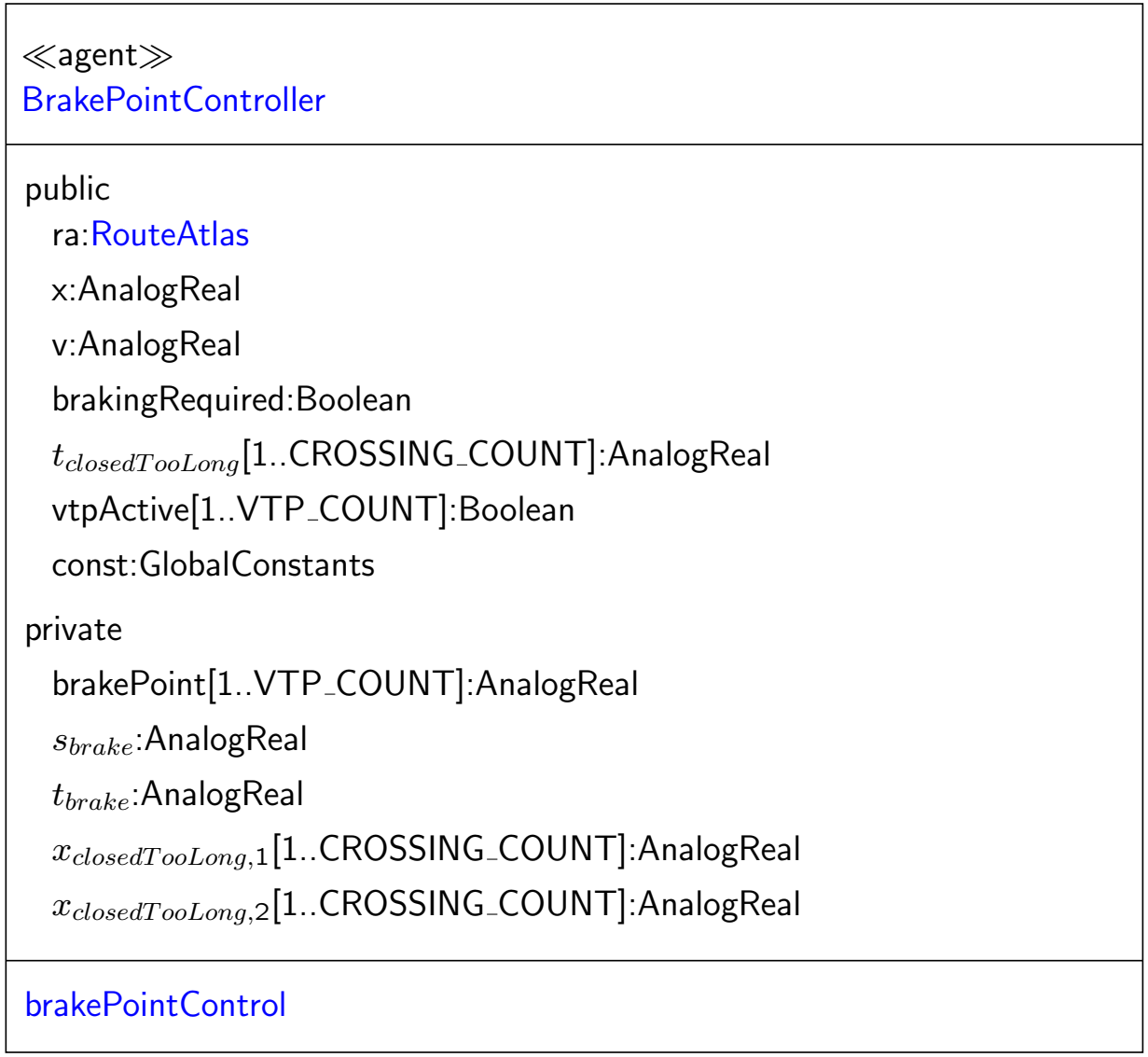
- public variables as seen in structure diagram

Class view of BrakePointController

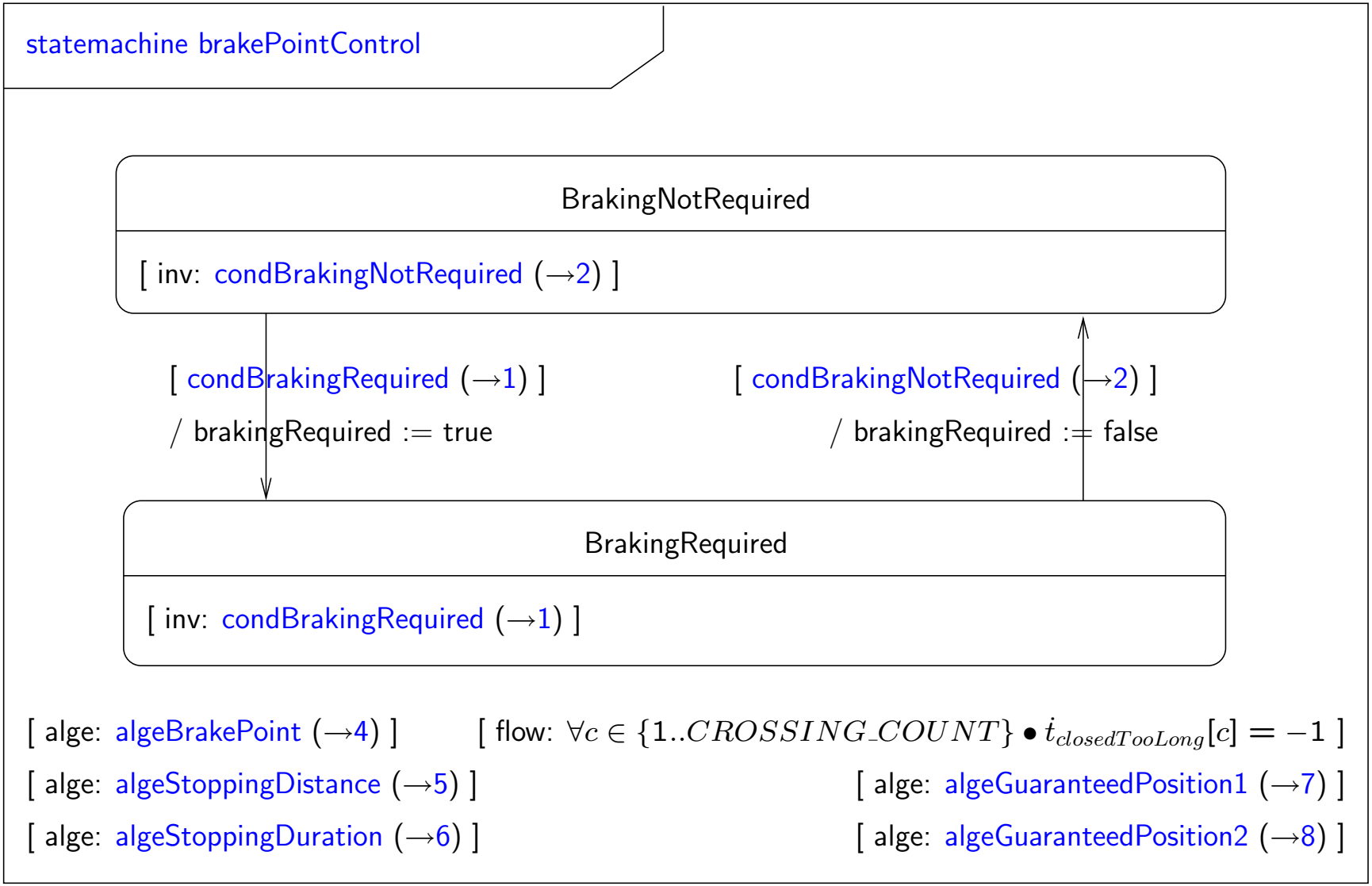


- private variables controlled continuously (here)

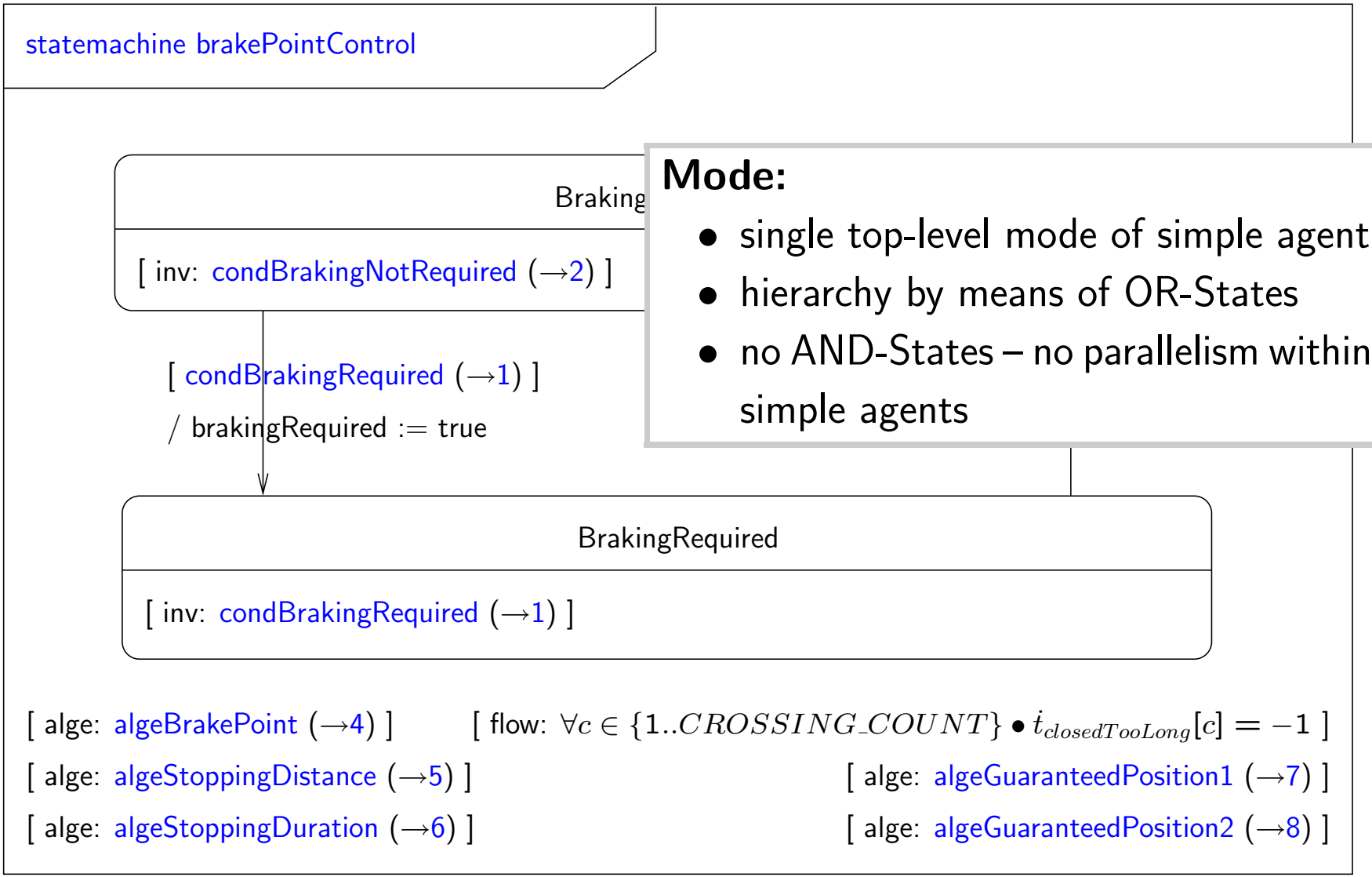
Class view of BrakePointController



Class view of BrakePointController



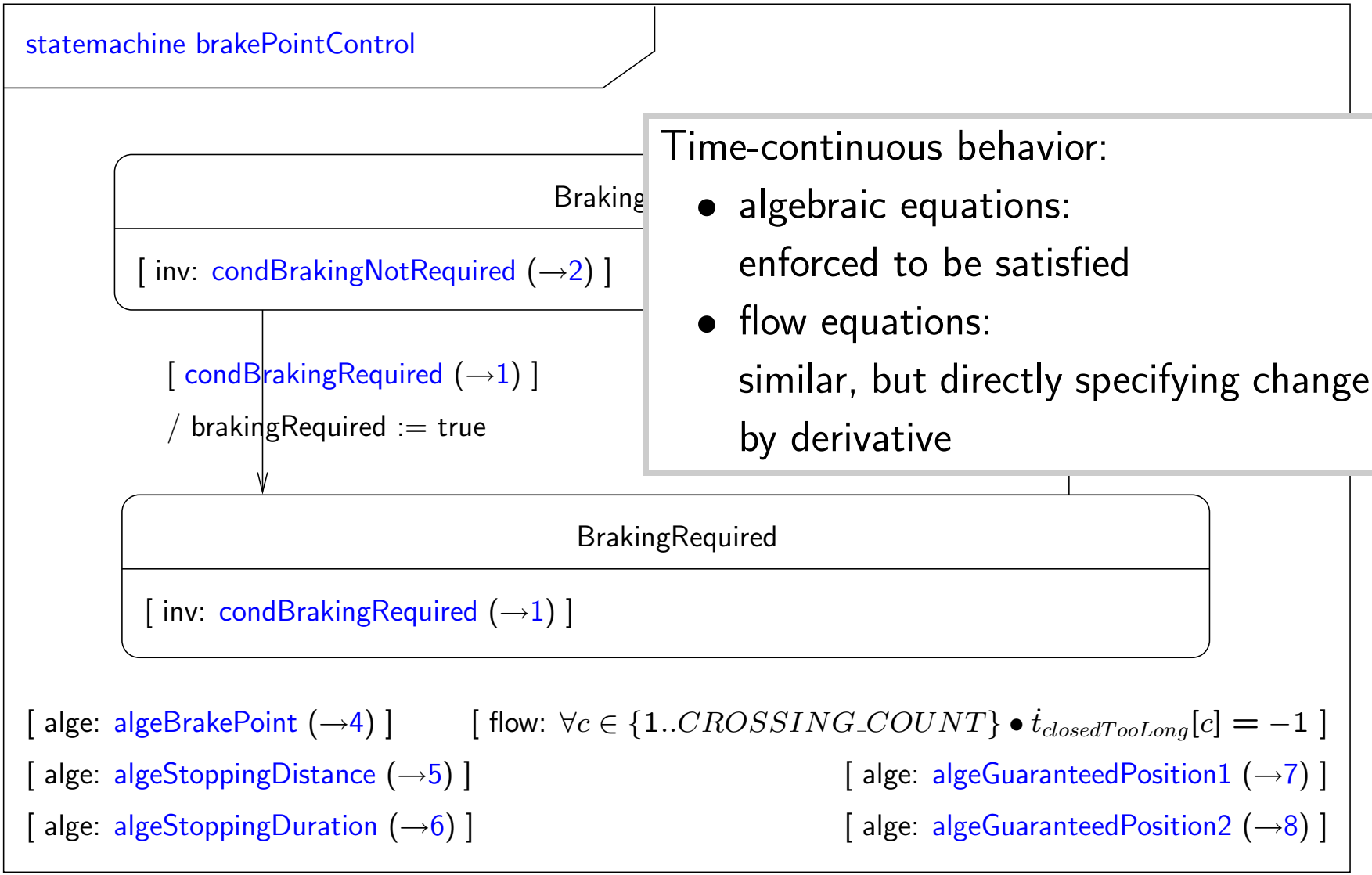
Behavior of BrakePointController



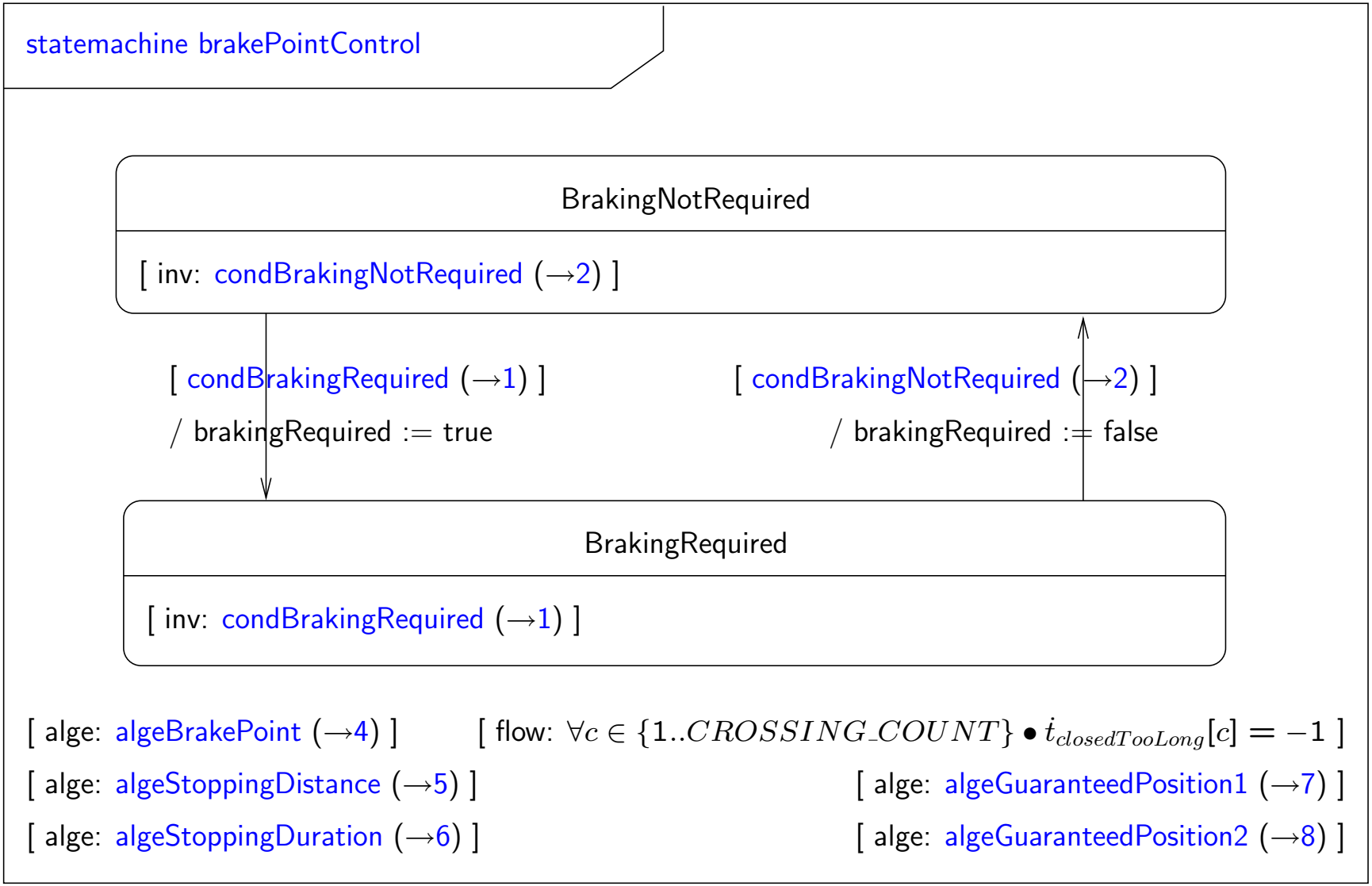
Mode:

- single top-level mode of simple agent
- hierarchy by means of OR-States
- no AND-States – no parallelism within simple agents

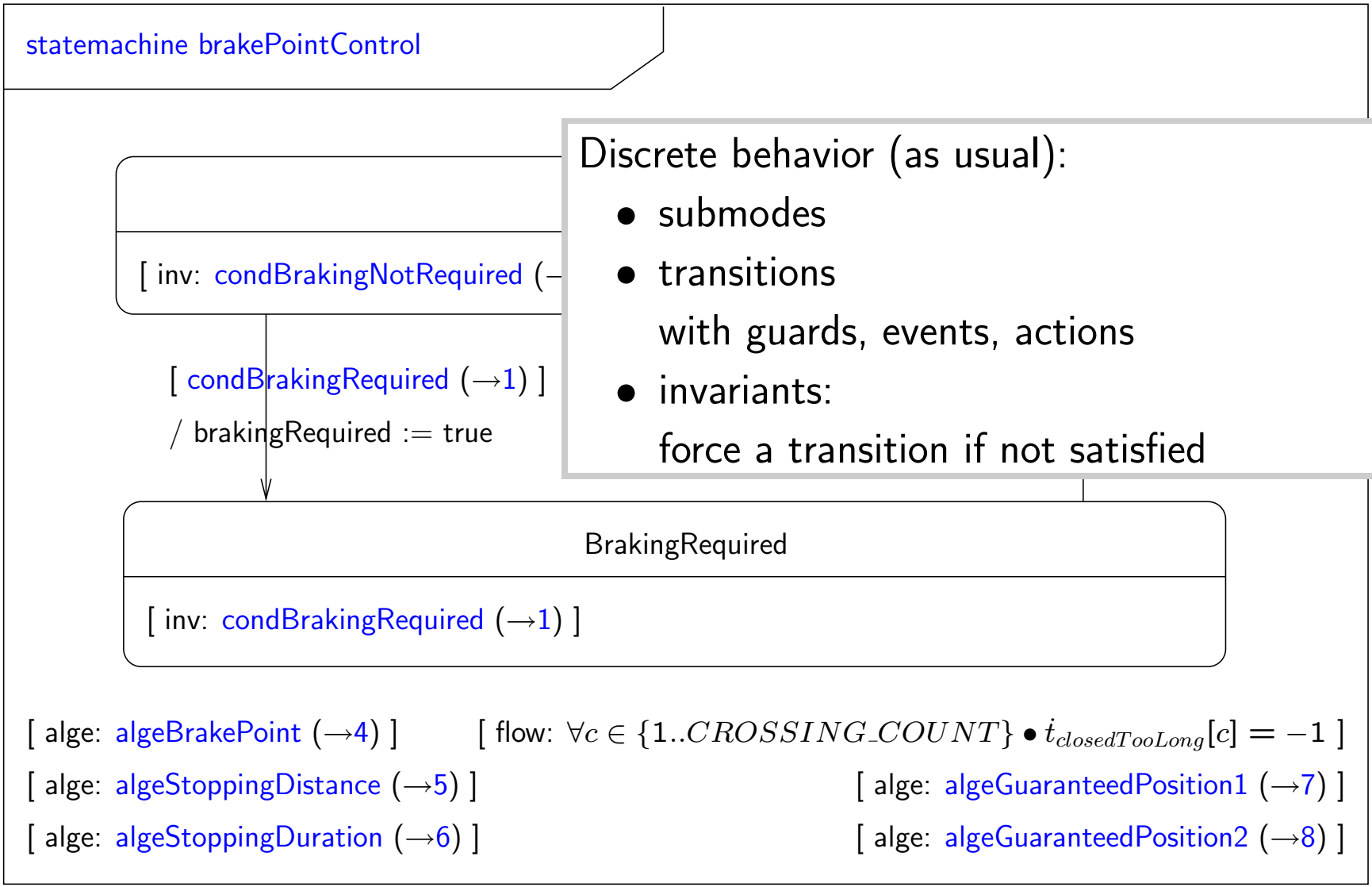
Behavior of BrakePointController



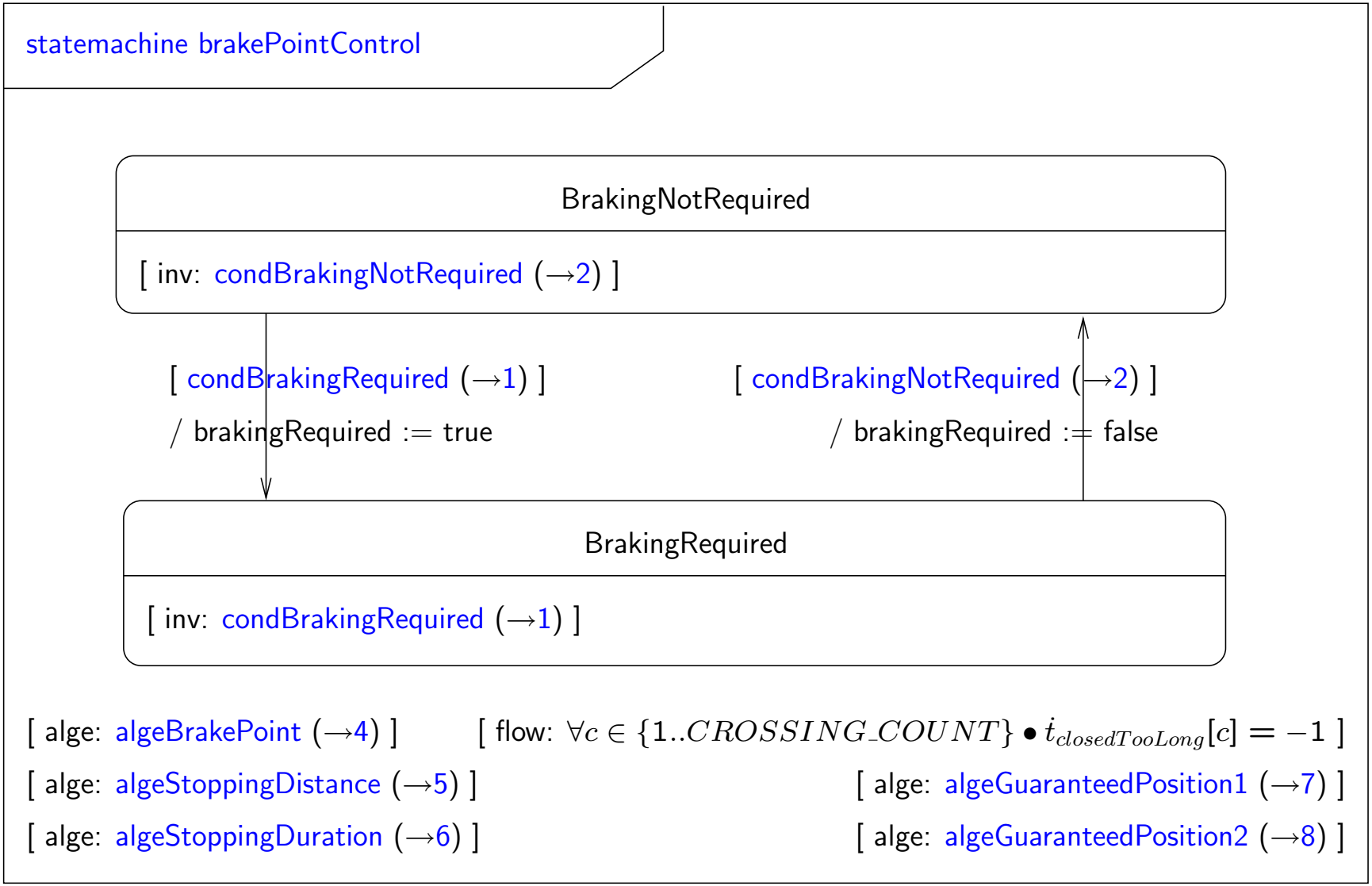
Behavior of BrakePointController



Behavior of BrakePointController



Behavior of BrakePointController



Behavior of BrakePointController

The end.

$$\textbf{condBrakingRequired} \equiv \tag{1}$$
$$\exists i \in \{1..VTP\_COUNT\} \bullet$$
$$brakePoint[i].x \leq x \wedge \text{ra}.vtp[i].v < v \wedge \text{ra}.vtp[i].x > x \wedge$$
$$(vtpActive[i] \vee \text{condTooLate}\ (\rightarrow 3))$$

$$\textbf{condBrakingNotRequired} \equiv \tag{2}$$
$$\forall i \in \{1..VTP\_COUNT\} \bullet$$
$$\neg(brakePoint[i].x \leq x \wedge \text{ra}.vtp[i].v < v \wedge \text{ra}.vtp[i].x > x \wedge$$
$$(vtpActive[i] \vee \text{condTooLate}\ (\rightarrow 3)))$$

$$\textbf{condTooLate} \equiv \tag{3}$$
$$\begin{aligned}
&\mathsf{ra}.vtp[i].type = VTP_TYPE.CROSSING \wedge \\
&((x_{closedTooLong,1}[\mathsf{ra}.vtp[i].cr.id] \leq \mathsf{ra}.vtp[i].cr.x_{end} + const.l \\
&\quad \wedge t_{closedTooLong}[\mathsf{ra}.vtp[i].cr.id] \leq t_{brake}) \\
&\quad \vee (x_{closedTooLong,2}[\mathsf{ra}.vtp[i].cr.id] \leq \mathsf{ra}.vtp[i].cr.x_{end} + const.l \\
&\quad \wedge t_{closedTooLong}[\mathsf{ra}.vtp[i].cr.id] > t_{brake}))
\end{aligned}$$

algeBrakePoint $\equiv$

$$\forall i \in \{1..VTP\_COUNT\} \bullet brakePoint[i] = \mathsf{ra}.vtp[i].x - \frac{\mathsf{ra}.vtp[i].v^2 - v^2}{2 \cdot const.a_{min}} \tag{4}$$

algeStoppingDistance $\equiv$ (5)

$$s_{brake} = \frac{-v^2}{2 \cdot const.a_{min}}$$

algeStoppingDuration $\equiv$ (6)

$$t_{brake} = \frac{-v}{const.a_{min}}$$

algeGuaranteedPosition1 $\equiv$ (7)

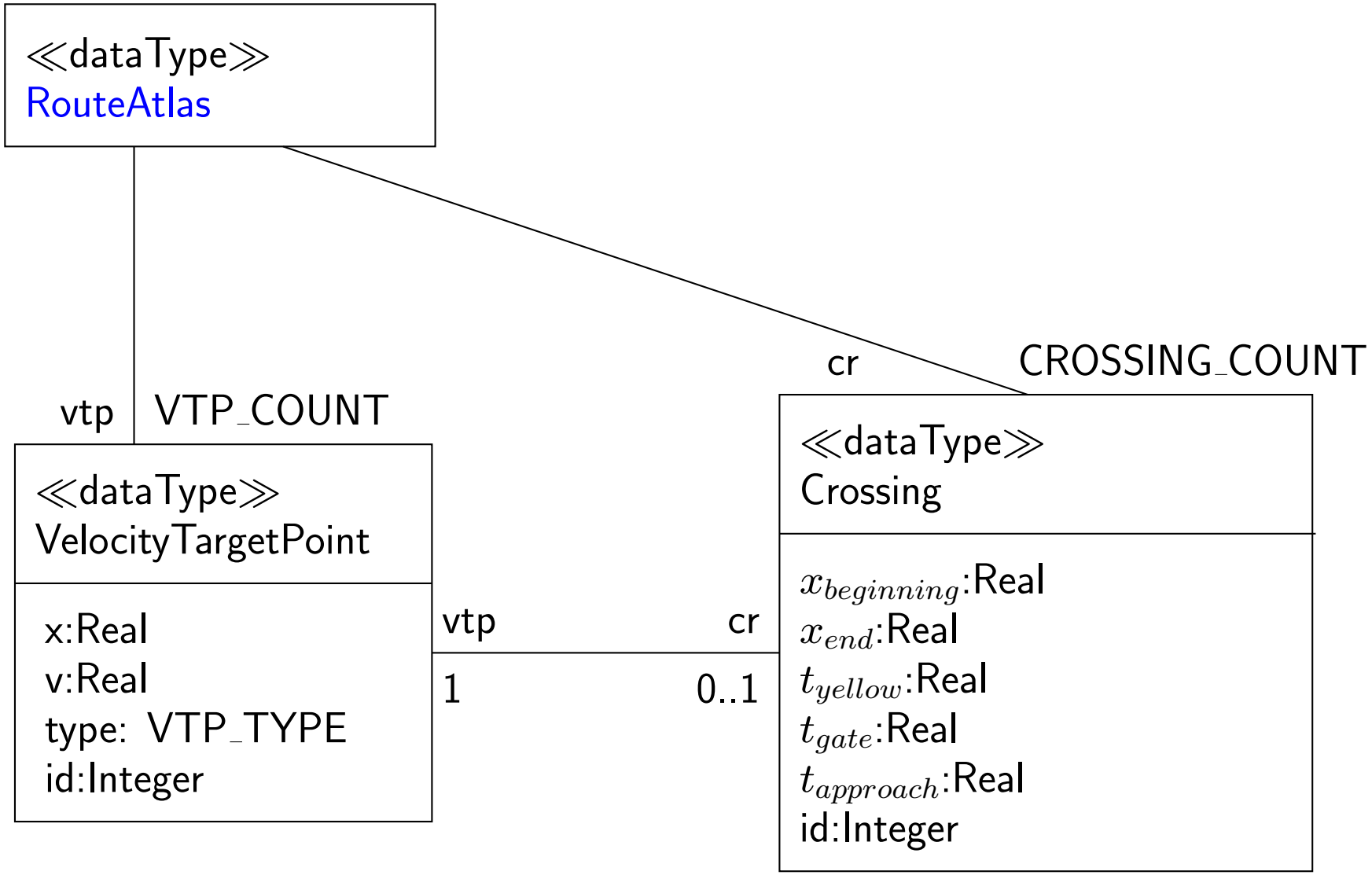
$\forall c \in \{1..CROSSING\_COUNT\} \bullet$

$x_{closedTooLong,1}[c] = x + \frac{const.a_{min}}{2} \cdot t_{closedTooLong}[c]^2 + v \cdot t_{closedTooLong}[c]$

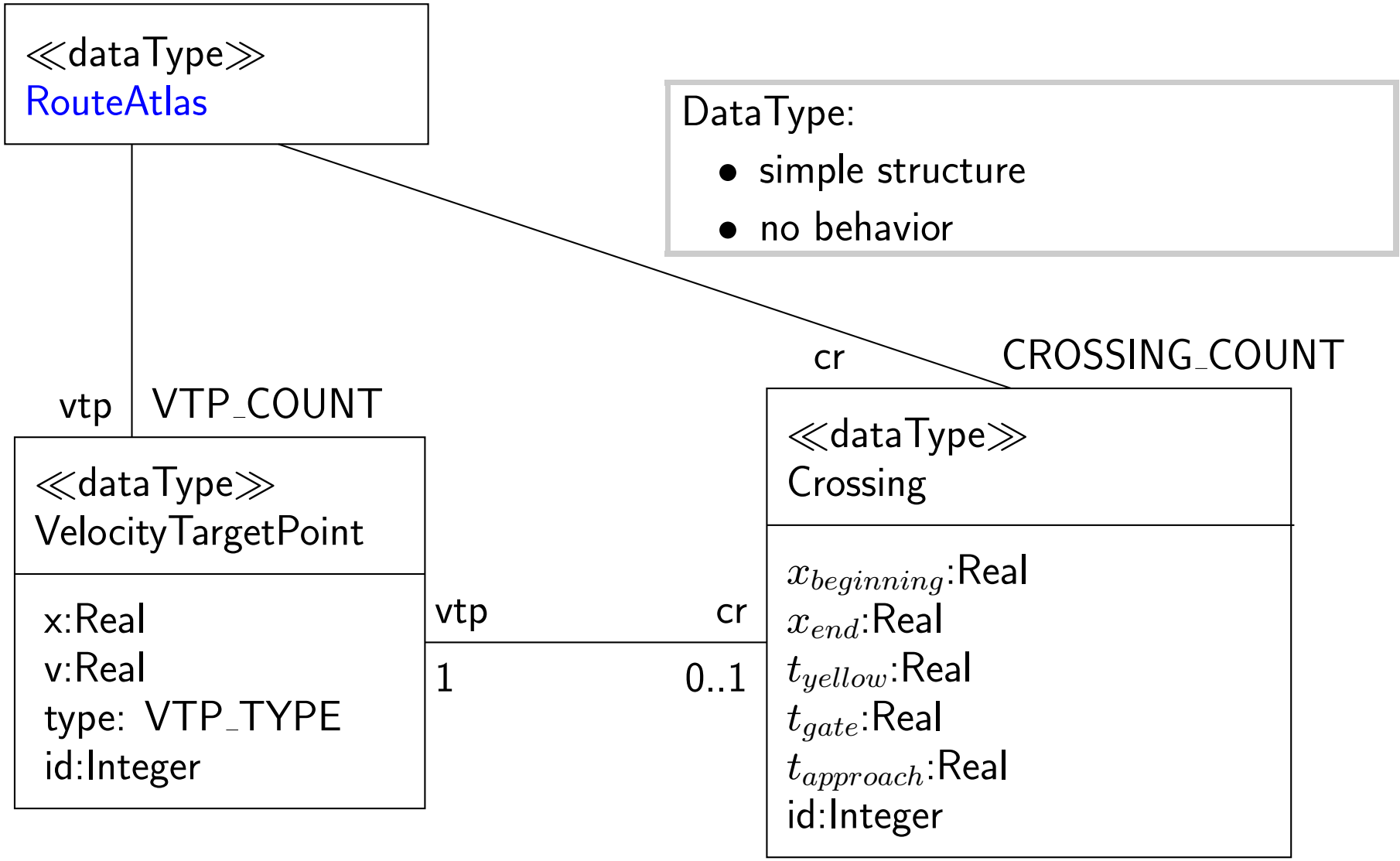
algeGuaranteedPosition2 $\equiv$ (8)

$\forall c \in \{1..CROSSING\_COUNT\} \bullet$

$x_{closedTooLong,2}[c] = x + s_{brake} + (t_{closedTooLong}[c] - t_{brake}) \cdot const.v_{pass}$


≪dataType≫
RouteAtlas
cr
CROSSING_COUNT
vtp
VTP_COUNT
≪dataType≫
VelocityTargetPoint
x:Real
v:Real
type: VTP_TYPE
id:Integer
vtp
cr
1
0..1
≪dataType≫
Crossing
$x_{beginning}$:Real
x_{end}:Real
t_{yellow}:Real
t_{gate}:Real
$t_{approach}$:Real
id:Integer

Datatypes


≪dataType≫
RouteAtlas
DataType:
• simple structure
• no behavior
cr
CROSSING_COUNT
vtp
VTP_COUNT
≪dataType≫
VelocityTargetPoint
x:Real
v:Real
type: VTP_TYPE
id:Integer
vtp
cr
1
0..1
≪dataType≫
Crossing
$x_{beginning}$:Real
x_{end}:Real
t_{yellow}:Real
t_{gate}:Real
$t_{approach}$:Real
id:Integer

Datatypes