Assignment 3 - 2AA4

Mohamed Bengezi bengezim

April 4, 2017

The contents of this report include the specifications of RegionT and PathCalculation modules, as well as a critique of the interface for the modules for Part 1 of Assignment 3.

Region Module

Template Module

RegionT

Uses

PointT, Constants

Syntax

Exported Types

RegionT = ?

Exported Access Programs

Routine name	In	Out	Exceptions
RegionT	PointT, real, real	RegionT	InvalidRegionException
pointInRegion	PointT	boolean	

Semantics

State Variables

lower_left: PointT *//coordinates of the lower left corner of the region*
width: real *//width of the rectangular region*
height: real *//height of the rectangular region*

State Invariant

None

Assumptions

The RegionT constructor is called for each abstract object before any other access routine is called for that object. The constructor can only be called once.

Access Routine Semantics

RegionT(p, w, h):

- transition: $lower\_left, width, height := p, w, h$
- output: $out := self$
- exception: $exc := (\neg(0 \leq p.xcrd() \leq \text{Contants.MAX\_X} - w) \vee (0 \leq p.ycrd() \leq \text{Constants.MAX\_Y} - h)) \Rightarrow \text{InvalidRegionException})$

pointInRegion(p):

- output: $out := \exists(p, q : PointT | q.ycrd() \in [lower\_left.ycrd()..height] \wedge q.xcrd() \in [lower\_left.xcrd()..width] : p.dist(q) < TOLERANCE)$
- exception: none

Path Calculation Module

Module

PathCalculation

Uses

Constants, PointT, RegionT, PathT, Obstacles, Destinations, SafeZone, Map

Syntax

Exported Access Programs

Routine name	In	Out	Exceptions
is_validSegment	PointT, PointT	boolean	
is_validPath	PathT	boolean	
is_shortestPath	PathT	boolean	
totalDistance	PathT	real	
totalTurns	PathT	integer	
estimatedTime	PathT	real	

Semantics

State Invariant

None

Assumptions

The RegionT constructor is called for each abstract object before any other access routine is called for that object. The constructor can only be called once.

Access Routine Semantics

is_validSegment(p_1, p_2):

- output: $out := \forall(p, q : PointT | (q.dist(p_1) + q.dist(p_2) = p_1.dist(p_2)) \wedge (p.dist(q) \leq$ Constants.TOLERANCE) : $\forall(i : \mathbb{N} | i \in [0..|Obs|-1] : \neg((Obstacles.getval(i)).pointInRegion(p))))$
- exception: None

is_validPath(p):

- output: $out := \exists(q : PointT \wedge i : \mathbb{N} | (q.dist(p.getval(i)) + q.dist(p.getval(i+1)) = p.getval(i).dist(Path.getval(i+1)))\forall(i, j, k : \mathbb{N} \wedge s : SafeZone \wedge d : Destinations | i \in [0..s.size() - 1] \wedge Maps.get\_safeZone.getval(i).pointInRegion(p.getval(0)) \wedge Maps.get\_safeZone.getval(i).pointInRegion(p.getval(p.size() - 1)) \wedge j \in [0..Maps.get\_destinations.size() - 1] \wedge k \in [1..p.size() - 1] \wedge Maps.get\_destinations.getval(j).pointInRegion(p.getval(0)) \wedge Maps.get\_destinations.getval(j).pointInRegion(p.getval(k)) : p.isvalidSegment(p.getval(k-1), p.getval(k))))$
- exception: None

is_shortestPath(p):

- output: $out := \forall(x : PathT | PathCalculation.is\_validPath(x) : PathCalculation.totalDistance(p) \leq PathCalculation.totalDistance(x))$
- exception: none

totalDistance(p):

- output: $out := +(i : \mathbb{N} | i \in [0..p.size()-1] \wedge i < p.size()-2 : p.getval(i).dist(p.getval(i+1)))$
- exception: None

totalTurns(p):

- output: $out := +(i : \mathbb{N} | i \in [0..|p| - 3] \wedge (p.getVal([i+1]).dist(p.getVal([i])) + p.getVal([i+2]).dist(p.getVal([i+1]))) \neq p.getVal([i+2]).dist(p.getVal([i])) : 1)$
- exception: none

estimatedTime(p):

- output: $out := (Constants.VECLOCITY\_LINEAR \times totalDistance(p)) + (+(i : \mathbb{N} | i \in [0..|p|-3] : Constants.VELOCITY\_ANGULAR \times pathAngle(p.getVal(i), p.getVal(i+1), p.getVal(i+2))))$
- exception: none

Local Functions

angle: PointT $\times$ PointT $\times$ PointT $\rightarrow$ real pathAngle(p, q, r):

- output: $out := \arccos(((r.ycrd() - q.ycrd()) \times (q_y crd() - p.ycrd()) + ((r.xcrd() - q.xcrd()) \times (q.xcrd() - p.xcrd())))/(r.dist(q) \times q.dist(p)))$
- exception: none

Point ADT Module

Template Module

PointT

Uses

Syntax

Exported Types

PointT = ?

Exported Constants

MAX_X = 10 MAX_Y = 10

Exported Access Programs

Routine name	In	Out	Exceptions
PointT	real, real	PointT	InvalidPointException
xcrd		real	
ycrd		real	
dist	PointT	real	

Semantics

State Variables

xc: real
yc: real

State Invariant

none

Assumptions

The constructor PointT is called for each abstract object before any other access routine is called for that object. The constructor cannot be called on an existing object.

Access Routine Semantics

PointT(x, y):

- transition: $xc, yc := x, y$
- output: $out := self$
- exception $exc := ((\neg(0 \leq x \leq \text{MAX\_X}) \vee \neg(0 \leq y \leq \text{MAX\_Y})) \Rightarrow \text{InvalidPointException})$

xcrd():

- output: $out := xc$
- exception: none

ycrd():

- output: $out := yc$
- exception: none

dist(p):

- output: $out := \sqrt{(self.xc - p.xc)^2 + (self.yc - p.yc)^2}$
- exception: none