```
MACHINE
  Robot_0
REFINES
  Generic
SEES
  Robot_0_Ctx
VARIABLES  t , pA , Target , Direction
INVARIANTS
  inv1 :  pA ∈ ℝ ⇸ S
  inv3 :  [0, t] ⊆ dom(pA)
  inv5 :  x_p = pA
  inv6 :  Target ∈ ℝ × ℝ
  inv7 :  Direction ∈ ℝ × ℝ
  inv8 :  DeltaNeighborhood(SpeedLimit, 0 ↦ 0, Direction)
  inv9 :  x_s = (Target ↦ Direction)
  inv10 :  plannar_distance(Target, 0 ↦ 0) ↦ minus(CriticalDistance ↦ CloseEnough) ∈ lt
  inv11 :  ∀t_ · t_ ∈ [0, t] ⇒ plannar_distance(pA(t_), 0 ↦ 0) ↦ CriticalDistance ∈ lt
EVENTS
  INITIALISATION
  WITH
    x_s' :  x_s' = Target' ↦ Direction'
    x_p' :  x_p' = pA'
  THEN
    act1 :  t := 0
    act2 :  pA := {0 ↦ (px0 ↦ py0)}
    act3 :  Target :| Target' ∈ ℝ × ℝ ∧ plannar_distance(Target', 0 ↦ 0) ↦ minus(CriticalDistance ↦ CloseEnough) ∈ lt
    act4 :  Direction := 0 ↦ 0
  END

  Behave
  REFINES  Behave
  ANY  e , tp
  WHERE
    grd1 :  e ∈ DE(S)
    grd2 :  Solvable([t, tp], e)
    grd3 :  plannar_distance(Target, pA(t)) ↦ CloseEnough ∈ gt
    grd4 :  tp ∈ ℝ+
    grd5 :  t < tp
    grd6 :
      CBAPsolutionOfFIS(t, tp, pA, e, {px_ ↦ py_ | plannar_distance(Target, px_ ↦ py_) ↦ CloseEnough ∈ gt
                                        ∧plannar_distance(0 ↦ 0, px_ ↦ py_) ↦ CriticalDistance ∈ lt})
  WITH
    x_p' :  x_p' = pA'
    Inv :  Inv = {px_ ↦ py_ | plannar_distance(Target, px_ ↦ py_) ↦ CloseEnough ∈ gt}
  THEN
    act1 :
      t, pA :|
        pA' ∈ ℝ ⇸ S ∧ t' = tp∧
        [0, t'] ⊆ dom(pA')∧
        CBAPsolutionOf(t, t', pA, pA', e, {px_ ↦ py_ | plannar_distance(Target, px_ ↦ py_) ↦ CloseEnough ∈ gt
                                           ∧plannar_distance(0 ↦ 0, px_ ↦ py_) ↦ CriticalDistance ∈ lt})
  END

  sense_close_enough
  REFINES  Sense
  ANY  next_direction , next_target
  WHERE
    grd1 :  next_direction ∈ ℝ × ℝ
    grd2 :  next_target ∈ ℝ × ℝ
    grd3 :  plannar_distance(Target, pA(t)) ↦ CloseEnough ∈ leq
    grd4 :  DeltaNeighborhood(SpeedLimit, 0 ↦ 0, next_direction)
    grd5 :  plannar_distance(next_target, 0 ↦ 0) ↦ minus(CriticalDistance ↦ CloseEnough) ∈ lt
  WITH
    s :  s = {next_target ↦ next_direction}
    p :  p = {Target ↦ Direction} × {t} × {px_ ↦ py_ | plannar_distance(Target, px_ ↦ py_) ↦ CloseEnough ∈ leq}
    x_s' :  x_s' = next_target ↦ next_direction
  THEN
    act1 :  Direction := next_direction
    act2 :  Target := next_target
  END

  transition_change_direction
  REFINES  Transition
```

```
ANY    new_direction
WHERE
    grd1 :  new_direction ∈ ℝ × ℝ
    grd2 :  DeltaNeighborhood(SpeedLimit, 0 ↦ 0, new_direction)
WITH
    s :  s = {Target} × {new_direction}
    x_s' :  x_s' = Target ↦ new_direction
THEN
    act1 :  Direction := new_direction
END

transition_change_target
REFINES  Transition
ANY    next_target
WHERE
    grd1 :  next_target ∈ ℝ × ℝ
    grd2 :  plannar_distance(next_target, 0 ↦ 0) ↦ minus(CriticalDistance ↦ CloseEnough) ∈ lt
WITH
    s :  s = {next_target} × {Direction}
    x_s' :  x_s' = next_target ↦ Direction
THEN
    act1 :  Target := next_target
END

actuate_movement
REFINES  Actuate
ANY    e , tp
WHERE
    grd1 :  e ∈ DE(S)
    grd2 :  Solvable([t, tp], e)
    grd3 :  plannar_distance(Target, pA(t)) ↦ CloseEnough ∈ gt
    grd7 :  tp ∈ ℝ+
    grd8 :  t < tp
    grd9 :
        CBAPsolutionOfFIS(t, tp, pA, e, {px_ ↦ py_ | plannar_distance(Target, px_ ↦ py_) ↦ CloseEnough ∈ gt
                                                  ∧plannar_distance(0 ↦ 0, px_ ↦ py_) ↦ CriticalDistance ∈ lt})
WITH
    x'_p :  x'_p = pA'
    Inv :  Inv = {px_ ↦ py_ | plannar_distance(Target, px_ ↦ py_) ↦ CloseEnough ∈ gt}
    s :  s = {Target} × {Direction}
THEN
    act1 :
        t, pA :|
            pA' ∈ ℝ ⇸ S ∧ t' = tp∧
            [0, t'] ⊆ dom(pA')∧
            CBAPsolutionOf(t, t', pA, pA', e, {px_ ↦ py_ | plannar_distance(Target, px_ ↦ py_) ↦ CloseEnough ∈ gt
                                                  ∧plannar_distance(0 ↦ 0, px_ ↦ py_) ↦ CriticalDistance ∈ lt})
END

END
```