Projet CPS 2019: Spécification formelle.

GERDAY Nathan

Ecran

Service: Screen

Observators:
const `Height`: [Screen] → int
const `Width`: [Screen] → int
`CellNature`: [Screen] × int × int → Cell
  pre `CellNature(S,x,y)` **requires** $0 \leq$ `y` $<$ `Height(S)` **and** $0 \leq$ `x` $<$ `Width(S)`

Constructors:
`init`: int × int → [Screen]
  pre `init(h,w)` **requires** $0 <$ `h` **and** $0 <$ `w`

Operators:
`Dig`: [Screen] × int × int → [Screen]
  pre `Dig(S,x,y)` **requires** `CellNature(S,x,y)` = **PLT**
`Fill`: [Screen] × int × int → [Screen]
  pre `Fill(S,x,y)` **requires** `CellNature(S,x,y)` = **HOL**
`OpenDoor`: [Screen] × int × int → [Screen]
  pre `OpenDoor(S,x,y)` **requires** `CellNature(S,x,y)` = **DOR**
`RevealTrap`: [Screen] × int × int → [Screen]
  pre `RevealTrap(S,x,y)` **requires** `CellNature(S,x,y)` = **TRP**

Observations:
[init]: `Height(init(h,w))` = `h`
`Width(init(h,w))` = `w`
forall (x, y) **in** `[0;Width(S)[`× `[0;Height(S)[`, `CellNature(init(h,w),x,y)` = **EMP**
[Dig]: `CellNature(Dig(S,x,y)),x,y` = **HOL**
forall (x, y) **in** `[0;Width(S)[` × `[0;Height(S)[`,
  (`x` ≠ `u` **or** `y` ≠ `v`) **implies** `CellNature(Dig(S,u,v)),x,y` = `CellNature(x,y)`
[Fill]: `CellNature(Fill(S,x,y),x,y)` = **PLT**
forall (x, y) **in** `[0;Width(S)[`× `[0;Height(S)[`,
  (`x` ≠ `u` **or** `y` ≠ `v`) **implies** `CellNature(Fill(S,u,v)),x,y` = `CellNature(x,y)`
[OpenDoor]: `CellNature(Fill(S,x,y),x,y)` = **EMP**
forall (x, y) **in** `[0;Width(S)[`× `[0;Height(S)[`,
  (`x` ≠ `u` **or** `y` ≠ `v`) **implies** `CellNature(OpenDoor(S,u,v)),x,y` = `CellNature(x,y)`
[RevealTrap]: `CellNature(Fill(S,x,y),x,y)` = **EMP**
forall (x, y) **in** `[0;Width(S)[`× `[0;Height(S)[`,
  (`x` ≠ `u` **or** `y` ≠ `v`) **implies** `CellNature(RevealTrap(S,u,v)),x,y` = `CellNature(x,y)`

Ecran éditable

Pas de changement par rapport à la spécification fournie

Service: EditableScreen **includes** Screen
Observators: `Playable`: [EditableScreen] → bool
Operators: `SetNature`: [EditableScreen] × int × int × Cell → [EditableScreen]
pre `SetNature(S,x,y,C)` **requires** $0 \leq$ `y` $<$ `Height(S)` **and** $0 \leq$ `x` $<$ `Width(S)`
Observations:
[invariant]: `Playable(S)` **min**
forall (x, y) **in** `[0;Width(S)[`× `[0;Height(S)[`, `CellNature(S,x,y)` ≠ **HOL**
and forall x **in** `[0;Width(S)[`, `CellNature(S,x,0)` = **MTL**
[SetNature]: `CellNature(SetNature(S,x,y,C)),x,y` = C
forall (x, y) **in** `[0;Width(S)[`× `[0;Height(S)[`,
(`x` ≠ `u` **or** `y` ≠ `v`) **implies** `CellNature(SetNature(S,u,v,C)),x,y)` = `CellNature(S,x,y)`

Environnement

Service: Environment **includes** Screen
Observators : `CellContent`: int × int → Set{Character + Item}
pre `CellContent(E,x,y)` **requires** $0 \leq$ `y` $<$ `Height(S)` **and** $0 \leq$ `x` $<$ `Width(S)`
Constructors: `init`: EditableScreen → Environment
Operators: `AddToCellContent`: [EditableScreen] × int × int × {Character + Item}
→ [EditableScreen]
pre `AddToCellContent(S,x,y,e)` **requires** $0 \leq$ `y` $<$ `Height(S)`
and $0 \leq$ `x` $<$ `Width(S)`
and exists Guard g **in** `CellContent(E,x,y)` **implies not** e **is** Guard
`RemoveFromCellContent`: [EditableScreen] × int × int × {Character + Item}
→ [EditableScreen]
pre `RemoveFromCellContent(S,x,y,e)` **requires** $0 \leq$ `y` $<$ `Height(S)`
and $0 \leq$ `x` $<$ `Width(S)` **and exists** e **in** `CellContent(E,x,y)`
Observations:
[invariant]: **forall** (x, y) **in** `[0;Width(E)[`× `[0;Height(E)[`,
`CellNature(E,x,y)` **in** {**MTL**, **PLT**} **implies** `CellContent(x,y)` = ∅
[init]: **forall** (x, y) **in** `[0;Width(E)[`× `[0;Height(E)[`,
`CellNature(init(S),x,y)` = EditableScreen::`CellNature(S,x,y)`
and `CellContent(init(S),x,y)` = {}
[AddToCellContent]: CellContent(AddToCellContent(S, x, y, e), x, y) = CellContent(S, x, y) **union** {e}
forall (x,y) in `[0;Width(E)[ [0;Height(E)[`,
(`x` ≠ `u` **or** `y` ≠ `v`)
implies `CellNature(AddToCellContent(S,u,v,e)),x,y)` = `CellNature(S,x,y)`
and CellContent(RemoveFromCellContent(S, u, v, e), x, y) = CellContent(S, x, y)
[RemoveFromCellContent]: CellContent(RemoveFromCellContent(S, x, y, e), x, y) = CellContent(S, x, y) \{e}
forall (x,y) in `[0;Width(E)[ [0;Height(E)[`,
(`x` ≠ `u` **or** `y` ≠ `v`)
implies `CellNature(RemoveFromCellContent(S,u,v,e)),x,y)`
= `CellNature(S,x,y)`
and CellContent(RemoveFromCellContent(S, u, v, e), x, y) = CellContent(S, x, y)

Personnage

Service: Character

Observators: **const** `Envi`: [Character] → Environment
`Hgt`: [Character] → int
`Col`: [Character] → int

Constructors: `init`: Environment × int × int → [Character]
pre init(E,x,y) **requires** E ≠ null **and not** Environment::CellNature(E,x,y) **in** {**MTL**, **PLT**, **DOR**}
and 0 ≤ `y` < `Environment::Height(E)` **and** 0 ≤ `x` < `Environment::Width(E)`

Operators: `GoLeft`: [Character] → [Character]
`GoRight`: [Character] → [Character]
`GoUp`: [Character] → [Character]
`GoDown`: [Character] → [Character]

Observations:

`[invariant]`: Environment::CellNature(Envi(C),Col(C),Hgt(C)) **in** {**EMP**, **HOL**, **LAD**, **HDR**, **NPL**, **NGU**}

`[init]`: Hgt(init(E,x,y)) = y
Col(init(E,x,y)) = x
Envi(init(E,x,y)) = e
exists init(E,x,y) **in** Environment::CellContent(Envi(init(E,x,y), x, y))

`[GoLeft]`: Hgt(GoLeft(C)) = Hgt(C)
Col(C) = 0 **implies** Col(GoLeft(C)) = Col(C)
Environment::CellNature(Envi(C),Col(C)-1,Hgt(C)) **in** {**MTL**, **PLT**, **DOR**}
implies Col(GoLeft(C)) = Col(C)
Environment::CellNature(Envi(C),Col(C),Hgt(C)) **not in** {**LAD**, **HDR**}
and Environment::CellNature(Envi(C),Col(C),Hgt(C)-1) **not in** {**PLT**, **MTL**, **LAD**,**DOR**}
and not exists Guard g **in** Environment::CellContent(Envi(C),Col(C),Hgt(C)-1)
implies Col(GoLeft(C)) = Col(C)
exists Guard g **in** Environment::CellContent(Envi(C),Col(C)-1,Hgt(C))
implies Col(GoLeft(C)) = Col(C)
(Col(C) ≠ 0) **and** Environment::CellNature(Envi(C),Col(C)-1,Hgt(C)) **not in** {**MTL**, **PLT**,**DOR**}
and (Environment::CellNature(Envi(C),Col(C),Hgt(C)) **in** {**LAD**, **HDR**}
or Environment::CellNature(Envi(C),Col(C),Hgt(C)-1) **in** {**PLT**, **MTL**, **LAD**,**DOR**}
or exists Guard g **in** Environment::CellContent(Envi(C),Col(C),Hgt(C)-1))
and not (**exists** Guard g **in** Environment::CellContent(Envi(C),Col(C)-1,Hgt(C)))
implies Col(GoLeft(C)) = Col(C)-1

`[GoRight]`: Hgt(GoRight(C)) = Hgt(C)
Col(C) = Environment::Width(Envi(GoRight(C))) - 1 **implies** Col(GoRight(C)) = Col(C)
Environment::CellNature(Envi(C),Col(C)+1,Hgt(C)) **in** {**MTL**, **PLT**, **DOR**}
implies Col(GoRight(C)) = Col(C)
Environment::CellNature(Envi(C),Col(C),Hgt(C)) **not in** {**LAD**, **HDR**}
and Environment::CellNature(Envi(C),Col(C),Hgt(C)-1) **not in** {**PLT**, **MTL**, **LAD**,**DOR**}
and not exists Guard g **in** Environment::CellContent(Envi(C),Col(C),Hgt(C)-1)
implies Col(GoRight(C)) = Col(C)
exists Guard g **in** Environment::CellContent(Envi(C),Col(C)+1,Hgt(C))
implies Col(GoRight(C)) = Col(C)
(Col(C) ≠ Environment::Width(Envi(GoRight(C))) - 1)
and Environment::CellNature(Envi(C),Col(C)+1,Hgt(C)) **not in** {**MTL**,**PLT**,**DOR**}
and (Environment::CellNature(Envi(C),Col(C),Hgt(C)) **in** {**LAD**, **HDR**}
or Environment::CellNature(Envi(C),Col(C),Hgt(C)-1) **in** {**PLT**, **MTL**, **LAD**,**DOR**}
or exists Guard g **in** Environment::CellContent(Envi(C),Col(C),Hgt(C)-1))
and not (**exists** Guard g **in** Environment::CellContent(Envi(C),Col(C)+1,Hgt(C)))
implies Col(GoRight(C)) = Col(C)+1

`[GoUp]`: Col(GoUp(C)) = Col(C)
Hgt(C) = Environment::Height(Envi(GoUp(C))) - 1 **implies** Hgt(GoUp(C)) = Hgt(C)
Environment::CellNature(Envi(C),Col(C),Hgt(C)+1) **in** {**MTL**, **PLT**, **DOR**}
implies Hgt(GoUp(C)) = Hgt(C)
Environment::CellNature(Envi(C),Col(C),Hgt(C)) ≠ **LAD implies** Hgt(GoUp(C)) = Hgt(C)
exists Guard g **in** Environment::CellContent(Envi(C),Col(C),Hgt(C)+1)
implies Hgt(GoUp(C)) = Hgt(C)
(Hgt(C) ≠ Environment::Height(Envi(GoUp(C))) - 1)
and Environment::CellNature(Envi(C),Col(C),Hgt(C)+1) **not in** {**MTL**, **PLT**, **DOR**}
and Environment::CellNature(Envi(C),Col(C),Hgt(C)) = **LAD**
and exists Guard g in Environment::CellContent(Envi(C),Col(C),Hgt(C)+1)
implies Hgt(GoUp(C)) = Hgt(C)+1

`[GoDown]`: Col(GoDown(C)) = Col(C)
Hgt(C) = 0 **implies** Hgt(GoDown(C)) = Hgt(C)
Environment::CellNature(Envi(C),Col(C),Hgt(C)-1) **in** {**MTL**, **PLT**, **DOR**}
implies Hgt(GoDown(C)) = Hgt(C)
exists Guard g **in** Environment::CellContent(Envi(C),Col(C),Hgt(C)-1)
implies Hgt(GoDown(C)) = Hgt(C)
(Hgt(C) ≠ 0)
and Environment::CellNature(Envi(C),Col(C),Hgt(C)-1) **not in** {**MTL**, **PLT**, **DOR**}
and exists Guard g in Environment::CellContent(Envi(C),Col(C),Hgt(C)-1)
implies Hgt(GoDown(C)) = Hgt(C)-1

Joueur

Service: Player **includes** Character

Observators:
const `Engine`: [Player] → Engine
`FacingRight`: [Player] → bool
`CurrentlyHeldItem`: [Player] → Item
`NumberOfUsagesLeftForCurrentItem`: [Player] → int

Constructors:
`init`: int × int × Engine → [Player]
pre init(x,y,E) **requires** E ≠ null
and not Environment::CellNature(Engine::Environment(E),x,y) **in** {**MTL**, **PLT**, **DOR**, **NPL**}
and 0 ≤ `y` < `Environment::Height(Engine::Environment(E))`
and 0 ≤ `x` < `Environment::Width(Engine::Environment(E))`

Operators:
`DigLeft`: [Player] → [Player]
`DigRight`: [Player] → [Player]
`UseItem`: [Player] → [Player]
`PickupItem`: [Player] × ItemType → [Player]
`Step`: [Player] → [Player]

Observations:

`[invariants]`: Environment::CellNature(Envi(C),Col(C),Hgt(C)) **in** {**EMP**, **HOL**, **LAD**, **HDR**, **NGU**}

`[init]`:
Hgt(init(x,y,E)) = y
Col(init(x,y,E)) = x
Envi(init(x,y,E)) = Engine::Environment(E)
exists init(x,y,E) Environment::CellContent(Envi(init(x,y,E)), x, y)
Engine(init(x,y,E)) = E
FacingRight(init(x,y,E))
Item::Nature(CurrentlyHeldItem(init(x,y,E))) = Sword
NumberOfUsagesLeftForCurrentItem(init(x,y,E)) = 3

`[DigLeft]`: **NoCellNatureChanged(C) defined by**
(**forall** (i,j) in `[0;Environment::Width(Environment(DigLeft(C)))[`
`[0;Environment::Height(Environment(DigLeft(C)))[`,
Environment::CellNature(Envi(DigLeft(C), i, j)) = Environment::CellNature(Envi(C, i, j))
Hgt(DigLeft(C)) = Hgt(C)
Col(DigLeft(C)) = Col(C)
Col(C) = 0 **implies** NoCellNatureChanged
Environment::CellNature(Envi(C), Col(C), Hgt(C)-1) **not in** {**MTL**, **PLT**, **LAD**, **DOR**, **NPL**}
and not exists Guard g **in** Environment::CellContent(Envi(C), Col(C), Hgt(C)-1)
implies NoCellNatureChanged
Environment::CellNature(Envi(C), Col(C)-1, Hgt(C)) **not in** {**EMP**, **HOL**, **LAD**, **HDR**}
implies NoCellNatureChanged
Environment::CellNature(Envi(C), Col(C)-1, Hgt(C)-1) ≠ PLT
implies NoCellNatureChanged
Col(C) ≠ 0
and (Environment::CellNature(Envi(C), Col(C), Hgt(C)-1) **in** {**MTL**, **PLT**, **LAD**, **DOR**, **NPL**}
or exists Guard g in Environment::CellContent(Envi(C), Col(C), Hgt(C)-1))
and Environment::CellNature(Envi(C), Col(C)-1, Hgt(C)) **in** {**EMP**, **HOL**, **LAD**, **HDR**}
and Environment::CellNature(Envi(C), Col(C)-1, Hgt(C)-1) = PLT
implies Environment::CellNature(Envi(DigLeft(C)), Col(C)-1, Hgt(C)-1) = **HOL**
and forall (i,j) in `[0;Environment::Width(Environment(DigLeft(C)))[`
`[0;Environment::Height(Environment(DigLeft(C)))[`,
((i ≠ Col(DigLeft(C))-1) **or** (j ≠ Hgt(DigLeft(C))-1) **implies**
Environment::CellNature(Envi(DigLeft(C), i, j)) = Environment::CellNature(Envi(C, i, j)
FacingRight(DigLeft(C))
CurrentlyHeldItem(DigLeft(C)) = CurrentlyHeldItem(C)
NumberOfUsagesLeftForCurrentItem(DigLeft(C)) = NumberOfUsagesLeftForCurrentItem(C)

`[DigRight]`: **NoCellNatureChanged**`(C)` **defined by**

(**forall** (i,j) in `[0;Environment::Width(Environment(DigRight(C)))[`
`[0;Environment::Height(Environment(DigRight(C)))[`,
Environment::CellNature(Envi(DigRight(C), i, j)) = Environment::CellNature(Envi(C, i, j))

Hgt(DigRight(C)) = Hgt(C)
Col(DigRight(C)) = Col(C)
Col(C) = Environment::Width(Envi(DigRight(C))) - 1 **implies** NoCellNatureChanged
Environment::CellNature(Envi(C), Col(C), Hgt(C)-1) **not in** {**MTL**, **PLT**, **LAD**, **DOR**, **NPL**}
and not exists Guard g **in** Environment::CellContent(Envi(C), Col(C), Hgt(C)-1)
implies NoCellNatureChanged
Environment::CellNature(Envi(C), Col(C)+1, Hgt(C)) **not in** {**EMP**, **HOL**, **LAD**, **HDR**}
implies NoCellNatureChanged
Environment::CellNature(Envi(C), Col(C)+1, Hgt(C)-1) ≠ PLT
implies NoCellNatureChanged
Col(C) ≠ Environment::Width(Envi(DigRight(C))) - 1
and (Environment::CellNature(Envi(C), Col(C), Hgt(C)-1) **in** {**MTL**, **PLT**, **LAD**, **DOR**, **NPL**}
or exists Guard g in Environment::CellContent(Envi(C), Col(C), Hgt(C)-1))
and Environment::CellNature(Envi(C), Col(C)+1, Hgt(C)) **in** {**EMP**, **HOL**, **LAD**, **HDR**}
and Environment::CellNature(Envi(C), Col(C)+1, Hgt(C)-1) = PLT
implies Environment::CellNature(Envi(DigRight(C)), Col(C)+1, Hgt(C)-1) = **HOL**
and forall (i,j) in `[0;Environment::Width(Environment(DigRight(C)))[`
`[0;Environment::Height(Environment(DigRight(C)))[`,
((i ≠ Col(DigRight(C))+1) **or** (j ≠ Hgt(DigRight(C))-1) **implies**
Environment::CellNature(Envi(DigRight(C), i, j)) = Environment::CellNature(Envi(C, i, j)
not FacingRight(DigLeft(C))
CurrentlyHeldItem(DigLeft(C)) = CurrentlyHeldItem(C)
NumberOfUsagesLeftForCurrentItem(DigLeft(C)) = NumberOfUsagesLeftForCurrentItem(C)

`[UseItem]`: **CanUseItem(C) defined by**
CurrentlyHeldItem(C) ≠ null
and NumberOfUsagesLeftForCurrentItem(C) ≥ 1
CurrentlyHeldItem(C) ≠ null **and** NumberOfUsagesLeftForCurrentItem(C) = 1
implies CurrentlyHeldItem(UseItem('C)) = null
and NumberOfUsagesLeftForCurrentItem(UseItem(C)) = 0
CurrentlyHeldItem(C) ≠ null **and** NumberOfUsagesLeftForCurrentItem(C) > 1
implies CurrentlyHeldItem(UseItem(C)) = CurrentlyHeldItem(C)
and NumberOfUsagesLeftForCurrentItem(UseItem(C)) = NumberOfUsagesLeftForCurrentItem(C) - 1
CanUseItem **and** Item::Nature(CurrentlyHeldItem(C)) = Key **and** FacingRight(C)
and Col(C) < Environment::Width(Envi(C)) - 1
and Environment::CellNature(Envi(C), Col(C)+1, Hgt(C)) = DOR
implies Environment::CellNature(Envi(UseItem(C), Col(C)+1, Hgt(C)) = EMP
CanUseItem **and** Item::Nature(CurrentlyHeldItem(C)) = Key **and not** FacingRight(C)
and Col(C) > 0
and Environment::CellNature(Envi(C), Col(C)-1, Hgt(C)) = DOR
implies Environment::CellNature(Envi(UseItem(C), Col(C)-1, Hgt(C)+1) = EMP
forall Guard g **in** Engine::Guards(Engi(UseItem(C))),
CanUseItem **and** Item::Nature(CurrentlyHeldItem(C)) = Flash
implies Guard::TimeLeftParalyzed(g) = 10
CanUseItem **and** Item::Nature(CurrentlyHeldItem(C)) = Sword **and** Col(C) - 2 ≥ 0
and exists Guard g **in** Environment::CellContent(Envi(C), Col(C) - 2, Hgt(C))
implies exists g **in** Environment::CellContrent(Envi(UseItem(C)),
Coord::X(Guard::InitCoords(g)), Coord::Y(Guard::InitCoords(g)))
CanUseItem **and** Item::Nature(CurrentlyHeldItem(C)) = Sword **and** Col(C) - 1 ≥ 0
and exists Guard g **in** Environment::CellContent(Envi(C), Col(C) - 1, Hgt(C))
implies exists g **in** Environment::CellContrent(Envi(UseItem(C)),
Coord::X(Guard::InitCoords(g)), Coord::Y(Guard::InitCoords(g)))
CanUseItem **and** Item::Nature(CurrentlyHeldItem(C)) = Sword
and Col(C) + 2 < Environment::Width(Envi(C))
and exists Guard g **in** Environment::CellContent(Envi(C), Col(C) + 2, Hgt(C))
implies exists g **in** Environment::CellContrent(Envi(UseItem(C)),
Coord::X(Guard::InitCoords(g)), Coord::Y(Guard::InitCoords(g)))
CanUseItem **and** Item::Nature(CurrentlyHeldItem(C)) = Sword
and Col(C) + 1 < Environment::Width(Envi(C))
and exists Guard g **in** Environment::CellContent(Envi(C), Col(C) + 1, Hgt(C))
implies exists g **in** Environment::CellContrent(Envi(UseItem(C)),
Coord::X(Guard::InitCoords(g)), Coord::Y(Guard::InitCoords(g)))

ExistsObstacleBetween(x1, x2, y) defined by
exists Cell c **in** (**union** (**forall** i **in** `[x1;x2]`, Environment::CellNature(Envi(C), i, y)))
with (c **in** {**MTL**, **PLT**, **DOR**, **NPL**})
forall i **in** `[0;Environment::Width(Envi(C))[`,
CanUseItem **and** Item::Nature(CurrentlyHeldItem(C)) = Gun
and FacingRight(C)
and i > Col(C)
and not ExistsObstacleBetween(Col(C), i, Hgt(C))
and exists Guard g**in** Environment::CellContent(Envi(C), i, Hgt(C))
implies exists g **in** Environment::CellContent(Envi(UseItem(C),
Coord::X(Guard::InitCoords(g)), Coord::Y(Guard::InitCoords(g)))
CanUseItem **and** Item::Nature(CurrentlyHeldItem(C)) = Gun
and not FacingRight(C)
and i < Col(C)
and not ExistsObstacleBetween(i, Col(C), Hgt(C))
and exists Guard g**in** Environment::CellContent(Envi(C), i, Hgt(C))
implies exists g **in** Environment::CellContent(Envi(UseItem(C),
Coord::X(Guard::InitCoords(g)), Coord::Y(Guard::InitCoords(g)))
Col(UseItem(C)) = Col(C)
Hgt(UseItem(C)) = Hgt(C)
FacingRight(UseItem(C)) = FacingRight(C)

`[PickupItem]`: FacingRight(PickupItem(C, t)) = FacingRight(C)
Col(PickupItem(C, t)) = Col(C)
Hgt(PickupItem(C, t)) = Hgt(C)
CurrentlyHeldItem(C) ≠ null **and** Item::Nature(CurrentlyHeldItem(C)) = t **and** t = Key
implies NumberOfUsagesLeftForCurrentItem(PickupItem(C, t)) =
NumberOfUsagesLeftForCurrentItem(C)+1
and CurrentlyHeldItem(PickupItem(C, t)) = CurrentlyHeldItem(C)
CurrentlyHeldItem(C) ≠ null **and** Item::Nature(CurrentlyHeldItem(C)) = t **and** t = Flash
implies NumberOfUsagesLeftForCurrentItem(PickupItem(C, t)) =
NumberOfUsagesLeftForCurrentItem(C)+1
and CurrentlyHeldItem(PickupItem(C, t)) = CurrentlyHeldItem(C)
CurrentlyHeldItem(C) ≠ null **and** Item::Nature(CurrentlyHeldItem(C)) = t **and** t = Gun
implies NumberOfUsagesLeftForCurrentItem(PickupItem(C, t)) =
NumberOfUsagesLeftForCurrentItem(C)+1
and CurrentlyHeldItem(PickupItem(C, t)) = CurrentlyHeldItem(C)
CurrentlyHeldItem(C) ≠ null **and** Item::Nature(CurrentlyHeldItem(C)) = t **and** t = Sword
implies NumberOfUsagesLeftForCurrentItem(PickupItem(C, t)) =
NumberOfUsagesLeftForCurrentItem(C)+3
and CurrentlyHeldItem(PickupItem(C, t)) = CurrentlyHeldItem(C)
not (CurrentlyHeldItem(C) ≠ null **and** Item::Nature(CurrentlyHeldItem(C)) = t) **and** t = Key
implies NumberOfUsagesLeftForCurrentItem(PickupItem(C, t)) = 1
Item::Nature(CurrentlyHeldItem(PickupItem(C, t))) = t
not (CurrentlyHeldItem(C) ≠ null **and** Item::Nature(CurrentlyHeldItem(C)) = t) **and** t = Flash
implies NumberOfUsagesLeftForCurrentItem(PickupItem(C, t)) = 1
Item::Nature(CurrentlyHeldItem(PickupItem(C, t))) = t
not (CurrentlyHeldItem(C) ≠ null **and** Item::Nature(CurrentlyHeldItem(C)) = t) **and** t = Gun
implies NumberOfUsagesLeftForCurrentItem(PickupItem(C, t)) = 1
Item::Nature(CurrentlyHeldItem(PickupItem(C, t))) = t
not (CurrentlyHeldItem(C) ≠ null **and** Item::Nature(CurrentlyHeldItem(C)) = t) **and** t = Sword
implies NumberOfUsagesLeftForCurrentItem(PickupItem(C, t)) = 3
Item::Nature(CurrentlyHeldItem(PickupItem(C, t))) = t

`[Step]`: **Falling(C) defined by**
(Environment::CellNature(Envi(C), Col(C), Hgt(C)) **not in** {**LAD**,**HDR**}
and Environment::CellNature(Envi(C), Col(C), Hgt(C)-1) **in** {**EMP**,**HDR**,**HOL**,**NGU**}
and not exists Guard g **in** Environment::CellContent(Envi(C), Col(C), Hgt(C)-1))
Falling **implies** Step(C) = GoDown(C)
not Falling **and** NextCommand(C) = MOVEL **implies** Step(C) = GoLeft(C)
not Falling **and** NextCommand(C) = MOVER **implies** Step(C) = GoRight(C)
not Falling **and** NextCommand(C) = MOVED **implies** Step(C) = GoDown(C)
not Falling **and** NextCommand(C) = MOVEU **implies** Step(C) = GoUp(C)
not Falling **and** NextCommand(C) = DIGL **implies** Step(C) = DigLeft(C)
not Falling **and** NextCommand(C) = DIGR **implies** Step(C) = DigRight(C)
not Falling **and** NextCommand(C) = USEITEM **implies** Step(C) = UseItem(C)
not Falling **and** NextCommand(C) = NONE **implies** Step(C) = C

Les fonctions de déplacements de Player sont quasiment les mêmes que celles de Character, on change seulement le fait qu'on ne puisse pas aller dans une case NPL

`[GoLeft]`: Hgt(GoLeft(C)) = Hgt(C)
Col(C) = 0 **implies** Col(GoLeft(C)) = Col(C)
Environment::CellNature(Envi(C),Col(C)-1,Hgt(C)) **in** {**MTL**, **PLT**, **DOR**, **NPL**}
implies Col(GoLeft(C)) = Col(C)
Environment::CellNature(Envi(C),Col(C),Hgt(C)) **not in** {**LAD**, **HDR**}
and Environment::CellNature(Envi(C),Col(C),Hgt(C)-1) **not in** {**PLT**, **MTL**, **LAD**,**DOR**, **NPL**}
and not exists Guard g **in** Environment::CellContent(Envi(C),Col(C),Hgt(C)-1)
implies Col(GoLeft(C)) = Col(C)
exists Guard g **in** Environment::CellContent(Envi(C),Col(C)-1,Hgt(C))
implies Col(GoLeft(C)) = Col(C)
(Col(C) ≠ 0) **and** Environment::CellNature(Envi(C),Col(C)-1,Hgt(C)) **not in** {**MTL**, **PLT**,**DOR**, **NPL**}
and (Environment::CellNature(Envi(C),Col(C),Hgt(C)) **in** {**LAD**, **HDR**}
or Environment::CellNature(Envi(C),Col(C),Hgt(C)-1) **in** {**PLT**, **MTL**, **LAD**,**DOR**, **NPL**}
or exists Guard g **in** Environment::CellContent(Envi(C),Col(C),Hgt(C)-1))
and not (**exists** Guard g **in** Environment::CellContent(Envi(C),Col(C)-1,Hgt(C)))
implies Col(GoLeft(C)) = Col(C)-1

`[GoRight]`: Hgt(GoRight(C)) = Hgt(C)
Col(C) = Environment::Width(Envi(GoRight(C))) - 1 **implies** Col(GoRight(C)) = Col(C)
Environment::CellNature(Envi(C),Col(C)+1,Hgt(C)) **in** {**MTL**, **PLT**, **DOR**, **NPL**}
implies Col(GoRight(C)) = Col(C)
Environment::CellNature(Envi(C),Col(C),Hgt(C)) **not in** {**LAD**, **HDR**}
and Environment::CellNature(Envi(C),Col(C),Hgt(C)-1) **not in** {**PLT**, **MTL**, **LAD**,**DOR**, **NPL**}
and not exists Guard g **in** Environment::CellContent(Envi(C),Col(C),Hgt(C)-1)
implies Col(GoRight(C)) = Col(C)
exists Guard g **in** Environment::CellContent(Envi(C),Col(C)+1,Hgt(C))
implies Col(GoRight(C)) = Col(C)
(Col(C) ≠ Environment::Width(Envi(GoRight(C))) - 1)
and Environment::CellNature(Envi(C),Col(C)+1,Hgt(C)) **not in** {**MTL**, **PLT**,**DOR**, **NPL**}
and (Environment::CellNature(Envi(C),Col(C),Hgt(C)) **in** {**LAD**, **HDR**}
or Environment::CellNature(Envi(C),Col(C),Hgt(C)-1) **in** {**PLT**, **MTL**, **LAD**,**DOR**, **NPL**}
or exists Guard g **in** Environment::CellContent(Envi(C),Col(C),Hgt(C)-1))
and not (**exists** Guard g **in** Environment::CellContent(Envi(C),Col(C)+1,Hgt(C)))
implies Col(GoRight(C)) = Col(C)+1

`[GoUp]`: Col(GoUp(C)) = Col(C)
Hgt(C) = Environment::Height(Envi(GoUp(C))) - 1 **implies** Hgt(GoUp(C)) = Hgt(C)
Environment::CellNature(Envi(C),Col(C),Hgt(C)+1) **in** {**MTL**, **PLT**, **DOR**, **NPL**}
implies Hgt(GoUp(C)) = Hgt(C)
Environment::CellNature(Envi(C),Col(C),Hgt(C)) ≠ **LAD implies** Hgt(GoUp(C)) = Hgt(C)
exists Guard g **in** Environment::CellContent(Envi(C),Col(C),Hgt(C)+1)
implies Hgt(GoUp(C)) = Hgt(C)
(Hgt(C) ≠ Environment::Height(Envi(GoUp(C))) - 1)
and Environment::CellNature(Envi(C),Col(C),Hgt(C)+1) **not in** {**MTL**, **PLT**, **DOR**, **NPL**}
and Environment::CellNature(Envi(C),Col(C),Hgt(C)) = **LAD**
and exists Guard g in Environment::CellContent(Envi(C),Col(C),Hgt(C)+1)
implies Hgt(GoUp(C)) = Hgt(C)+1

`[GoDown]`: Col(GoDown(C)) = Col(C)
Hgt(C) = 0 **implies** Hgt(GoDown(C)) = Hgt(C)
Environment::CellNature(Envi(C),Col(C),Hgt(C)-1) **in** {**MTL**, **PLT**, **DOR**, **NPL**}
implies Hgt(GoDown(C)) = Hgt(C)
exists Guard g **in** Environment::CellContent(Envi(C),Col(C),Hgt(C)-1)
implies Hgt(GoDown(C)) = Hgt(C)
(Hgt(C) ≠ 0)
and Environment::CellNature(Envi(C),Col(C),Hgt(C)-1) **not in** {**MTL**, **PLT**, **DOR**, **NPL**}
and exists Guard g in Environment::CellContent(Envi(C),Col(C),Hgt(C)-1)
implies Hgt(GoDown(C)) = Hgt(C)-1

Garde

Service: Guard **includes** Character

Observators:
const `Id`: [Guard] → int
const `Engine`: [Guard] → Engine
const `Target`: [Guard] → Character
const `InitCoords`: [Guard] → Coord
const `Nature`: [Guard] → GuardType
`Behaviour`: [Guard] → Command
`TimeInHole`: [Guard] → int
`IdCounter`: [Guard] → int
`CarryingTreasure`: [Guard] → bool
`TimeLeftParalyzed`: [Guard] → int

Constructors:
`init`: Engine × int × int × Character → [Guard]
pre init(E,x,y,t) **requires** E ≠ null
and not Environment::CellNature(Engine::Environment(E),x,y) **in** {**MTL**, **PLT**, **DOR**, **NGU**}
and 0 ≤ `y` < `Environment::Height(Engine::Environment(E))`
and 0 ≤ `x` < `Environment::Width(Engine::Environment(E))`

Operators:
`ClimbLeft`: [Guard] → [Guard]
pre `ClimbLeft(G)` **requires** `Environment::CellNature(Envi(G),Hgt(G),Col(G))` = **HOL**
`ClimbRight`: [Guard] → [Guard]
pre `ClimbRight(G)` **requires** `Environment::CellNature(Envi(G),Hgt(G),Col(G))` = **HOL**
`MoveToInitCoords`: [Guard] → [Guard]
`Paralyze`: [Guard] → [Guard]
`Step`: [Guard] → [Guard]

Observations:
[invariant]: Environment::CellNature(Envi(C),Col(C),Hgt(C)) **in** {**EMP**, **HOL**, **LAD**, **HDR**, **NPL**}
Environment::CellNature(Envi(G),Col(G),Hgt(G)) = **LAD**
and Hgt(G) < Character::Hgt(Target(G))
implies Behaviour(G) = **MOVEU**
Environment::CellNature(Envi(G),Col(G),Hgt(G)) = **LAD**
and Hgt(G) > Character::Hgt(Target(G))
implies Behaviour(G) = **MOVED**
Environment::CellNature(Envi(G), Col(G), Hgt(G)) = **HOL**
and Character::Col(Target(G)) = Col(G) **and** Character::Hgt(Target(G)) = Hgt(G)+1
and Environment::CellNature(Envi(G),Col(G)+1,Hgt(G)+1) **in** {**EMP,HDR,LAD,NPL**}
implies Behaviour(G) = **MOVER**
Environment::CellNature(Envi(G), Col(G), Hgt(G)) = **HOL**
and Character::Col(Target(G)) = Col(G) **and** Character::Hgt(Target(G)) = Hgt(G)+1
and Environment::CellNature(Envi(G),Col(G)+1,Hgt(G)+1) **not in** {**EMP,HDR,LAD,NPL**}
implies Behaviour(G) = **MOVEL**
(Environment::CellNature(Envi(G),Col(G),Hgt(G)) ≠ LAD **or** Character::Hgt(Target(G)) = Hgt(G))
and (Environment::CellNature(Envi(G),Col(G),Hgt(G)) **in** {**HOL,HDR,LAD**}
or Environment::CellNature(Envi(G),Col(G),Hgt(G)-1) **in** {**PLT,MTL,LAD,DOR,NGU,TRP**}
or exists Guard g **in** Environment::CellContent(Envi(G),Col(G),Hgt(G)-1))
and Character::Col(Target(G) < Col(G)
implies Behaviour(G) = **MOVEL**
(Environment::CellNature(Envi(G),Col(G),Hgt(G)) ≠ LAD **or** Character::Hgt(Target(G)) = Hgt(G))
and (Environment::CellNature(Envi(G),Col(G),Hgt(G)) **in** {**HOL,HDR,LAD**}
or Environment::CellNature(Envi(G),Col(G),Hgt(G)-1) **in** {**PLT,MTL,LAD,DOR,NGU,TRP**}
or exists Guard g **in** Environment::CellContent(Envi(G),Col(G),Hgt(G)-1))
and Character::Col(Target(G) > Col(G)
implies Behaviour(G) = **MOVER**

[init]: Engine(init(E,x,y,t)) = e
Nature(init(E,x,y,t)) = NORMAL
Coord::X(InitCoords(init(E,x,y,t))) = x
Coord::Y(InitCoords(init(E,x,y,t))) = y
Target(init(E,x,y,t)) = t
Id(init(E,x,y,t)) = IdCounter(init(E,x,y,t)) - 1
not CarryingTreasure(init(E,x,y,t))
TimeInHole(init(E,x,y,t)) = 0
TimeLeftParalyzed(init(E,x,y,t)) = 0

[ClimbLeft]: Col(C) = 0 **implies** Col(ClimbLeft(C)) = Col(C) **and** Hgt(ClimbLeft(C)) = Hgt(C)
Screen::CellNature(Envi(C),Col(C)-1,Hgt(C) +1) **in** {**MTL**, **PLT**, **DOR**, **NGU** }
implies Col(ClimbLeft(C)) = Col(C) **and** Hgt(ClimbLeft(C)) = Hgt(C)
exists Guard g **in** Environment::CellContent(Envi(C),Col(C)-1,Hgt(C)+1)
implies Col(ClimbLeft(C)) = Col(C) **and** Hgt(ClimbLeft(C)) = Hgt(C)
Col(C) ≠ 0 **and**
Screen::CellNature(Envi(C),Col(C)-1,Hgt(C)+1) **not in** {**MTL**, **PLT**, **DOR**, **NGU** }
and not exists Guard g **in** Environment::CellContent(Envi(C),Col(C)-1,Hgt(C)+1)
and exists Player p **in** Environment::CellContent(Envi(C),Col(C),Hgt(C)+1)
implies Col(ClimbLeft(C)) = Col(C) **and** Hgt(ClimbLeft(C)) = Hgt(C)+1
Col(C) ≠ 0 **and**
Screen::CellNature(Envi(C),Col(C)-1,Hgt(C)+1) **not in** {**MTL**, **PLT**, **DOR**, **NGU** }
and not exists Guard g **in** Environment::CellContent(Envi(C),Col(C)-1,Hgt(C)+1)
and not exists Player p **in** Environment::CellContent(Envi(C),Col(C),Hgt(C)+1)
implies Col(ClimbLeft(C)) = Col(C)-1 **and** Hgt(ClimbLeft(C)) = Hgt(C)+1

[ClimbRight]: Col(C) = Environment::Width(Envi(GoRight(C))) - 1
implies Col(ClimbRight(C)) = Col(C) **and** Hgt(ClimbRight(C)) = Hgt(C)
Screen::CellNature(Envi(C),Col(C)+1,Hgt(C) +1) **in** {**MTL**, **PLT**, **DOR**, **NGU** }
implies Col(ClimbRight(C)) = Col(C) **and** Hgt(ClimbRight(C)) = Hgt(C)
exists Guard g **in** Environment::CellContent(Envi(C),Col(C)+1,Hgt(C)+1)
implies Col(ClimbRight(C)) = Col(C) **and** Hgt(ClimbRight(C)) = Hgt(C)
Col(C) ≠ Environment::Width(Envi(GoRight(C))) - 1
and Screen::CellNature(Envi(C),Col(C)+1,Hgt(C)+1) **not in** {**MTL**, **PLT**, **DOR**, **NGU** }
and not exists Guard g **in** Environment::CellContent(Envi(C),Col(C)+1,Hgt(C)+1)
and exists Player p **in** Environment::CellContent(Envi(C),Col(C),Hgt(C)+1)
implies Col(ClimbRight(C)) = Col(C) **and** Hgt(ClimbRight(C)) = Hgt(C)+1
Col(C) ≠ Environment::Width(Envi(GoRight(C))) - 1
and Screen::CellNature(Envi(C),Col(C)+1,Hgt(C)+1) **not in** {**MTL**, **PLT**, **DOR**, **NGU** }
and not exists Guard g **in** Environment::CellContent(Envi(C),Col(C)+1,Hgt(C)+1)
and not exists Player p **in** Environment::CellContent(Envi(C),Col(C),Hgt(C)+1)
implies Col(ClimbRight(C)) = Col(C)+1 **and** Hgt(ClimbRight(C)) = Hgt(C)+1

[MoveToInitCoords]: **exists** Guard g **in** Environment::CellContent(Envi(C),
Coord::X(InitCoords(C)), Coord::Y(InitCoords(C))) **implies exists** g **in**
Environment::CellContent(Envi(MoveToInitCoords(C)),
Coord::X(InitCoords(g)), Coord::Y(InitCoords(g)))
TimeInHole(MoveToInitCoords(C)) = 0
Col(MoveToInitCoords(C)) = Coord::X(InitCoords(MoveToInitCoords(C)))
Hgt(MoveToInitCoords(C)) = Coord::Y(InitCoords(MoveToInitCoords(C)))
IdCounter(MoveToInitCoords(C)) = IdCounter(C)
CarryingTreasure(MoveToInitCoords(C)) = CarryingTreasure(C)
TimeLeftParalyzed(MoveToInitCoords(C)) = 0

[Paralyze]: CarryingTreasure(Paralyze(C)) = CarryingTreasure(C)
TimeInHole(Paralyze(C)) = TimeInHole(C)
Col(Paralyze(C)) = Col(C)
Hgt(Paralyze(C)) = Hgt(C)
IdCounter(Paralyze(C)) = IdCounter(C)
TimeLeftParalyzed(Paralyze(C)) = 10

[Step]: **Falling(C) defined by**
(Environment::CellNature(Envi(C), Col(C), Hgt(C)) **not in** {**LAD**,**HDR**}
and Environment::CellNature(Envi(C), Col(C), Hgt(C)-1) **in** {**EMP**,**HDR**,**HOL**,**NPL**}
and not exists Guard g **in** Environment::CellContent(Envi(C), Col(C), Hgt(C)-1))
TimeLeftParalyzed(C) = 0 **implies** TimeLeftParalyzed(Step(C))
TimeLeftParalyzed(C) > 0 **and** Falling **implies** TimeLeftParalyzed(Step(C)) = TimeLeftParalyzed(C)
TimeLeftParalyzed(C) > 0 **and not** Falling **implies** TimeLeftParalyzed(Step(C)) = TimeLeftParalyzed(C) - 1
exists Treasure t **in** Environment::CellContent(Envi(C), Col(C), Hgt(C))
and not CarryingTreasure(C)
implies CarryingTreasure(Step(C))
and not exists t **in** Environment::CellContent(Envi(Step(C)), Col(C), Hgt(C))
exists Treasure t **in** Environment::CellContent(Envi(C), Col(C), Hgt(C))
and CarryingTreasure(C)
implies CarryingTreasure(Step(C))
and exists t **in** Environment::CellContent(Envi(Step(C)), Col(C), Hgt(C))
not exists Treasure t **in** Environment::CellContent(Envi(C), Col(C), Hgt(C))
and not CarryingTreasure(C)
implies not CarryingTreasure(Step(C))
and not exists t **in** Environment::CellContent(Envi(Step(C)), Col(C), Hgt(C))
not exists Treasure t **in** Environment::CellContent(Envi(C), Col(C), Hgt(C))
and CarryingTreasure(C)
and Environment::CellNature(Envi(C), Col(C), Hgt(C)-1) = HOL
and not Environment::CellNature(Envi(C), Col(C), Hgt(C)) **in** {**LAD**, **HDR**, **HOL**}
and not exists Guard g **in** Environment::CellContent(Envi(C), Col(C), Hgt(C)-1)
implies not CarryingTreasure(Step(C))
and exists Treasure t2 **in** Environment::CellContent(Envi(Step(C)), Col(C), Hgt(C))
not exists Treasure t **in** Environment::CellContent(Envi(C), Col(C), Hgt(C))
and CarryingTreasure(C)
(**and** Environment::CellNature(Envi(C), Col(C), Hgt(C)-1) ≠ HOL
or Environment::CellNature(Envi(C), Col(C), Hgt(C)) **in** {**LAD**, **HDR**, **HOL**}
or exists Guard g **in** Environment::CellContent(Envi(C), Col(C), Hgt(C)-1))
implies CarryingTreasure(Step(C))
and not exists Treasure t2 **in** Environment::CellContent(Envi(Step(C)), Col(C), Hgt(C))
TimeLeftParalyzed(C) = 0 **and** Environment::CellNature(Envi(C), Col(C), Hgt(C)) = HOL
and TimeInHole(C) < 5
implies TimeInHole(Step(C)) = TimeInHole(C)+1
TimeLeftParalyzed(C) = 0 **and** Environment::CellNature(Envi(C), Col(C), Hgt(C)) = HOL
and TimeInHole(C) ≥ 5
and Behaviour(C) = MOVEL
implies TimeInHole(Step(C)) = 0 **and** Step(C) = ClimbLeft(C)
and TimeInHole(C) ≥ 5
and Behaviour(C) = MOVER
implies TimeInHole(Step(C)) = 0 **and** Step(C) = ClimbRight(C)
Falling **implies** Step(C) = GoDown(C)
not Falling **and** TimeLeftParalyzed(C) = 0
and Environment::CellNature(Envi(C), Col(C), Hgt(C)) ≠ HOL
and Behaviour(C) = MOVEL
implies Step(C) = GoLeft(C)
not Falling **and** TimeLeftParalyzed(C) = 0
and Environment::CellNature(Envi(C), Col(C), Hgt(C)) ≠ HOL
and Behaviour(C) = MOVER
implies Step(C) = GoRight(C)
not Falling **and** TimeLeftParalyzed(C) = 0
and Environment::CellNature(Envi(C), Col(C), Hgt(C)) ≠ HOL
and Behaviour(C) = MOVEU
implies Step(C) = GoUp(C)
not Falling **and** TimeLeftParalyzed(C) = 0
and Environment::CellNature(Envi(C), Col(C), Hgt(C)) ≠ HOL
and Behaviour(C) = MOVED
implies Step(C) = GoDown(C)
IdCounter(Step(C)) = IdCounter(C)

Les fonctions de déplacements de Guard sont quasiment les mêmes que celles de Character, on change seulement le fait qu'on ne puisse pas aller dans une case NGU

`[GoLeft]`: Hgt(GoLeft(C)) = Hgt(C)
Col(C) = 0 **implies** Col(GoLeft(C)) = Col(C)
Environment::CellNature(Envi(C),Col(C)-1,Hgt(C)) **in** {**MTL**, **PLT**, **DOR**, **NGU**}
implies Col(GoLeft(C)) = Col(C)
Environment::CellNature(Envi(C),Col(C),Hgt(C)) **not in** {**LAD**, **HDR**}
and Environment::CellNature(Envi(C),Col(C),Hgt(C)-1) **not in** {**PLT**, **MTL**, **LAD**,**DOR**, **NGU**}
and not exists Guard g **in** Environment::CellContent(Envi(C),Col(C),Hgt(C)-1)
implies Col(GoLeft(C)) = Col(C)
exists Guard g **in** Environment::CellContent(Envi(C),Col(C)-1,Hgt(C))
implies Col(GoLeft(C)) = Col(C)
(Col(C) ≠ 0) **and** Environment::CellNature(Envi(C),Col(C)-1,Hgt(C)) **not in** {**MTL**, **PLT**,**DOR**, **NGU**}
and (Environment::CellNature(Envi(C),Col(C),Hgt(C)) **in** {**LAD**, **HDR**}
or Environment::CellNature(Envi(C),Col(C),Hgt(C)-1) **in** {**PLT**, **MTL**, **LAD**,**DOR**, **NGU**}
or exists Guard g **in** Environment::CellContent(Envi(C),Col(C),Hgt(C)-1))
and not (**exists** Guard g **in** Environment::CellContent(Envi(C),Col(C)-1,Hgt(C)))
implies Col(GoLeft(C)) = Col(C)-1

`[GoRight]`: Hgt(GoRight(C)) = Hgt(C)
Col(C) = Environment::Width(Envi(GoRight(C))) - 1 **implies** Col(GoRight(C)) = Col(C)
Environment::CellNature(Envi(C),Col(C)+1,Hgt(C)) **in** {**MTL**, **PLT**, **DOR**, **NGU**}
implies Col(GoRight(C)) = Col(C)
Environment::CellNature(Envi(C),Col(C),Hgt(C)) **not in** {**LAD**, **HDR**}
and Environment::CellNature(Envi(C),Col(C),Hgt(C)-1) **not in** {**PLT**, **MTL**, **LAD**,**DOR**, **NGU**}
and not exists Guard g **in** Environment::CellContent(Envi(C),Col(C),Hgt(C)-1)
implies Col(GoRight(C)) = Col(C)
exists Guard g **in** Environment::CellContent(Envi(C),Col(C)+1,Hgt(C))
implies Col(GoRight(C)) = Col(C)
(Col(C) ≠ Environment::Width(Envi(GoRight(C))) - 1)
and Environment::CellNature(Envi(C),Col(C)+1,Hgt(C)) **not in** {**MTL**, **PLT**,**DOR**, **NGU**}
and (Environment::CellNature(Envi(C),Col(C),Hgt(C)) **in** {**LAD**, **HDR**}
or Environment::CellNature(Envi(C),Col(C),Hgt(C)-1) **in** {**PLT**, **MTL**, **LAD**,**DOR**, **NGU**}
or exists Guard g **in** Environment::CellContent(Envi(C),Col(C),Hgt(C)-1))
and not (**exists** Guard g **in** Environment::CellContent(Envi(C),Col(C)+1,Hgt(C)))
implies Col(GoRight(C)) = Col(C)+1

`[GoUp]`: Col(GoUp(C)) = Col(C)
Hgt(C) = Environment::Height(Envi(GoUp(C))) - 1 **implies** Hgt(GoUp(C)) = Hgt(C)
Environment::CellNature(Envi(C),Col(C),Hgt(C)+1) **in** {**MTL**, **PLT**, **DOR**, **NGU**}
implies Hgt(GoUp(C)) = Hgt(C)
Environment::CellNature(Envi(C),Col(C),Hgt(C)) ≠ **LAD implies** Hgt(GoUp(C)) = Hgt(C)
exists Guard g **in** Environment::CellContent(Envi(C),Col(C),Hgt(C)+1)
implies Hgt(GoUp(C)) = Hgt(C)
(Hgt(C) ≠ Environment::Height(Envi(GoUp(C))) - 1)
and Environment::CellNature(Envi(C),Col(C),Hgt(C)+1) **not in** {**MTL**, **PLT**, **DOR**, **NGU**}
and Environment::CellNature(Envi(C),Col(C),Hgt(C)) = **LAD**
and exists Guard g in Environment::CellContent(Envi(C),Col(C),Hgt(C)+1)
implies Hgt(GoUp(C)) = Hgt(C)+1

`[GoDown]`: Col(GoDown(C)) = Col(C)
Hgt(C) = 0 **implies** Hgt(GoDown(C)) = Hgt(C)
Environment::CellNature(Envi(C),Col(C),Hgt(C)-1) **in** {**MTL**, **PLT**, **DOR**, **NGU**}
implies Hgt(GoDown(C)) = Hgt(C)
exists Guard g **in** Environment::CellContent(Envi(C),Col(C),Hgt(C)-1)
implies Hgt(GoDown(C)) = Hgt(C)
(Hgt(C) ≠ 0)
and Environment::CellNature(Envi(C),Col(C),Hgt(C)-1) **not in** {**MTL**, **PLT**, **DOR**, **NGU**}
and exists Guard g in Environment::CellContent(Envi(C),Col(C),Hgt(C)-1)
implies Hgt(GoDown(C)) = Hgt(C)-1

Moteur

Service: Engine

Observators: `const CommandManager`: [Engine] → CommandManager
`Environment`: [Engine] → Environment
`Player`: [Engine] → Player
`Guards`: [Engine] → List{Guard}
`Treasures`: [Engine] → List{Treasure}
`Status`: [Engine] → Status
`NextCommand`: [Engine] → Command
`Holes`: [Engine] → Set{Hole}
`NbLifes`: [Engine] → int
`Score`: [Engine] → int
`ScoreAtStartOfLevel`: [Engine] → int
`ScreenManager`: [Engine] → ScreenManager
`NbTreasuresLeft`: [Engine] → int
`CurrentLevel`: [Engine] → int

Constructors: `init`: ScreenManager × CommandManager × Engine → Engine
pre `init(sm,cm,e)` **requires** sm ≠ null **and** ScreenManager::NbScreen(sm) ≥ 1

Operators: `Step`: [Engine] → [Engine]
pre `Step(E)` **requires** `Status(E)` = **Playing**
`AddHole`: [Engine] × int × int → [Engine]
pre `AddHole(E,x,y)` **requires not exists** Hole h **in** Holes(E)
with (Hole::X(h) = x **and** Hole::Y(h) = y)
and Environment::CellNature(Environment(E), x, y) = **HOL**
`Display`: [Engine] → [Engine]

Observations:

`[invariants]`:

`[init]`: Environment::Height(Environment(init(sm,cm,e))) = Screen::Width(ScreenManager::Screen(sm,0))
Environment::Width(Environment(init(sm,cm,e))) = Screen::Height(ScreenManager::Screen(sm,0))
forall (i,j) in `[0;Environment::Width(Environment(init(sm,cm,e)))[`
`[0;Environment::Height(Environment(init(sm,cm,e)))[`,
Screen::CellNature(ScreenManager::Screen(sm, 0), i, j)
= Screen::CellNature(Environment(init(sm,cm,e), i, j)
Player::Col(Player(init(sm,cm,e))) = Coord::X(ScreenManager::PlayerFromScreen(sm, 0))
Player::Hgt(Player(init(sm,cm,e))) = Coord::Y(ScreenManager::PlayerFromScreen(sm, 0))
Player::Engine(Player(init(sm,cm,e))) = e
forall CoordItem c **in** ScreenManager::ItemsFromScreen(sm, 0),
exists Item i **in** Treasures(init(sm,cm,e))
with (Item::Hgt(i) = CoordItem::Y(c) **and** Item::Col(i) = CoordItem::X(c)
and Item::Nature(i) = CoordItem::ItemType(c))
forall CoordGuard c **in** ScreenManager::GuardsFromScreen(sm, 0),
exists Guard g **in** Guards(init(sm,cm,e))
with (Guard::Hgt(g) = CoordGuard::Y(c) **and** Guard::Col(i) = CoordGuard::X(c)
and Guard::Nature(i) = CoordGuard::Type(c))
Status(init(sm,cm,e)) = **Playing**
NextCommand(init(sm,cm,e)) = **NONE**
Holes(init(sm,cm,e)) = {}
NbLifes(init(sm,cm,e)) = 3
Score(init(sm,cm,e)) = 0
ScoreAtStartOfLevel(init(sm,cm,e)) = 0
ScreenManager(init(sm,cm,e)) = sm
NbTreasuresLeft(init(sm,cm,e)) = **count** CoordItem c **in**
ScreenManager::ItemsFromScreen(sm,0) **with** (CoordItem::ItemType(c) = Treasure)
CurrentLevel(init(sm, cm, e)) = 0
CommandManager(init(sm, cm, e)) = cm

[Step]: **loadlevel(E, no)(C) defined by**
CurrentLevel(Step(E)) = no
and Status(Step(E)) = Playing
and Status(Step(E)) = Playing
and Environment::Height(Environment(Step(E)) = Screen::Width(ScreenManager::Screen(sm,no))
and Environment::Width(Environment(Step(E)) = Screen::Height(ScreenManager::Screen(sm,no))
and forall (i,j) in [0;Environment::Width(Environment(Step(E))[
[0;Environment::Height(Environment(Step(E))[,
Screen::CellNature(ScreenManager::Screen(sm, no), i, j) =
Screen::CellNature(Environment(Step(E), i, j)
and Player::Col(Player(Step(E)) = Coord::X(ScreenManager::PlayerFromScreen(sm, no))
and Player::Hgt(Player(Step(E)) = Coord::Y(ScreenManager::PlayerFromScreen(sm, no))
and forall CoordItem c **in** ScreenManager::ItemsFromScreen(sm, 0),
exists Item i **in** Treasures(Step(E))
with (Item::Hgt(i) = CoordItem::Y(c) **and** Item::Col(i) = CoordItem::X(c)
and Item::Nature(i) = CoordItem::ItemType(c))
and forall CoordGuard c **in** ScreenManager::GuardsFromScreen(sm, 0),
exists Guard g **in** Guards(Step(E))
with (Guard::Hgt(g) = CoordGuard::Y(c) **and** Guard::Col(i) = CoordGuard::X(c)
and Guard::Nature(i) = CoordGuard::Type(c))
and Status(Step(E)) = **Playing**
and NextCommand(Step(E)) = **NONE**
and Holes(Step(E)) = {}
and NbLifes(Step(E)) = NbLifes(E) - 1
and Score(Step(E)) = Score(E)
and ScoreAtStartOfLevel(Step(E)) = Score(E)
and ScreenManager(Step(E)) = ScreenManager(E)
and NbTreasuresLeft(Step(E)) = **count** CoordItem c **in**
ScreenManager::ItemsFromScreen(sm,no) **with** (CoordItem::ItemType(c) = Treasure)
death(E) defined by
(NbLifes(E) = 1 **implies** NbLifes(Step(E)) = 0 **and** Status(Step(E)) = Loss)
and (NbLifes(E) > 1
implies NbLifes(Step(E)) = NbLifes(E) -1
and NbLifes(Step(E)) = NbLifes(E) -1
and Score(Step(E)) = ScoreAtStartOfLevel(E)
and loadlevel(E, CurrentLevel(E))
CommandManager(E) ≠ null
implies NextCommand(Step(E)) = CommandManager::CurrentCommand(CommandManager(E))
CommandManager(E) = null
implies NextCommand(Step(E)) = NONE
exists Guard g **in** Environment::CellContent(Envi(E), Player::Col(Player(E)), Player::Hgt(Player(E)))
and NbLifes(E) = 1
implies NbLifes(Step(E)) = 0 **and** Status(Step(E)) = Loss
exists Guard g **in** Environment::CellContent(Envi(E), Player::Col(Player(E)), Player::Hgt(Player(E)))
and NbLifes(E) > 1
implies death(E)
not exists Guard g **in** Environment::CellContent(Envi(E), Player::Col(Player(E)), Player::Hgt(Player(E)))
and exists Treasure t **in** Environment::CellContent(Envi(E), Player::Col(Player(E)), Player::Hgt(Player(E)))
implies t **not in** Treasure(Step(E))
and NbTreasuresLeft(Step(E)) = NbTreasuresLeft(E)-1
and Score(Step(E)) = Score(E)+1
and (NbTreasuresLeft(E) = 1
and CurrentLevel(E) < ScreenManager::NbScreen(ScreenManager(E)) - 1
implies CurrentLevel(Step(E)) = CurrentLevel(E)+1
and loadlevel(E, CurrentLevel(E)))
and (NbTreasuresLeft(E) = 1
and CurrentLevel(E) = ScreenManager::NbScreen(ScreenManager(E)) - 1
implies CurrentLevel(Step(E)) = CurrentLevel(E)
and Status(Step(E)) = Win

`[Step]`: **not exists** Guard g **in** Environment::CellContent(Envi(E), Player::Col(Player(E)), Player::Hgt(Player(E)))
and exists Item i **in** Environment::CellContent(Envi(E), Player::Col(Player(E)), Player::Hgt(Player(E)))
implies Item::Nature(Player::CurrentlyHeldItem(Player(Step(E)))) = Item::Nature(i)
not exists Guard g **in** Environment::CellContent(Envi(E), Player::Col(Player(E)), Player::Hgt(Player(E)))
and Environment::CellNature(Envi(E), Player::Col(Player(E)), Player::Hgt(Player(E)) - 1) = **TRP**
implies Environment::CellNature(Envi(Step(E)),
Player::Col(Player(E)), Player::Hgt(Player(E)) - 1) = **EMP**
not (**exists** Treasure tin Environment::CellContent(Envi(E), Player::Col(Player(E)), Player::Hgt(Player(E)))
and NbTreasuresLeft(E) = 1)
and not exists Guard g
in Environment::CellContent(Envi(E), Player::Col(Player(E)), Player::Hgt(Player(E)))
implies forall Hole h **in** Holes(E),
(Hole::Time(h) < 14 **implies exists** Hole o
with (Hole::X(o) = Hole::X(h) **and** Hole::Y(o) == Hole::Y(h) **and** Hole::Time(o) = Hole::Time(h)))
and (Hole::Time(h) = 14 **implies not exists** Hole o
with (Hole::X(o) = Hole::X(h) **and** Hole::Y(o) == Hole::Y(h)))
and (Hole::Time(h) = 14 **and** Player::X(Player(E)) = Hole::X(h)
and Player::Y(Player(E)) = Hole::Y(h)
implies death(E))
not exists Hole h **in** Holes(E) **with** (Hole::Time(h) = 14 **and** Hole::X(h) = Player::Col(Player(E))
and Hole::Y(h) = Player::Hgt(Player(E)))
and not exists Guard g **in**
Environment::CellContent(Envi(E), Player::Col(Player(E)), Player::Hgt(Player(E)))
and not (**exists** Treasure t**in**
Environment::CellContent(Envi(E), Player::Col(Player(E)), Player::Hgt(Player(E)))
and NbTreasuresLeft(E) = 1)
implies Status(Step(E)) = Playing

`[AddHole]`: Holes(AddHole(E,x,y)) = Holes(E) **union** Hole h **with** (Hole::X(h) = x **and** Hole::Y(h) = y)
Environment(AddHole(E,x,y)) = Environment(E)
Player(AddHole(E,x,y)) = Player(E)
Guards(AddHole(E,x,y)) = Guards(E)
Treasures(AddHole(E,x,y)) = Treasures(E)
Status(AddHole(E,x,y)) = Status(E)
NextCommand(AddHole(E,x,y)) = NextCommand(E)
NbLifes(AddHole(E,x,y)) = NbLifes(E)
Score(AddHole(E,x,y)) = Score(E)
ScoreAtStartOfLevel(AddHole(E,x,y)) = ScoreAtStartOfLevel(E)
ScreenManager(AddHole(E,x,y)) = ScreenManager(E)
NbTreasuresLeft(AddHole(E,x,y)) = NbTreasuresLeft(E)
CurrentLevel(AddHole(E,x,y)) = CurrentLevel(E)

`[Display]`: Environment(AddHole(E,x,y)) = Environment(E)
Player(Display(E)) = Player(E)
Guards(Display(E)) = Guards(E)
Treasures(Display(E)) = Treasures(E)
Status(Display(E)) = Status(E)
NextCommand(Display(E)) = NextCommand(E)
Holes(Display(E)) = Holes(E)
NbLifes(Display(E)) = NbLifes(E)
Score(Display(E)) = Score(E)
ScoreAtStartOfLevel(Display(E)) = ScoreAtStartOfLevel(E)
ScreenManager(Display(E)) = ScreenManager(E)
NbTreasuresLeft(Display(E)) = NbTreasuresLeft(E)
CurrentLevel(Display(E)) = CurrentLevel(E)

Gestionnaire d'écran

Service: ScreenManager

Observators: `NbScreen`: [ScreenManager] → int

`LevelSetup`: [ScreenManager] × int → LevelSetup
pre `LevelSetup(S,i)` **requires** $0 \leq$ `i` $<$ `NbScreen(S)`

`Screen`: [ScreenManager] × int → EditableScreen
pre `Screen(S,i)` **requires** $0 \leq$ `i` $<$ `NbScreen(S)`

`GuardsFromScreen`: [ScreenManager] × int → List{CoordGuard}
pre `GuardsFromScreen(S,i)` **requires** $0 \leq$ `i` $<$ `NbScreen(S)`

`ItemsFromScreen`: [ScreenManager] × int → List{CoordItem}
pre `ItemsFromScreen(S,i)` **requires** $0 \leq$ `i` $<$ `NbScreen(S)`

`PlayerFromScreen`: [ScreenManager] × int → Coord
pre `PlayerFromScreen(S,i)` **requires** $0 \leq$ `i` $<$ `NbScreen(S)`

Constructors: `init`: → [ScreenManager]

Operators: `AddScreen`: [ScreenManager] × EditableScreen × List{CoordGuard} × List{CoordItem} × Coord → [ScreenManager]
pre `AddScreen(S,es,g,i,p)` **requires** p ≠ null **and** es ≠ null

`RemoveScreen`: [ScreenManager] × int → [ScreenManager]
pre `RemoveScreen(S,i)` **requires** $0 \leq$ `i` $<$ `NbScreen(S)`

Observations:

`[invariants]`: `Screen(S, i)` **min** LevelSetup::Screen(LevelSetup(S, i))
`GuardsFromScreen(S, i)` **min** LevelSetup::Guards(LevelSetup(S, i))
`ItemsFromScreen(S, i)` **min** LevelSetup::Items(LevelSetup(S, i))
`PlayerFromScreen(S, i)` **min** LevelSetup::Player(LevelSetup(S, i))

`[init]`: NbScreen(init()) = 0

`[AddScreen]`: NbScreen(AddScreen(S,es,g,i,p)) = NbScreen(S) + 1
Screen(AddScreen(S,es,g,i,p), NbScreen(S)) = es
PlayerFromScreen(AddScreen(S,es,g,i,p), NbScreen(S)) = p
g = null **implies** GuardsFromScreen(AddScreen(S,es,g,i,p), NbScreen(S)) = {}
g ≠ null **implies** GuardsFromScreen(AddScreen(S,es,g,i,p), NbScreen(S)) = g
i = null **implies** ItemsFromScreen(AddScreen(S,es,g,i,p), NbScreen(S)) = {}
i ≠ null **implies** ItemsFromScreen(AddScreen(S,es,g,i,p), NbScreen(S)) = i
forall cpt in `[0, NbScreen(S)[`,
LevelSetup(AddScreen(S,es,g,i,p), cpt) = LevelSetup(S, cpt)

`[RemoveScreen]`: NbScreen(RemoveScreen(S,i)) = NbScreen(S) - 1
forall cpt in `[0, NbScreen(S)[`,
cpt < i **implies** LevelSetup(RemoveScreen(S,i), cpt) = LevelSetup(S, cpt)
cpt ≥ i **implies** LevelSetup(RemoveScreen(S,i), cpt) = LevelSetup(S, cpt + 1)