Chapter 1

BridgeRstudent

Exemple du cours : Machine AbstractBridge et raffinement ConcreteBridge

1.1 Machine abstraite

Module AbstractBridge.

1.1.1 Contexte

Module Context.

Parameter *max_nb_cars* : **nat**.

Axiom *max_nb_cars_not_zero* : *max_nb_cars* > O.

End Context.

1.1.2 Etat

```
Record State : Set := mkState {
  nb_cars_entered: nat
}.
```

1.1.3 Invariants

```
Definition Inv_1 (B:State) : Prop :=
 nb_cars_entered B ≤ Context.max_nb_cars.
```

1.1.4 Evénement : Init

```
Module INIT.

Definition Guard (limit:nat) : Prop :=
  Context.max_nb_cars = limit.

Definition action (limit:nat) : State :=
  mkState 0.

Theorem PO_Safety:
  ∀ lim : nat,
    Guard lim
    → let B := action lim
       in Inv_1 B.

End INIT.
```

1.1.5 Evénement : CarEnterFromMainland

```
Module CARENTER.

Definition Guard (B:State) : Prop :=
  B.(nb_cars_entered) < Context.max_nb_cars.

Definition action (B:State) : State :=
  mkState (S B.(nb_cars_entered)).

Theorem PO_Safety:
  ∀ (B:State),
    Inv_1 B
    → Guard B
    → let B' := action B
       in Inv_1 B'.

End CARENTER.
```

1.1.6 Evénement : CarLeave

```
Module CARLEAVE.

Definition Guard (B:State) : Prop :=
  B.(nb_cars_entered) > 0.

Definition action (B:State) : State :=
  mkState (pred B.(nb_cars_entered)).

Theorem PO_Safety:
  ∀ (B:State),
    Inv_1 B
```

```
    → Guard B
    → let B' := action B
       in Inv_1 B'.
End CarLeave.
```

1.1.7 Deadlock freedom

```
Theorem PO_Deadlock_Freedom:
  ∀ (B:State),
    Inv_1 B
    → CarEnter.Guard B
       ∨ CarLeave.Guard B.
End AbstractBridge.
```

1.2 Machine concrete

```
Module ConcreteBridge.
```

1.2.1 Contexte

```
Module Context.
Definition max_nb_cars : nat := AbstractBridge.Context.max_nb_cars.
Lemma max_nb_cars_not_zero:
  max_nb_cars > 0.
End Context.
```

1.2.2 Etat

```
Record State : Set := mkState {
  nb_cars_to_island: nat;
  nb_cars_to_mainland: nat;
  nb_cars_on_island: nat

}.
Definition total_nb_cars (B:State) : nat :=
  B.(nb_cars_to_island)
  + B.(nb_cars_to_mainland)
  + B.(nb_cars_on_island).
```

1.2.3 Invariants

Invariants de glue

```
Definition Glue_1 (B:State) (AB:AbstractBridge.State) : Prop :=
  total_nb_cars B = AbstractBridge.nb_cars_entered AB.
```

Invariants concrets

```
Definition Inv_1 (B:State) : Prop :=
  B.(nb_cars_to_island) = 0
  ∨ B.(nb_cars_to_mainland) = 0.
```

1.2.4 Evénement : Init

```
Module Init.

Definition Guard (limit:nat) : Prop :=
  Context.max_nb_cars = limit.

Definition action (limit:nat) : State :=
  mkState 0 0 0.

Theorem PO_Strengthening:
  ∀ lim : nat,
    Guard lim
    → AbstractBridge.Init.Guard lim.

Theorem PO_Safety:
  ∀ lim : nat,
    Guard lim
    → let B := action lim
       in Inv_1 B.

Theorem PO_Simulation:
  ∀ lim : nat,
    Guard lim
    → let B := action lim in
       let AB := AbstractBridge.Init.action lim
       in
       Glue_1 B AB.

End Init.
```

1.2.5 Evénement : CarEnterFromMainland

```
Module CarEnterFromMainland.
```

```
Definition Guard (B:State) : Prop :=
  B.(nb_cars_to_mainland) = 0
  ∧ B.(nb_cars_to_island) + B.(nb_cars_on_island) < Context.max_nb_cars.

Definition action (B:State) : State :=
  mkState
    (S B.(nb_cars_to_island))
    B.(nb_cars_to_mainland)
        B.(nb_cars_on_island).

Theorem PO_Strengthening:
  ∀ B : State, ∀ AB : AbstractBridge.State,
    AbstractBridge.Inv_1 AB
    → Glue_1 B AB
    → Inv_1 B
    → Guard B
    → AbstractBridge.CarEnter.Guard AB.

Theorem PO_Safety:
  ∀ (B:State), ∀ (AB:AbstractBridge.State),
    AbstractBridge.Inv_1 AB
    → Glue_1 B AB
    → Inv_1 B
    → Guard B
    → let B' := action B
       in Inv_1 B'.

Theorem PO_Simulation:
  ∀ (B:State), ∀ (AB:AbstractBridge.State),
    AbstractBridge.Inv_1 AB
    → Glue_1 B AB
    → Inv_1 B
    → Guard B
    → let B' := action B
       in let AB' := AbstractBridge.CarEnter.action AB
          in
          Glue_1 B' AB'.

End CarEnterFromMainland.
```

1.2.6 Evénement : CarLeaveToIsland

```
Module CarLeaveToIsland.

Definition Guard (B:State) : Prop :=
  B.(nb_cars_to_island) > 0.
```

```
Definition action (B:State) : State :=
  mkState
    (pred B.(nb_cars_to_island))
    B.(nb_cars_to_mainland)
        (S B.(nb_cars_on_island)).

Theorem PO_Safety:
  ∀ (B:State) (AB:AbstractBridge.State),
    AbstractBridge.Inv_1 AB
    → Glue_1 B AB
    → Inv_1 B
    → Guard B
    → let B' := action B
       in Inv_1 B'.

Theorem PO_Simulation:
  ∀ (B:State), ∀ (AB:AbstractBridge.State),
    AbstractBridge.Inv_1 AB
    → Glue_1 B AB
    → Inv_1 B
    → Guard B
    → let B' := action B
       in
       Glue_1 B' AB.

Definition variant (B:State) : nat :=
  B.(nb_cars_to_island).

Theorem PO_Convergence:
  ∀ B : State, ∀ AB : AbstractBridge.State,
    AbstractBridge.Inv_1 AB
    → Glue_1 B AB
    → Inv_1 B
    → Guard B
    → let B' := action B
       in
       variant B' < variant B.

End CarLeaveToIsland.
```

1.2.7 Evénement : CarEnterFromIsland

```
Module CarEnterFromIsland.

Definition Guard (B:State) : Prop :=
  B.(nb_cars_on_island) > 0
```

```
  ∧ B.(nb_cars_to_island) = 0.
Definition action (B:State) : State :=
  mkState
    B.(nb_cars_to_island)
        (S B.(nb_cars_to_mainland))
        (pred B.(nb_cars_on_island)).
Theorem PO_Safety:
  ∀ B : State, ∀ AB : AbstractBridge.State,
    AbstractBridge.Inv_1 AB
    → Glue_1 B AB
    → Inv_1 B
    → Guard B
    → let B' := action B
       in Inv_1 B'.
Theorem PO_Simulation:
  ∀ (B:State), ∀ (AB:AbstractBridge.State),
    AbstractBridge.Inv_1 AB
    → Glue_1 B AB
    → Inv_1 B
    → Guard B
    → let B' := action B
       in
       Glue_1 B' AB.
Definition variant (B:State) : nat :=
  B.(nb_cars_on_island).
Theorem PO_Convergence:
  ∀ B : State, ∀ AB : AbstractBridge.State,
    AbstractBridge.Inv_1 AB
    → Glue_1 B AB
    → Inv_1 B
    → Guard B
    → let B' := action B
       in
       variant B' < variant B.
End CarEnterFromIsland.
```

1.2.8 Evénement : CarLeaveToMainland

```
Module CarLeaveToMainland.
Definition Guard (B:State) : Prop :=
```

```
  B.(nb_cars_to_mainland) > 0.
Definition action (B:State) : State :=
  mkState
    B.(nb_cars_to_island)
        (pred B.(nb_cars_to_mainland))
        B.(nb_cars_on_island).
Theorem PO_Strengthening:
  ∀ B : State, ∀ AB : AbstractBridge.State,
    AbstractBridge.Inv_1 AB
    → Glue_1 B AB
    → Inv_1 B
    → Guard B
    → AbstractBridge.CarLeave.Guard AB.
Lemma pred_plus:
  ∀ n m : nat, n > 0 → (pred n) + m = pred (n + m).
Theorem PO_Safety:
  ∀ B:State, ∀ AB:AbstractBridge.State,
    AbstractBridge.Inv_1 AB
    → Glue_1 B AB
    → Inv_1 B
    → Guard B
    → let B' := action B
       in Inv_1 B'.
Theorem PO_Simulation:
  ∀ (B:State), ∀ (AB:AbstractBridge.State),
    AbstractBridge.Inv_1 AB
    → Glue_1 B AB
    → Inv_1 B
    → Guard B
    → let B' := action B
       in let AB' := AbstractBridge.CarLeave.action AB
          in
          Glue_1 B' AB'.
End CarLeaveToMainland.
```

1.2.9 Relative deadlock freedom

```
Theorem PO_Deadlock_Freedom:
  ∀ B:State, ∀ AB:AbstractBridge.State,
    AbstractBridge.Inv_1 AB
    → Glue_1 B AB
```

$\rightarrow$ Inv_1 B
$\rightarrow$ CarEnterFromMainland.Guard B
$\vee$ CarLeaveToIsland.Guard B
$\vee$ CarEnterFromIsland.Guard B
$\vee$ CarLeaveToMainland.Guard B.

`End` ConcreteBridge.