Chapter 1

Bridge

```
Module Bridge.
```

1.1 Contexte : constantes et axiomes

```
Module Context.

Parameter max_nb_cars : nat.

Axiom max_nb_cars_not_zero : max_nb_cars > 0.

End Context.
```

1.2 Etat de la machine

```
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).
```

1.2.1 Invariants

```
Definition Inv_1 (B:State) : Prop :=
```

total_nb_cars $B \leq$ *Context.max_nb_cars*.

```
Definition Inv_2 (B:State) : Prop :=
  B.(nb_cars_to_island) = 0
  ∨ B.(nb_cars_to_mainland) = 0.
```

1.2.2 Evénement : initialisation

```
Module INIT.

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

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

Obligation de preuve : sûreté

```
Lemma PO_Safety_Inv_1:
  ∀ lim : nat,
    Guard lim
    → let B := action lim
      in Inv_1 B.

Lemma PO_Safety_Inv_2:
  ∀ lim : nat,
    Guard lim
    → let B := action lim
      in Inv_2 B.

Lemma PO_Safety:
  ∀ lim : nat,
    Guard lim
    → let B := action lim
      in Inv_1 B ∧ Inv_2 B.

End INIT.
```

1.2.3 Evénement : entrée depuis le continent

```
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).
```

Obligation de preuve : sûreté

```
Lemma PO_Safety_Inv_1:
  ∀ (B:State),
    Inv_1 B → Inv_2 B
    → Guard B
    → let B' := action B
      in Inv_1 B'.

Lemma PO_Safety_Inv_2:
  ∀ (B:State),
    Inv_1 B → Inv_2 B
    → Guard B
    → let B' := action B
      in Inv_2 B'.

Theorem PO_Safety:
  ∀ (B:State),
    Inv_1 B → Inv_2 B
    → Guard B
    → let B' := action B
      in Inv_1 B' ∧ Inv_2 B'.
```

Obligation de preuve : convergence

```
Definition variant (B:State) : nat :=
  Context.max_nb_cars -
  ( B.(nb_cars_to_island) + B.(nb_cars_on_island) ).

Lemma minus_S:
  ∀ n m : nat,
    m < n → n - S m < n - m.

Theorem PO_Convergence:
  ∀ B : State,
    Inv_1 B → Inv_2 B
    → Guard B
    → let B' := action B
      in
      variant B' < variant B.
```

End CarEnterFromMainland.

1.2.4 Evénement : sortie vers l'île

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)).
```

Obligation de preuve : sûreté

```
Lemma PO_Safety_Inv_1:
  ∀ (B:State),
    Inv_1 B → Inv_2 B
    → Guard B
    → let B' := action B
      in Inv_1 B'.

Lemma PO_Safety_Inv_2:
  ∀ (B:State),
    Inv_1 B → Inv_2 B
    → Guard B
    → let B' := action B
```

```
        in Inv_2 B'.

Lemma PO_Safety:
  ∀ (B:State),
    Inv_1 B → Inv_2 B
    → Guard B
    → let B' := action B
        in Inv_1 B' ∧ Inv_2 B'.

End CarLeaveToIsland.
```

1.2.5 Evénement : entrée depuis l'île

```
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)).
```

Obligation de preuve : sûreté

```
Lemma PO_Safety_Inv_1:
  ∀ (B:State),
    Inv_1 B → Inv_2 B
    → Guard B
    → let B' := action B
        in Inv_1 B'.

Lemma PO_Safety_Inv_2:
  ∀ (B:State),
    Inv_1 B → Inv_2 B
    → Guard B
    → let B' := action B
        in Inv_2 B'.

Lemma PO_Safety:
  ∀ (B:State),
    Inv_1 B → Inv_2 B
    → Guard B
    → let B' := action B
        in Inv_1 B' ∧ Inv_2 B'.
```

End CarEnterFromIsland.

1.2.6 Evénement : sortie vers le continent

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).
```

Obligation de preuve : sûreté

```
Lemma PO_Safety_Inv_1:
  ∀ (B:State),
    Inv_1 B → Inv_2 B
    → Guard B
    → let B' := action B
      in Inv_1 B'.

Lemma PO_Safety_Inv_2:
  ∀ (B:State),
    Inv_1 B → Inv_2 B
    → Guard B
    → let B' := action B
      in Inv_2 B'.

Lemma PO_Safety:
  ∀ (B:State),
    Inv_1 B → Inv_2 B
    → Guard B
    → let B' := action B
      in Inv_1 B' ∧ Inv_2 B'.
```

End CarLeaveToMainland.

1.2.7 Obligation de preuve : absence de deadlock

```
Theorem PO_Deadlock_Freedom:
  ∀ (B:State),
    Inv_1 B → Inv_2 B
    → CarEnterFromMainland.Guard B
        ∨ CarLeaveToIsland.Guard B
        ∨ CarEnterFromIsland.Guard B
        ∨ CarLeaveToMainland.Guard B.

End Bridge.
```