





**APPROPRIATION DURING ADT**

**SURVIVAL STRATON**

```
import { identity } from "fp-ts/lib/function";

type Direction = "North" | "East" | "South" | "West"

type Command<A, B, C> =
  | { kind: "Face", direction: Direction, proof: (_a: A) => A }
  | { kind: "Start", proof: (_a: A) => B }
  | { kind: "Stop", proof: (_a: A) => B }
  | { kind: "Chain", first: Command<A, B, any>, second: Command<B, C, any> }

const face = (d: Direction)
  : Command<"Idle", "Idle", void> => ({ kind: "Face", direction: d, proof: identity<"Idle"> })
const start = (): Command<"Idle", "Moving", void> => ({ kind: "Start", proof: (_: "Idle") => "Moving" })
const stop_ = (): Command<"Moving", "Idle", void> => ({ kind: "Stop", proof: (_: "Moving") => "Idle" })
const chain = <A, B, C>(first: Command<A, B, any>, second: Command<B, C, any>)
  : Command<A, C, any> => ({ kind: "Chain", first, second }) as Command<A, C, any>
```

MAGNUM2-QUALITÉ D'SI-THOMAS HAESLÉ & QUENTIN BURG

On paramètre le type `command` avec 3 paramètres

Pour `Face`, `B` & `C` sont des types fantômes

Pour `Start` et `Stop`, `C` est un type fantôme

On ajoute des preuves :

`Face` conserve l'état de la machine: **"Idle"**. La fonction `Identity` peut être utilisée comme preuve.

`Start` fait avancer la machine à états : **"Idle"** => **"Moving"**

`Start` fait avancer la machine à états : **"Idle"** => **"Moving"**

`Chain` fait avancer la machine à états en composant des commandes par associativité : **la preuve est directement encodée dans leurs types**

# SUIVRE LES TRANSITION

## APPROXIMATION D'UN GADT

```
import { identity } from "fp-ts/lib/function";

type Direction = "North" | "East" | "South" | "West"

type Command<A, B, C> =
  | { kind: "Face", direction: Direction, proof: (_a: A) => A }
  | { kind: "Start", proof: (_a: A) => B }
  | { kind: "Stop", proof: (_a: A) => B }
  | { kind: "Chain", first: Command<A, B, any>, second: Command<B, C, any> }

const face = (d: Direction)
  : Command<"Idle", "Idle", void> => ({ kind: "Face", direction: d, proof: identity<"Idle"> })
const start = (): Command<"Idle", "Moving", void> => ({ kind: "Start", proof: (_: "Idle") => "Moving" })
const stop_ = (): Command<"Moving", "Idle", void> => ({ kind: "Stop", proof: (_: "Moving") => "Idle" })
const chain = <A, B, C>(first: Command<A, B, any>, second: Command<B, C, any>)
  : Command<A, C, any> => ({ kind: "Chain", first, second }) as Command<A, C, any>
```

On paramètre le type command avec 3 paramètres

Pour Face, B & C sont des types fantômes

Pour Start et Stop, C est un type fantôme

On ajoute des preuves :

Face conserve l'état de la machine: "Idle". La fonction Identity peut être utilisée comme preuve.

Start fait avancer la machine à états : "Idle" => "Moving"

Start fait avancer la machine à états : "Idle" => "Moving"

Chain fait avancer la machine à états en composant des commandes par associativité : la preuve est directement encodée dans leurs types



# SUIVRE LES TRANSITION

ON PEUT CRÉER UNIQUEMENT DES COMMANDES VALIDES

```
/* COMPILER */
let validCmd =
  chain(
    chain(
      face("East"),
      start()
    ),
    stop_()
  );

/* ERREUR DE COMPILATION */
let invalidCmd = chain(
  face("East"),
  stop_()
);
```