





**DESCRIPTION**

**SURVIVAL STRACTIONS**

```
let _ = Start --> Start
```

## Compilation ERROR

Error: This expression has type (idle, moving) command  
but an expression was expected of type (moving, 'a) command  
Type idle is not compatible with type moving

MAGNET 2-DUALS - TRANSLES

```
let _ = Face West --> Start --> Stop
```

**On peut créer uniquement des commandes valides**

# SUIVRE LES TRANSITION

## DESCRIPTION D'UN GADT

```
let _ = Start --> Start
```

### Compilation ERROR

Error: This expression has type (idle, moving) command  
but an expression was expected of type (moving, 'a) command  
Type idle is not compatible with type moving

```
let _ = Face West --> Start --> Stop
```

On peut créer uniquement des commandes valides



# TAKE AWAY

## NOUS AVONS VU

Les GADT sont des types « OU » qui possèdent des témoins de type

Ils permettent (entre autre) de valider à la compilation les transitions d'états

ADT + GADT permettent de valider qu'on ne **représente** que des états autorisés **et qu'on effectue** que des transitions d'états autorisés

Nous avons une machine à états finis, validée par compilation

Seulement en Haskell, OCaml, Scala

