



- ▶ Introduction and overview
- ▶ Basic types, definitions and functions
- ▶ Basic data structures
- ▼ **More advanced data structures**

Table of Contents

Tagged values

Week 3 Échéance le déc 12, 2016 at 23:30 UTC

Recursive types

Week 3 Échéance le déc 12, 2016 at 23:30 UTC

Tree-like values

Week 3 Échéance le déc 12, 2016 at 23:30 UTC

Case study: a story teller

Week 3 Échéance le déc 12, 2016 at 23:30 UTC

Polymorphic algebraic datatypes

Week 3 Échéance le déc 12, 2016 at 23:30 UTC

Advanced topics

Week 3 Échéance le déc 12, 2016 at 23:30 UTC

- ▶ Higher order functions
- ▶ Exceptions, input/output and imperative constructs
- ▶ Modules and data abstraction

TYPE DIRECTED PROGRAMMING (40/40 points)

In this exercise, you will experiment with *type-directed programming*.

We give you the example program of the lecture in which two type definitions have been changed as in the given prelude. A case `Tired` has been added to type `state`, and a case `Sleep` has been added to type `action`.

By clicking the *typecheck* button, you can notice that several warnings are issued by the OCaml compiler. Go through the code and fix these warnings as follow.

1. Update `apply_action` so that the `Sleep` action turns a character from the `Tired` state to the `Hungry` state.
2. Update `possible_changes_for_character` so that the `Tired` state behaves as the `Hungry` state.
3. Update `describe_state` so that the description of the `Tired` state is `"tired"`.
4. Update `tell_action` so that `tell_action Sleep` is `"took a nap"`.

THE GIVEN PRELUDE

```
type story = {  
  context      : context;  
  perturbation : event;  
  adventure    : event list;  
  conclusion   : context;  
}  
  
and context = { characters : character list }  
and character = { name : string; state : state; location : location }  
and event = Change of character * state | Action of character * action  
and state = Happy | Hungry | Tired  
and action = Eat | GoToRestaurant | Sleep  
and location = Apartment | Restaurant
```

YOUR OCAML ENVIRONMENT

```
1 let compatible_actions_for_character character context =  
2   match character with  
3   | { location = Restaurant } -> [Eat]  
4   | { location = Apartment } -> [GoToRestaurant]  
5   ;;  
6  
7 let apply_action character = function  
8   | Eat ->  
9     { state = Happy;  
10      location = character.location; name = character.name }  
11   | GoToRestaurant ->  
12     { location = Restaurant;  
13      state = character.state; name = character.name }  
14   | Sleep ->  
15     { state = Hungry; location = character.location; name = character.name }  
16   ;;  
17  
18 let compatible_actions context =  
19   let rec aux = function  
20     | [] -> []  
21     | character :: cs ->  
22       let can_do = compatible_actions_for_character character context in  
23       let rec aux' = function  
24         | [] -> []  
25         | a :: actions -> Action (character, a) :: aux' actions  
26       in  
27         aux' can_do  
28   in  
29     aux context.characters  
30   ;;  
31  
32 let possible_changes_for_character character =  
33   match character with
```

Evaluate >

Switch >>

Typecheck

Reset Templ

Full-screen |

Check & Sa

Exercise complete (click for details)

40 pts

▼ Exercise 1: apply_action

Completed, 10 pts

Found `apply_action` with compatible type.

Computing `apply_action {name = "Cagdas"; state = Tired; location = Apartment}` Sleep

Correct value {name = "Cagdas"; state = Hungry; location = Apartment} 1 pt

Computing

Computing apply_action {name = "Ralf"; state = Happy; location = Restaurant} GoToRestaurant	
Correct value {name = "Ralf"; state = Happy; location = Restaurant}	1 pt
Computing apply_action {name = "Benjamin"; state = Tired; location = Restaurant} Eat	
Correct value {name = "Benjamin"; state = Happy; location = Restaurant}	1 pt
Computing apply_action {name = "Cagdas"; state = Happy; location = Restaurant} Eat	
Correct value {name = "Cagdas"; state = Happy; location = Restaurant}	1 pt
Computing apply_action {name = "Gregoire"; state = Happy; location = Appartement} Sleep	
Correct value {name = "Gregoire"; state = Hungry; location = Appartement}	1 pt
Computing apply_action {name = "Yann"; state = Hungry; location = Appartement} Eat	
Correct value {name = "Yann"; state = Happy; location = Appartement}	1 pt
Computing apply_action {name = "Yann"; state = Tired; location = Appartement} Eat	
Correct value {name = "Yann"; state = Happy; location = Appartement}	1 pt
Computing apply_action {name = "Benjamin"; state = Happy; location = Appartement} GoToRestaurant	
Correct value {name = "Benjamin"; state = Happy; location = Restaurant}	1 pt
Computing apply_action {name = "Ralf"; state = Hungry; location = Appartement} Eat	
Correct value {name = "Ralf"; state = Happy; location = Appartement}	1 pt
✓ Exercise 2: possible_changes_for_character	Completed, 10 pts
Found possible_changes_for_character with compatible type.	
Computing possible_changes_for_character {name = "Ralf"; state = Happy; location = Restaurant}	
Correct value [Hungry]	1 pt
Computing possible_changes_for_character {name = "Cagdas"; state = Happy; location = Restaurant}	
Correct value [Hungry]	1 pt
Computing possible_changes_for_character {name = "Cagdas"; state = Hungry; location = Restaurant}	
Correct value []	1 pt
Computing possible_changes_for_character {name = "Benjamin"; state = Hungry; location = Appartement}	
Correct value []	1 pt
Computing possible_changes_for_character {name = "Yann"; state = Happy; location = Restaurant}	
Correct value [Hungry]	1 pt
Computing possible_changes_for_character {name = "Cagdas"; state = Hungry; location = Restaurant}	
Correct value []	1 pt
Computing possible_changes_for_character {name = "Yann"; state = Hungry; location = Appartement}	
Correct value []	1 pt
Computing possible_changes_for_character {name = "Gregoire"; state = Happy; location = Appartement}	
Correct value [Hungry]	1 pt
Computing possible_changes_for_character {name = "Yann"; state = Happy; location = Appartement}	
Correct value [Hungry]	1 pt
Computing possible_changes_for_character {name = "Benjamin"; state = Happy; location = Appartement}	
Correct value [Hungry]	1 pt
✓ Exercise 3: describe_state	Completed, 10 pts
Found describe_state with compatible type.	
Computing describe_state Tired	
Correct value "tired"	1 pt
Computing describe_state Tired	
Correct value "tired"	1 pt
Computing describe_state Hungry	
Correct value "hungry"	1 pt

Computing describe_state Tired	
Correct value "tired"	1 pt
Computing describe_state Hungry	
Correct value "hungry"	1 pt
Computing describe_state Hungry	
Correct value "hungry"	1 pt
Computing describe_state Hungry	
Correct value "hungry"	1 pt
Computing describe_state Happy	
Correct value "happy"	1 pt
Computing describe_state Tired	
Correct value "tired"	1 pt
v Exercise 4: tell_action	Completed, 10 pts
Found tell_action with compatible type.	
Computing tell_action Sleep	
Correct value "took a nap"	1 pt
Computing tell_action Sleep	
Correct value "took a nap"	1 pt
Computing tell_action GoToRestaurant	
Correct value "went to the restaurant"	1 pt
Computing tell_action Sleep	
Correct value "took a nap"	1 pt
Computing tell_action GoToRestaurant	
Correct value "went to the restaurant"	1 pt
Computing tell_action Eat	
Correct value "ate"	1 pt
Computing tell_action Eat	
Correct value "ate"	1 pt
Computing tell_action Sleep	
Correct value "took a nap"	1 pt
Computing tell_action Sleep	
Correct value "took a nap"	1 pt
Computing tell_action Eat	
Correct value "ate"	1 pt

[A propos](#)

[Aide](#)

[Contact](#)

[Conditions générales d'utilisation](#)

[Charte utilisateurs](#)

[Politique de confidentialité](#)

[Mentions légales](#)