

Homework 7

GADTs

98-317: Hype for Types

Due: 23 Oct 2018 at 6:30 PM

1 Introduction

In class, we talked about how generalizing ordinary datatypes can allow us to write code that works perfectly and gives us more guarantees at compile time, but also looks really strange. We can write code where branches of a case statement return two different values! In this assignment, you'll look at a couple of pieces of code that involve GADTs (Generalized Algebraic Datatypes) and determine if they typecheck, and if so, what their type is.

Turning in the Homework You should submit a PDF with your solutions to Autolab under GADTs.

2 Types, Types, Types

For the following problems, assume the following two GADTs have been declared.

```
datatype 'a charm =  
  Many : int list -> int charm  
  | Single : int -> unit charm  
  | Empty : 'a charm
```

```
datatype 'a strange =  
  Hidden : 'b -> int strange  
  | Exposed : 'a -> 'a strange
```

In the `strange` type, we've used an *existential type* - one where a type variable appears on the left hand side of the arrow but not the right.

Task 1 Is this expression well typed? If so, give its type. If not, explain why not.

```
fn (x : 'a charm) =>  
  case x of  
    Many _ => []  
  | Single i => i  
  | Empty => ()
```

Task 2 Is this expression well typed? If so, give its type. If not, explain why not.

```
fn (x : 'a charm) =>  
  case x of  
    Many _ => []  
  | Single i => 10
```

Task 3 Is this expression well typed? If so, give its type. If not, explain why not.

```
fn (x : int strange) =>  
  case x of  
    Hidden b => b + 1  
  | Exposed i => i + 1
```

Task 4 Is this expression well typed? If so, give its type. If not, explain why not.

```
fn (x : 'a strange) =>  
  case x of  
    Exposed i => i
```