Contents

- Motivation : why do
- Type domains revi
- Define your own typ
- Constructors
- Usage
- Different from type synonyms !

Motivation

- Types seen so far : cha
- Type constructor
- Why do we need mor
  - Readability
  - Built-in validation
- Example : if a type called "dice" has only six legal values (One
  - f : : dice -> bool is more informative
  - A value that is **not** one of the legal values (One ...

Type Domains Revisited

- We can **define**
- The type bool has jus
- The type [char] has potentially infinite legal values : [], ['A'], ['B'], ['h','
- Pattern-matching allows equality checks against these legal val

Define your own Types

$mybool ::= Mytrue \mid Myfalse$

$MUST\ use\ capitals$!!!!!

$c\ type$")

$f ::$

$f\ True = 34$

$f\ any = 13$

$g :: mybool -> num$

$g\ Mytrue = 34$

$g\ any = 13$

Constructors

- The wrong way :
  - wrong_dice : :
  - also_wrong
- The correct way :
  - dice : := *One* | *Two* | *Three* | *Four* | *Five* | *Six*
- Constructors start with a capital letter
  - Built-in type bool adheres to this rule
  - Built-in types num and char break the rule (for convenience)

Constructors (2)

- Each legal value must have a constructor
- Each legal value may also have some extra data (if so, we specify its type following the constructor) :

x : *tankLevel*

$x = Parttank\ (Gallons\ 3)$

Usage

$$coord3D :== Coord\ (num,\ num,\ num)$$

$$midpoint :: coord3D \rightarrow coord3$$

$$midpoint\ (Coord\ (x1,\ y1,\ z1))\ (Coord\ (x2,\ y2,\ z2))$$
$$= Coord\ (\ (x1\ +\ x2)/2,\ (y1 + y2)/2,\ (z1 + z2)/2\ )$$

Usage (2)

$studentdata ::= StudentByname\ [(char, num)]$
$\quad | \ StudentByregid\ num\ [num]$

$topmarks\ [] = []$

$topmarks\ ((StudentByname\ any\ x) : rest) = (\quad >= 70$

$topmarks\ ((StudentByregid\ id\ x) : rest)$

$= (x : (topmarks\ rest)),\ if\ x >= 70$
$= topmarks\ rest,\ otherwise$

Different from type synonyms


```
syncoord == (num, num, num)
```



```
coord3D == Coord (nu
y :: coord3D
y = Coord (3, 4, 16)
```

Summary

- Motivation : why do
- Type domains revi
- Define your own types !
- Constructors
- Usage
- Different from type synonyms