基本データ型上の関数

胡 振江

東京大学 計数工学科

2007 年 10 月 29 日, 11 月 12 日

Outline

論理型

論理型 (Bool) は True と False だけを含む.

$$\textbf{data}\ \mathit{Bool} = \mathit{False} \mid \mathit{True}$$

論理型上の関数

論理否定

$$
\begin{array}{lcl}
not & :: & Bool \to Bool \\
not\ False & = & True \\
not\ True & = & False
\end{array}
$$

Remark:

- パターンマチング (Pattern matching) による定義
- 書き換え規則 (Rewriting rules)
- $not\ \bot = \bot$

論理型上の関数

論理積・論理和

$$
\begin{aligned}
(\wedge), (\vee) &:: Bool \to Bool \to Bool \\
\\
False \wedge x &= False \\
True \wedge x &= x \\
\\
False \vee x &= x \\
True \vee x &= True
\end{aligned}
$$

Remark:

- $\bot \wedge False = \bot$
- $False \wedge \bot = False$
- $True \wedge \bot = \bot$
- Haskell では、”$\wedge$’ $\Rightarrow$ ”&&”, ”$\vee$” $\Rightarrow$ ”||”

論理型上の関数

論理積の別の定義

$$
\begin{array}{lcl}
(\wedge) & :: & \mathit{Bool} \to \mathit{Bool} \to \mathit{Bool} \\
\mathit{False} \wedge \mathit{False} & = & \mathit{False} \\
\mathit{False} \wedge \mathit{True} & = & \mathit{False} \\
\mathit{True} \wedge \mathit{True} & = & \mathit{True} \\
\mathit{True} \wedge \mathit{False} & = & \mathit{False}
\end{array}
$$

Remark:

- $\bot \wedge \mathit{False} = \bot$
- $\mathit{False} \wedge \bot = \bot$
- $\mathit{True} \wedge \bot = \bot$

論理型上の関数

比較演算子

$$
\begin{array}{lcl}
(==) & :: & Bool \to Bool \to Bool \\
x == y & = & (x \wedge y) \vee (not\ x \wedge not\ y)
\end{array}
$$

$$
\begin{array}{lcl}
(\neq) & :: & Bool \to Bool \to Bool \\
x \neq y & = & not\ (x == y)
\end{array}
$$

Remark:

- Haskell では、"$\neq$" $\Rightarrow$ "/="
- 同値性を定義したい型には論理型だけではなく、他に多くの型がある。論理型上の定義はその一例に過ぎない。

論理型上の関数

比較演算子の定義：class/instance

class $Eq\ \alpha$ **where**
$(==), (\neq) :: \alpha \to \alpha \to Bool$
$x \neq y = not\ (x == y)$

instance $Eq\ Bool$ **where**
$x == y = (x \wedge y) \vee (not\ x \wedge not\ y)$

論理型上の関数

その他の比較演算子の定義

$$\textbf{class}\ \mathit{Eq}\ \alpha \Rightarrow \mathit{Ord}\ a\ \textbf{where}$$
$$(<), (\leq), (>), (\geq) :: \alpha \to \alpha \to \mathit{Bool}$$
$$x \leq y\ =\ (x \leq y) \vee (x == y)$$
$$x > y\ =\ \mathit{not}\ (x \leq y)$$
$$x \geq y\ =\ (x > y) \vee (x == y)$$

$$\textbf{instance}\ \mathit{Ord}\ \mathit{Bool}\ \textbf{where}$$
$$\mathit{False} < \mathit{False}\ =\ \mathit{False}$$
$$\mathit{False} < \mathit{True}\ =\ \mathit{True}$$
$$\mathit{True} < \mathit{False}\ =\ \mathit{False}$$
$$\mathit{True} < \mathit{True}\ =\ \mathit{False}$$

例

xor

$$
\begin{array}{lcl}
xor & :: & Bool \to Bool \to Bool \\
xor\ p\ q & = & (p \wedge not\ q) \vee (not\ p \wedge q)
\end{array}
$$

imply

$$
\begin{array}{lcl}
imply & :: & Bool \to Bool \to Bool \\
imply\ p\ q & = & not\ p \vee q
\end{array}
$$

leap: 閏年を判定する関数

$$
\begin{array}{lcl}
leap & :: & Int \to Bool \\
leap\ y & = & y\ `mod`\ 4 == 0 \wedge \\
 & & imply\ (y\ `mod`\ 100 == 0)\ (y\ `mod`\ 400 == 0)
\end{array}
$$

Outline

自然数型

自然数型はすべての自然数 (0, 1, 2, ...) 含む。

$$\textbf{data}\ \mathit{Nat} = \mathit{Zero} \mid \mathit{Succ}\ \mathit{Nat}$$

Remark:

- *Nat* は再帰的に定義されている。
- *Zero*, *Succ* はデータ構成子である。
- 例：*Zero*, *Succ Zero*, *Succ* (*Succ Zero*)

自然数上の関数定義

加算

$$
\begin{array}{lcl}
(+) & :: & Nat \to Nat \to Nat \\
m + Zero & = & m \\
m + (Succ\ n) & = & Succ\ (m + n)
\end{array}
$$

練習問題：$Zero + Succ\ (Succ\ Zero)$ を評価列を示せ。

自然数上の関数定義

乗算

$$
\begin{array}{lcl}
(\times) & :: & Nat \to Nat \to Nat \\
m \times Zero & = & Zero \\
m \times (Succ\ n) & = & (m \times n) + m
\end{array}
$$

ベキ算

$$
\begin{array}{lcl}
(\uparrow) & :: & Nat \to Nat \to Nat \\
m \uparrow Zero & = & Succ\ Zero \\
m \uparrow Succ\ n & = & (m \uparrow n) \times m
\end{array}
$$

自然数上の関数定義

比較演算

instance *Eq Nat* **where**

$$
\begin{aligned}
Zero == Zero &= True \\
Zero == Succ\ n &= False \\
Succ\ m == Zero &= False \\
Succ\ m == Succ\ n &= m == n
\end{aligned}
$$

instance *Ord Nat* **where**

$$
\begin{aligned}
Zero < Zero &= False \\
Zero < Succ\ n &= True \\
Succ\ m < Zero &= False \\
Succ\ m < Succ\ n &= m < n
\end{aligned}
$$

自然数上の関数定義

比較演算

自然数を次のように定義すれば、比較演算子の定義が自動的に導出される。

$$\textbf{data}\ \mathit{Nat} \ = \ \mathit{Zero} \mid \mathit{Succ}\ \mathit{Nat}$$
$$\textbf{deriving}\ (\mathit{Eq}, \mathit{Ord})$$

自然数上の関数定義

減算

$$
\begin{array}{lcl}
(-) & :: & Nat \to Nat \to Nat \\
m - Zero & = & m \\
(Succ\ m) - (Succ\ n) & = & m - n
\end{array}
$$

減算は部分関数である。

$$
\begin{array}{lcl}
 & & Succ\ Zero - Succ\ (Succ\ Zero) \\
= & & \quad \{ \text{ second equation for (-) } \} \\
 & & Zero - Succ\ Zero \\
= & & \quad \{ \text{ case exhaustion } \} \\
 & & \bot
\end{array}
$$

自然数上の関数定義

階乗

$$
\begin{array}{lcl}
fact & :: & Nat \rightarrow Nat \\
fact\ Zero & = & Succ\ Zero \\
fact\ (Succ\ n) & = & Succ\ n \times fact\ n
\end{array}
$$

Fibonacchi 関数

$$
\begin{array}{lcl}
fib & :: & Nat \rightarrow Nat \\
fib\ Zero & = & Zero \\
fib\ (Succ\ Zero) & = & Succ\ Zero \\
fib\ (Succ\ (Succ\ n)) & = & fib\ (Succ\ n) + fib\ n
\end{array}
$$

Outline

文字型

文字型 *Char* は ASCII (American Standard Code for Information Interchange) 文字の集まりである.

上位3ビット→ ↓下位4ビット	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAC	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF/NL	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

文字を操作する関数

- $ord :: Char \rightarrow Int$: 文字を対応する ASCII 符号の整数に変換

$$ord\ 'b' \Rightarrow 98$$

- $chr :: Int \rightarrow Char$: ASCII 符号の整数を対応する文字に変換

$$chr\ 98 \Rightarrow 'b'$$

- 関係演算子：文字の間は比較できる.

```
instance Eq Char where
x == y = ord x == ord y

instance Ord Char where
x < y = ord x < ord y
```

文字型上の関数の定義

- isDigit: 文字が数字であることを判定する関数.

$$\begin{array}{lcl} \mathit{isDigit} & :: & \mathit{Char} \to \mathit{Bool} \\ \mathit{isDigit}\ x & = & '0' \leq x \wedge x \leq' 9' \end{array}$$

- capitalise: 小文字を大文字に変える関数.

$$\begin{array}{lcl} \mathit{captalise} & :: & \mathit{Char} \to \mathit{Char} \\ \mathit{capitalise}\ x & | & \mathit{isLower}\ x\ = \mathit{chr}(\mathit{offset} + \mathit{ord}\ x) \\ & | & \textbf{otherwise}\ = x \\ & & \textbf{where}\ \mathit{offset} = \mathit{ord}\ 'A' - \mathit{ord}\ 'a' \end{array}$$

評価例

$$
\begin{aligned}
& captalise\ 'a' \\
= & \quad \{ \text{ definition and } isLower\ 'a' = True \ \} \\
& chr(offset + ord\ 'a') \\
= & \quad \{ \text{ definition of offset } \} \\
& chr(ord\ 'A' - ord\ 'a' + ord\ 'a') \\
= & \quad \{ \text{ arithmetic } \} \\
& chr(ord\ 'A') \\
= & \quad \{ \text{ since } chr(ord\ c) = c \text{ for all } c \ \} \\
& 'A'
\end{aligned}
$$

Outline

文字列型

文字列型 *String* は文字の列の集まりである.

$$\{"", "hello", "This\ is\ a\ string.", \ldots\}$$

$$\textbf{type}\ String = [Char]$$

```
?  "a"
"a"
?  "Hello World"
"Hello World"
?  putStr "Hello World"
Hello World
```

文字列上の関数

- $show :: a \rightarrow String$: 任意の型のデータを文字列に変換

$$show\ 100 \Rightarrow "100"$$
$$show\ True \Rightarrow "True"$$
$$show\ (show\ 100) \Rightarrow "\backslash"100\backslash""$$

- $+\!\!+ :: String \rightarrow String \rightarrow String$: 二つの文字列をつなぐ連接演算子

$$"hello" +\!\!+ " " +\!\!+ "world" \Rightarrow "helloworld"$$

- 比較演算子：文字列の比較は通常の辞書式順に従う

Outline

整数型とその上の関数

整数型はすべての整数から構成されている。

Int : single precision integer

Integer : arbitrary precision integer

算術演算子	使用例
$+$ (加算)	$2+3 \Rightarrow 5$
$-$ (減算)	$2-3 \Rightarrow -1$
$*$ (乗算)	$2*3 \Rightarrow 6$
/ (除算)	$3/2 \Rightarrow 1.5$
^ (ベキ乗算)	2^3 $\Rightarrow$ 8
div (整数除算)	*div* 3 2 $\Rightarrow$ 1
	3 ‘*div*‘ 2 $\Rightarrow$ 1
mod (整数除余)	*mod* 5 3 $\Rightarrow$ 2
	5 ‘*mod*‘ 3 $\Rightarrow$ 2

関数定義

- 階乗

$$\begin{array}{lcl} fact & :: & Integer \to Integer \\ fact\ 0 & = & 1 \\ fact\ (n+1) & = & (n+1) * fact\ n \end{array}$$

- 整数の符号を計算する関数

$$\begin{array}{lcll} sign & :: & Int \to Int & \\ sign\ n & | & n > 0 & = 1 \\ & | & n == 0 & = 0 \\ & | & n < 0 & = -1 \end{array}$$

Outline

浮動小数点数型とその上の関数

浮動小数点数型はすべての浮動小数点数から構成されている.

Float : single precision floating-point numbers
Double : arbitrary precision floating-point numbers

演算子	使用例
$+$ (加算)	$2.3 + 3.3 \Rightarrow 5.6$
$-$ (減算)	$2.5 - 3 \Rightarrow -0.5$
$*$ (乗算)	$2.5 * 2.5 \Rightarrow 6.25$
/ (除算)	$3.2/2 \Rightarrow 1.6$

数型上の関数の定義

例： 数の絶対値を返す関数 abs.

$$\begin{aligned} abs &:: Num\ a \Rightarrow a \to a \\ abs\ x &= \textbf{if}\ x < 0\ \textbf{then}\ -x\ \textbf{else}\ x \end{aligned}$$

読みやすいために，次のように書いてもよい.

$$\begin{aligned} abs\ x &\mid x < 0 &&= -x \\ &\mid \textbf{otherwise} &&= x \end{aligned}$$

宿題

- Hugs システムを使って、基本型上の関数をテストする。
- 教科書の第二章を復習し、教科書中の練習問題を解く。