COMPSCI 1JC3

Introduction to Computational Thinking

Fall 2017

04 Logic

William M. Farmer

Department of Computing and Software
McMaster University

September 27, 2017

Admin

- You should post just three brief M&Ms.
- Assignment 1 is due this Friday, September 29.
- Assignment 2 will be posted this Friday and will be due Friday, October 13.
- Continue with your M&Ms.
- Office hours: To see me please send me a note with times.
- Are there any questions?

Advice

1. Don't fall behind!
   - It can be very expensive to catch up after falling behind.
   - Be prepared for the midterm crunch.
2. Use incremental development!
   - You will often not know how to do everything at the start of an assignment.
   - Start with the part you know how to do.
   - Incrementally add each requirement.

Review

1. Patriot missile disaster.
2. Input/output.
3. Writing and executing programs on punch cards.
4. Context of an expression.
5. Abstraction.

What is Logic?

- The study of the principles underlying sound reasoning.
- The branch of mathematics underlying mathematical reasoning.
- The branch of mathematics underlying computing.

What is a Logic?

- A logic is a reasoning system with:
  1. A language with a formal syntax and precise semantics.
  2. Concepts of truth and logical consequence.
  3. A proof system for establishing that statements in the language are true.
- Examples:
  - Propositional logic.
  - First-order logic.
  - Higher-order logic.

Uses of Logic in Computing

- There are many uses of logic in computing.
- Theoretical uses. To study:
  - Computation.
  - Programming languages.
  - Software design.
- Practical uses.
  - To write precise documentation about software artifacts that can be stored and manipulated by computers.
  - To reason about software artifacts, possibly with the help of mathematical software systems like proof assistants.
  - To implement electronic circuits.
  - To provide reasoning facilities in programming languages.
- Note: Software artifacts are by-products of the software development process that include requirements, design documents, programs, and testing plans.

Logic in Haskell

- A form of quantifier-free first-order logic is embedded into Haskell.
- This embedded logic includes:
  - Bool, a type of truth values (called booleans).
  - Boolean functions.
  - Predicates.
  - Conditional expressions.
  - Guarded function definitions.
  - Case expressions.

Booleans

- A boolean is a standard truth value, either true or false.
  - Booleans are named after the English mathematician George Boole (1815–1864).
- In Haskell, the type `Bool` consists of the two boolean values denoted by the literals `True` and `False`.
- A boolean expression (also called a formula) is an expression of type `Bool`.
- Examples of Boolean expressions:
  - `True`, `False`
  - Applications of the boolean functions like `True && False`.
  - Applications of predicates like `1 == 2`.
- Boolean expressions are used to make decisions.

Boolean Functions [1/2]

- A boolean function is a function of a type of the following form: `Bool -> ··· -> Bool -> Bool`
- Haskell has three predefined boolean functions:
  - Negation: not (written as `not` in Haskell).
  - Conjunction: and (`&&`).
  - Disjunction: or (`||`).
- The meanings of these boolean functions are given by the following truth table:

b1	b2	not b1	b1 and b2	b1 or b2
F	F	T	F	F
F	T	T	F	T
T	F	F	F	T
T	T	F	T	T

Boolean Functions [2/2]

- All boolean functions can be constructed using just:
  - `not` and `and`,
  - `not` and `or`,
  - `nand` (Sheffer's stroke), or
  - `nor` (Peirce's arrow).
- The meanings of nand and nor are given by the following truth table:

b1	b2	b1 nand b2	b1 nor b2
F	F	T	T
F	T	T	F
T	F	T	F
T	T	F	F

Another Boolean Function (iClicker)

The meaning of the boolean function `implies` is given by the following truth table:

b1	b2	b1 implies b2
F	F	T
F	T	T
T	F	F
T	T	T

Which of the following boolean expressions is equivalent to `(b1 implies b2)`?

A. `(not b1) and (not b2).`
B. `(not b1) or (not b2).`
C. `(not b1) and b2.`
D. `(not b1) or b2.`
E. `not(b1 and (not b2)).`

Number of Binary Boolean Functions (iClicker)

How many binary boolean functions are there?

A. 5.
B. 8.
C. 16.
D. Infinitely many.

Knowledge about Boolean Functions (iClicker)

Where did you first learn about the boolean functions `not`, `and`, and `or`?

A. Before today I knew very little about them.
B. At home.
C. In high school.
D. In a university course.
E. In a course on propositional logic.

Predicates

- A predicate is a function of a type of the following form:

  t_1 `->` $\cdots$ `->` t_n `->` `Bool`

- Haskell has the following predefined binary predicates:

  `==`, `/=`, `<`, `<=`, `>`, `>=`

- An application of a predicate is a boolean expression.
  - Example: `1 == 2`

Conditional Expressions

- A conditional expression is an expression whose value depends on the value of a boolean expression.
- The Haskell conditional expression has the form:

  `if` c `then` e_1 `else` e_2

- The condition c is a boolean expression.
- Conditional expressions allow code to be skipped in the process of evaluation.

Guarded Function Definitions

- A guarded function definition is a convenient alternative to defining a function using conditional expressions.
- In Haskell it has the form:

$$
\begin{array}{l}
f\ x_1\ \cdots\ x_n \\
\quad \texttt{|}\ g_1\ \texttt{=}\ e_1 \\
\qquad \vdots \\
\quad \texttt{|}\ g_n\ \texttt{=}\ e_n
\end{array}
$$

- The guards $g_1, \ldots, g_n$ are boolean expressions.

Case Expressions

- A case expression, which enables a value to be chosen from a set of options, is a generalization of a conditional expression.
- In Haskell it has the form:

$$
\begin{array}{l}
\texttt{case}\ e\ \texttt{of} \\
\quad p_1\ \texttt{->}\ e_1 \\
\qquad \vdots \\
\quad p_n\ \texttt{->}\ e_n
\end{array}
$$

- The value of the expression e is matched against the patterns $p_1, \ldots, p_n$.