



Saint Vincent College

Department of Computing
and Information Science



Types

Fr. Boniface, OSB
Lecture 16

(some slides modified from Tucker/Noonan, copyright McGraw-Hill, 2006)

“If they ask me, ‘What is his name?’ what am I to tell them?” (Ex. 3:13)

- A *type* is a collection of *values* and *operations* on those values.
- Example: Integer type
 - ▶ values: {..., -2, -1, 0, 1, 2, ...}
 - ▶ operations: {+, -, *, /, <, ...}
- The Boolean type
 - ▶ values: {true, false}
 - ▶ operations: { \wedge , \vee , \neg }

- Limited to finite set (bad for ints, reals)
- (partial) Workaround
 - ▶ Smalltalk - unbounded fractions
 - ▶ C++ - Big Integer library
 - ▶ Haskell - Integer unbounded
- Floating point
 - ▶ fixed, not exact
 - ▶ e.g. $0.2 * 5 \neq 1.0$
 - ▶ not same as real numbers from math

Data = bits + interpretation





Data = bits + interpretation

- Machine data carries no type information.



Data = bits + interpretation

- Machine data carries no type information.
- Basically, just a sequence of bits.



Data = bits + interpretation

- Machine data carries no type information.
- Basically, just a sequence of bits.
- Example:

0100 0000 0101 1000 0000 0000 0000 0000



Data = bits + interpretation

- Machine data carries no type information.
- Basically, just a sequence of bits.

- Example:

0100 0000 0101 1000 0000 0000 0000 0000

- ▶ The floating point number 3.375



Data = bits + interpretation

- Machine data carries no type information.
- Basically, just a sequence of bits.
- Example:
0100 0000 0101 1000 0000 0000 0000 0000
 - ▶ The floating point number 3.375
 - ▶ The 32-bit integer 1,079,508,992

Data = bits + interpretation



- Machine data carries no type information.
- Basically, just a sequence of bits.

- Example:

0100 0000 0101 1000 0000 0000 0000 0000

- ▶ The floating point number 3.375
- ▶ The 32-bit integer 1,079,508,992
- ▶ Two 16-bit integers 16472 and 0

Data = bits + interpretation



- Machine data carries no type information.
- Basically, just a sequence of bits.

- Example:

0100 0000 0101 1000 0000 0000 0000 0000

- ▶ The floating point number 3.375
- ▶ The 32-bit integer 1,079,508,992
- ▶ Two 16-bit integers 16472 and 0
- ▶ Four ASCII characters: @ X NUL NUL

Type Errors



- A *type error* is any error that arises because an operation is attempted on a data type for which it is undefined.
- (Runtime) type errors are common in assembly language programming.
- High level languages reduce the number of type errors (moved to compile time).
- A *type system* provides a basis for detecting type errors.



Static/Dynamic Typing

- A type system imposes *constraints* such as the values used in an addition must be numeric.
 - ▶ *Note: Cannot be expressed syntactically in EBNF.*
- Some languages do *compile-time type checking* (C)
- Others (e.g., Perl) do *runtime type checking*
- Still others (e.g., Java) do both.



Static vs. Dynamic

- A language is *statically typed* if the types of all variables are fixed when they are declared at compile time.
- A language is *dynamically typed* if the type of a variable can vary at run time depending on the value assigned.
- Can you give examples of each?

Strong Typing



- A language is *strongly typed* if its type system allows *all type errors* in a program to be *detected* either at compile time or at run time.
- A strongly typed language can be either statically or dynamically typed.
- *Union types* are a hole in the type system of many languages.
- Most dynamically typed languages associate a type with each value. (Values tagged at runtime.)

Machine-level Details



- Terminology in use with current 32-bit computers:
 - ▶ Nibble: 4 bits
 - ▶ Byte: 8 bits
 - ▶ Half-word: 16 bits
 - ▶ Word: 32 bits
 - ▶ Double word: 64 bits
 - ▶ Quad word: 128 bits

- In most languages, numeric types are *finite in size*.
- So $a + b$ may overflow the finite range.
- Unlike mathematics:
 - $a + (b + c) \neq (a + b) + c$
- Also in C-like languages, the equality and relational operators produce an *int*, not a *Boolean*



Overloading

- An operator or function is *overloaded* when its meaning varies depending on the types of its operands or arguments or result.
- Java: $a + b$ (ignoring size)
 - ▶ integer add
 - ▶ floating point add
 - ▶ string concatenation
- Mixed mode: one operand an int, the other floating point

Type Conversion



- A type conversion is a *narrowing* conversion if the result type permits fewer bits, thus potentially losing information.
- Otherwise it is termed a *widening* conversion.
- Should languages ban *implicit* narrowing conversions?
- Why?