CSE305 S'23 Week 7: Types and What they Represent

From my old notes: TH 5933

Amusing that a 15th-century Latin hymn came into 19th-century English as “O Wondrous Type!” with several relevant names and keywords:

Oh Wondrous Type! Oh vision fair
Of glory that the Church may share...

Alonzo Church applied Bertrand Russell’s “Theory of Types” to the Lambda Calculus, and this was the forerunner of types in programming languages...

Three lines later, the hymn could have replaced “who joy in…” by “who list for…” and had all 3 major compound types. (Nothing like class, however.)

1. Types started out as a way to ensure integrity of machine storage. Not user-definable.
2. Early 1950s: limited range of compound types oriented to specific applications.
4. Type as embracing the abstraction of a mathematical structure.
5. Type as essential unit of modeling objects in hierarchies.
6. Type systems bearing load of applications themselves. "The Vision Fair of Glory."

Example of 6: lecture here last Friday (3/10/23) titled "Type-based reasoning about concurrent programs via logical relations." Speaker from CMU, in lineage of this student of Church.

TH 5933

Russell's Paradox (1900) is about the idea of defining \( y = \{ x : x \notin x \} \). Now ask: is \( y \in y \)? If yes, then by definition of the part in \( \{ \cdots \} \), it means \( y \notin y \). If no, i.e., if you start with \( y \notin y \), then \( y \) should be in \( y \). This contradiction destroyed what we now call "Naive Set Theory Logic." Russell's own way out was to regulate that the predicate \( x \in y \) can be formulated only if we have assigned \( x \) some type \( T \) and \( y \) has type set-of-\( T \). "The First Type Check."
Machine Types

Will focus on one issue thankfully in our rearview mirror, two perennial ones, and one nightmare.

Packed Decimal

The 6-bit word allowed 64 characters, so ONLY CAPS were used and this was "computerlike." The 8-bit byte gave 256 characters, which the ASCII standard mapped as 32 control codes, 96 characters on a (US!) typewriter/terminal, and 128 for international symbols and primitive graphics. But one "fatal instinct" was to pack 2 digits into one byte, rather than use two bytes for a year written like 59, let alone the horrible waste of 4 precious bytes to write 1959 in ASCII.

Problem was: 99 in this scheme wraps to 00 meaning 1900, not 2000. The "Y2K Bug." A worldwide $$ effort to convert legacy software to full ASCII succeeded...too well?

Char and wchar

ASCII enshrined C/C++ char as 8-bit, and char also serves as the 8-bit unsigned integer type. Or rather, char gives the integers modulo 256. UNICODE gives $2^{16}$ characters—-which have not all been filled in yet and are still not enough for all Asian-Pacific scripts. The type wchar_t maps to this. There is also UTF-8, which is not a type per-se but rather a protocol for mixing ASCII and UNICODE.

Word Size

Machines in the early 1980s were still 16-bit. The Sinclair QL in 1984 was a hodgepodge of 8-bit, 16-bit, and its touted 32-bit floats. For a long time we have had a similar mix of 32-bit and 64-bit integer and floating-point types, with both extendable to 128 bits. For C/C++ and some other languages it remains platform-dependent, not language-dependent, which ones int and float and double map to.

Signed and Unsigned

[show chess code]

[Coverage then proceeded to Sebesta's chapter 6 slides, which are linked privately in the pinned Piazza Q&A post.]