Type Compatibility: Ghosts, Readability, and A Missing Rule for Atomic (Updates n3713)

Author: Martin Uecker
Document: n3742
Date: 2026-02-14

Changes since n3713:
- Minor improvement to the rationale
- Removal of Change 3 to avoid collision with other proposals

For defining type compatibility, several paragraphs use "shall" in a semantics section, i.e. outside of a constraints section, without expressing a requirement and without implying undefined behavior (cf 4 Conformance). For qualifiers, array types, and function types, a change was already proposed in N3484 and adopted into C2y in Graz, but a similar change is missing for pointer types. This is fixed by the proposed Change 1a and Change 1b below. Proposed Change 1b provides alternative wording that consistent with similar rules elsewhere in the standard removes the rules about qualifiers that are already specified as a general rule in 6.7.4.1§11 and are therefor redundant at this point.

When investigating this issue, it was also noted that we miss a corresponding rule for atomic types, which seems to be an omission. I am not aware of any compiler that would not treat atomic versions of compatible types as compatible. Such a rule is added by proposed Change 2.

Proposed Wording (N3783)

Wording Change 1a (pointers)

6.7.7.2 Pointer declarators

Semantics

2 ~~For~~ **T**wo pointer types ~~**to be**~~ **are** compatible~~,~~ **if and only if** both ~~**shall be**~~ **are** identically qualified and both ~~**shall be**~~ **are** pointers to compatible types.

Wording Change 1b (alternative to 1a)

6.7.7.2 Pointer declarators

Semantics

2 ~~For~~ **T**wo pointer types ~~**to be**~~ **are** compatible~~,~~ **if and only if** both ~~**shall be identically qualified and**~~ ~~**shall be**~~ **are** pointers to compatible types.

Wording Change 2 (missing rule for atomic)

6.7.3.5 Atomic type specifiers

Two atomic types are compatible if and only if the corresponding non-atomic types are compatible.