is not a (possibly) cv-ref-qualified `Person`, `enable_if` will not ~~define~~ the `type` member **`typedef`**, leading to a failure during the substitution process. Rather than being a compile-time error, such substitution failure will just remove `addPerson` from the overload set being considered, hence the term “substitution failure is not an error,” or SFINAE. If a client attempts to pass a non`Person` as an argument to the `addPerson` function, the compiler will issue an error that there is no matching function for call to `addPerson`, which is exactly the result we want.

Putting this all together means we get to call `addPerson` with *lvalues* and *rvalues* of type `Person`, and the value category will be appropriately usable within `addPerson`, generally with use of `std::forward` within that function’s definition.

See Also

- “**`auto`** Variables” (§2.1, p. 195) covers a feature that can introduce a forwarding reference with the **`auto`**`&&` syntax.
- “*Rvalue* References” (§2.1, p. 710) details how *rvalue* references can be confused with forwarding references due to similar syntax.
- “Variadic Templates” (§2.1, p. 873) explores how variadic templates are commonly used in conjunction with forwarding references to provide highly generic interfaces.

Further Reading

- Scott Meyers provides valuable insight on how to spot a forwarding reference as opposed to the **rvalue reference** (see Section 2.1.“*Rvalue* References” on page 710) in **meyers15b**, “Item 24: Distinguish universal references from rvalue references,” pp. 164–168.
- The original authors of this feature explain (1) that they “intentionally overloaded the `&&` syntax with this special case” (p. 2) but didn’t give it a distinct name, (2) why a distinct name would be useful in the standard itself, and (3) why they prefer the term forwarding reference over **universal reference** in **sutter14b**.
- Eric Niebler drills down on a pernicious pitfall involving the overloading of functions having a forwarding reference parameter — particularly when the function is a single argument constructor that can be used as a copy constructor — in **niebler13**.