Corner cases of function template argument matching

There are cases in which a template function could be written that can never be called, whether with explicit template parameters or by relying on template argument deduction:

```
template <typename... Ts, typename T>
void odd1(Ts... values);
```

By the Rule of Greedy Matching applied to `Ts`, `T` can never be specified explicitly. Moreover, `T` cannot be deduced either because it's not part of the function's parameter list. Hence, `odd1` is impossible to call. According to standard C++, such function declarations are **ill formed, no diagnostic required (IFNDR)**. Current compilers do allow such functions to be defined without a warning. However, any conforming compiler will disallow any attempt to *call* such an ill-fated function.

Another scenario is one whereby a variadic function can be instantiated, but one or more of its parameter packs must always have length zero:

```
template <typename... Ts, typename... Us, class T>
int odd2(Ts..., Us..., T);  // specious
```

Any attempt to call `odd2` by relying exclusively on template argument deduction without any explicit template arguments will force both `Ts` and `Us` to the empty list because, by the Rule of Fair Matching, neither `Ts` nor `Us` can benefit from template argument deduction. So calls with two, three, or more arguments fail:

```
int x2a = odd2(1, 2.5);           // Error, Ts=<>, Us=<>, too many arguments
int x2b = odd2(1, 2.5, "three");  // Error, Ts=<>, Us=<>,  "    "      "
```

However, there seem to be ways to invoke `odd2`, at least on contemporary compilers. First, calls using deduction with exactly one argument will merrily go:

```
int x2c = odd2(42);  // Ts=<>, Us=<>, T=42
```

Moreover, functions that pass an explicit argument list for `Ts` also seem to work:

```
int x2d = odd2<int, double>(1, 2.0, "three");
    // Ts=<int, double>, Us=<>, T=const char*
```

The call above passes `Ts` explicitly as `<int, double>`. Then, as always, `Us` is forced to the empty list, and `T` is deduced as `const char*` for the last argument. That way, the call goes through!

Or does it? Alas, the declaration of `odd2` is IFNDR. By the C++ Standard, if all valid instantiations of a variadic function template require a specific template parameter pack argument to be empty, the declaration is IFNDR. Although such a rule sounds arbitrary, it does have a good motivation: If all possible calls to `odd2` require `Us` to be the empty list, why is `Us` present in the first place? Such code is more indicative of a bug than of a meaningful intent. Also, diagnosing such cases might be quite difficult in the general case, so