P3506R0 2024-11-19 Reply-To: gdr@microsoft.com
Audience: EWG

P2900 Is Still Not Ready for C++26

Gabriel Dos Reis
Microsoft

In the pre-Tokyo meeting paper P3172R0, Microsoft expressed concerns regarding the design of the "Contracts" facility as presented in P2900R6 and identified several key areas that needed further work, in particular: the treatment of "safety" (as in "functional safety"), undefined behavior in the evaluation of contract predicates, lack of support for dynamic dispatch and indirect calls, lack of use of Contracts in the Standard Library implementation. That assessment led Microsoft to recommend against the inclusion of P2900R6 in C++26. Since then, SG21 has worked to address some of the issues. The conclusion of this his paper and the recommendation contained therein are based on P2900R11.

1 Continued Concerns

1. The treatment of undefined behavior as they pertain to the evaluation of contract predicates remains inadequate and the conclusion of P3172R0 remains the same: the facility is not viable without adequate limitation on the outcome of evaluation of contract predicates.
2. The solution adopted for contracts on virtual functions is too complex and departs from decades-old typical uses of virtual functions. A virtual function defined in a base class typically expresses an interface, and all its implementations (overriders) shall, at the minimum, satisfy the requirements expressed in the interface declaration. The new addition probably needs further simplification and definitely needs deployment in the field.
3. The problem of contracts on pointer to functions remains unsolved, and punted to later, if ever.
4. Support for contracts on coroutines was added, and the treatment seems acceptable. Furthermore field deployment/experiments are needed.
5. The experience report regarding the use of the facility in an implementation of the Standard Library essentially replaces existing assertions (already shipped in the implementation without Contracts) with contract_assert. While that is notable, it however falls short of what would be needed to start building confidence: the experience report that does not all use pre- or post-conditions, the main novelty of the Contract facility. Consequently, this viability concerns regarding field experience remains unaddressed.

2 Contract Predicate Evaluation and Exceptions

In addition to those concerns, another aspect of the design of P2900 remains seriously concerning: the evaluation of a contract predicate forcefully functionally converts any exception that might arise from the evaluation into a value "false" indicating failure to satisfy a contract. That design is actively harmful in many ways:

- It introduces an unnecessary overhead in terms of binary code size, and runtime costs on some popular platforms (e.g. x86) by wrapping contract predicate evaluation in try/catch blocks.
- It hinders composition as the only workaround for a programmer to let the exception to naturally propagate is to install a handler that filters on those exceptions and rethrow them. At least, that was the workaround suggested at the Tokyo meeting. That is an unacceptable constraints in many environments where contracts would be naturally useful.

3 CONSTIFICATION MUST GO

Constification was invented as a way to prevent side-effects in contract predicates. Alas, it caused more confusions as it was only a partial solution that created its own set of problems. The "patch" to that was to do even more constification, extending to global variables. A better solution is: (1) remove constification; (2) apply a solution along the lines of conveyor functions. Whatever we do, constification must go; it is not the solution.

4 CONCLUSION

In light of these issues, Microsoft continues to recommend against the inclusion of P2900 in C++26. The facility needs further work, and deployment experience.