Axioms should be assumable: a minimal fix for contracts

Timur Doumler (papers@timur.audio)
Ville Voutilainen (ville.voutilainen@gmail.com)

Document #:	P1812R0
Date:	2019-07-14
Project:	Programming Language C++
Audience:	Evolution Working Group, Core Working Group

Abstract

Recent papers propose to introduce a global assumption mode, and to make axioms unevaluated contexts. We add one crucial tweak: the assumption of axioms should not be affected by such an assumption mode. The result is a minimal fix for contracts that solves three problems at once.

1 The problems

The contracts wording in the current C++20 working draft [N4820] has three significant problems:

1. Unchecked contracts may always be assumed (i.e. if their predicate is false, behaviour is undefined). This is often dangerous and makes it more difficult to add contracts to an existing codebase.
2. It is unspecified whether unchecked contracts are evaluated; this precludes the usage of predicate functions with no definition.
3. Contracts cannot be used for portable optimisation hints, i.e. predicates that may always be assumed and are never evaluated, even though providing them was a design goal [P1773R0].

There are currently proposals addressing each problem separately. Here, we propose a solution that fixes all three problems at once.

2 Related proposals

Problem 1 is being addressed by [P1710R0] and [P1730R0], which propose to add a global *contract assumption mode* that turns the semantics of all unchecked contracts from assumed to ignored, and making ignored the default.

Independently, Problem 2 is being addressed by [P1704R0], which makes axioms unevaluated contexts. If only [P1704R0] were applied to the current working draft, with no other changes, then it would actually solve Problem 3, too: it would make axioms always assumable and never evaluated, such that they could be used as portable optimisation hints.

Unfortunately, when we add the global assumption mode, it takes that benefit away, by making axioms ignored by default, and only offering a way to enable assume semantics globally. It removes from the language the possibility to have a contract that is specified, in C++ code, to have assume semantics.

3 The solution

The solution to all three problems is to merge [P1730R0] and [P1704R0], but with one extra rule: *axioms may always be assumed*, and are not subject to the global assumption mode. This makes axioms useable as portable optimisation hints, in addition to solving the other problems addressed by [P1730R0] and [P1704R0].

Additionally, we propose to make the default assumption mode implementation-defined, as opposed to off by default. This removes an unnecessary restriction and makes our proposal compatible with [P1769R0].

4 Discussion

At first glance, it might seem that a special rule for axioms for the global assumption mode could make the contracts facility more complicated and harder to teach. However, this is not the case: axioms are already special if we make them unevaluated contexts. Making them always assumable does not create a new special case. If anything, it makes axioms more consistent: they are contracts that may always be assumed and are never evaluated.

Assigning these clear semantics to axioms also brings them in line with the commonly accepted meaning of the word "axiom", removing a source of great confusion [P1672R0].

Further, always assuming axioms is in line with the future evolution of contacts — we are not closing off any design space.

Finally, a common misunderstanding is that always assuming an axiom would somehow *mandate* that the compiler optimise based on the assumption. This is not the case; for `-O0`, an implementation will probably not act on the undefined behaviour introduced by an axiom, so there is no impact on the ability to debug code.

5 Design change or minimal fix?

We believe that a superior option would be to redesign contracts such that they offer literal semantics, as proposed in [P1429R2] or [P1607R0]. This would not only solve the three problems above, but also provide a flexible and coherent framework for many other use cases of contracts not discussed here. However, at this late stage it might not be possible to get a significant redesign of contracts for C++20. In this case, if only "minimal fixes" are allowed, we believe the fix proposed here is the best solution possible.

6 Proposed wording

The wording below merges the wording from [P1730R0] and [P1704R0] and modifies the result in such a way that it achieves the effects described in section 3. The proposed changes are relative to the current C++20 working draft [N4820].

In [dcl.attr.contract.check] paragraph 4, modify as follows:

> A translation may be performed with one of the following contract assumption modes: off or on. During constant expression evaluation ([expr.const]), only predicates of checked contracts are evaluated. In other contexts, it is unspecified whether the predicate for a `default` or `audit` level contract that is not checked under the current build level is evaluated; if the predicate of such a contract would evaluate to `false` and contract assumption mode is on, the behavior is undefined.

Add paragraph at the end:

The predicate `p` of a contract condition with `axiom` *contract-level* is an unevaluated operand. If an implementation is able to determine what the value would be returned by evaluating `p`, and this value is `false`, the behavior is undefined. [*Example:*

```
bool pred(int i);  // never defined

void f(int * p)
  [[expects axiom: p && pred(*p)]];

void g(int * p)
{
  f(p);
  if (p != nullptr) // the check can be elided
    *p = 0;
}
```

An implementation can eliminate the check `p != nullptr` in function `g`. If `p` was null, the precondition of `f` would evaluate to `false`, which would be undefined behavior. The potential contract violation can be determined without the call to function `pred` owing to the semantics of operator `&&`. — *end example*]

Acknowledgements

Many thanks to Gašper Ažman for his comments on early drafts of this paper.

References

[N4820] Richard Smith. Working Draft, Standard for Programming Language C++. http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2019/n4820.pdf, 2019-06-17.

[P1429R2] Joshua Berne and John Lakos. Contracts That Work. http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2019/p1429r2.pdf, 2019-06-14.

[P1607R0] Joshua Berne, Jeff Snyder, and Ryan McDougall. Minimizing Contracts. http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2019/p1607r0.pdf, 2019-03-10.

[P1672R0] Joshua Berne. "Axiom" is a false friend. http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2019/p1672r0.pdf, 2019-06-16.

[P1704R0] Joshua Berne and Andrzej Krzemieński. Undefined functions in axiom-level contract statements. http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2019/p1704r0.html, 2019-06-16.

[P1710R0] Ville Voutilainen. Adding a global contract assumption mode. http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2019/p1710r0.html, 2019-06-17.

[P1730R0] Hyman Rosen, John Lakos, and Alisdair Meredith. Adding a global contract assumption mode. http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2019/p1730r0.md, 2019-06-14.

[P1769R0] Ville Voutilainen and Richard Smith. The "default" contract build-level and continuation-mode should be implementation-defined. http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2019/p1769r0.html, 2019-06-17.

[P1773R0] Timur Doumler. Contracts have failed to provide a portable "assume". http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2017/p1773r0.pdf, 2019-06-17.