“Axiom” is a False Friend

Document #:	P1672R0
Date:	2019-06-16
Project:	Programming Language C++
Audience:	EWG

Reply-to: Joshua Berne <jberne4@bloomberg.net>

Contents

Abstract

The name axiom was assigned to contract level for checks that cannot be executed at runtime. This has led to a great deal of extra meaning being applied due to the false friendship with the classical definition of the word. Independently of whatever specific behaviors this level might have, we propose changing the name to absolutely any other word.

1 Overview

The consolidated contracts proposal, [P0380R1], introduced the `axiom`, `default`, and `audit` contract level with the following warning:

> We choose something short and again we assume that people will have to learn the exact meaning whatever names we use, so that “better names” could become “false friends” when people more often thought they understood them without looking up their meaning.

Recent discussions have shown that this false friendship between the `axiom` contract level and the word axiom has become fully reified, and we think this will cause great confusion if this were to persist.

[P0380R1] identifies the level that it chooses to name axiom as being the level with “No runtime checking”. The primary motivating example of an axiom level predicate presented in that paper was for requiring that a pair of input iterators passed to an algorithm be a valid range - something that is “impossible to execute as a side-effect-free contract predicate”. Elsewhere the level is defined as “a formal comment for humans and static analyzers; no run-time cost because the predicate is not

evaluated at run time" and the existence of the level was motivated with the statement "The axiom level was deemed essential by people wanting more precise comments and/or using contract systems augmented by static analysis."

Wikipedia defines axiom in the following way:

> An axiom or postulate is a statement that is taken to be true, to serve as a premise or starting point for further reasoning and arguments. The word comes from the Greek axíōma (ἀξίωμα) "that which is thought worthy or fit" or "that which commends itself as evident."

In the context of programming, this carries a heavy implication to many users of "does not need proof", or alternately, "can be assumed". Others in the discussion when not referencing this definition would still refer to axiom-level contracts as "fundamentally true" or call out axioms as a tool that can be used as a portable replacement for `__builtin_assume` due to their fundamentally true nature.

Practitioners of C++ will come to know the meaning of this level based entirely on how the standard specifies it. The identifying factor for this level is just that it is never checked at runtime – very much in line with the definition provided in the original proposal. The mathematical model of treating this level as a more fundamental truth will not match how the feature is used in practice, and designing with that model in mind leads to extensive conflict.

2 The Danger of False Friendship

The discussions about the expected behavior of contracts of this level have often come to an impasse due to conflicts between these two definitions for the level - one that was proposed, and one that has been adopted as a "false friend" because of the chosen name.

Consider a simple example where one might choose to use an `is_dereferencable` function declaration that is provided by your vendor to diagnose cases where a pointer to an already deleted object is passed around:

```
int f(int *p)
  [[expects axiom: is_dereferencable(p)]]
{
  // ...
}
```

Because of the use of a function that cannot be implemented, but which the compiler can understand, this predicate must use the "No runtime checking" contract level. If this level is interpreted as "A predicate that can be taken to be true without proof" then it brings with it very unfortunate corollaries - that `p` is not `nullptr`.

As with all contract checks, these issues have no impact at all if the predicate itself is not violated. The issue arises when a check like this is violated deep within a larger program and one has to identify what has happened. Consider filling out the body of the function to gather non-null pointers into a `std::vector`:

```
static std::vector<int *> not_null_pointers;
int f(int *p)
  [[expects axiom: is_dereferencable(p)]]
{
  if (p != nullptr) {
    ps.push_back(p);
  }
}
```

The presence of the contract check, which one hopefully added to document requirements and make this a safer piece of code, also elides the need to check for `p` being `null`. The poor programmer looking at this function to determine how a `nullptr` got into the static list of `int*`s is going to look at this function and easily incorrectly conclude it can't be the source of bad values.

Fundamentally, just because something can't be proven doesn't mean taking it as a given truth is going to be a safe thing to do and an easy thing to reason about.

3 Alternatives

We believe an alternative word that meets the following criteria should be chosen:

- Indicates that the predicate will never be evaluated.
- Does *not* indicate that the predicate will be considered inherently true.

The reason why it is important to pick a name that meets both of these criteria is to avoid a repeat of the false friendship we have today with "canonical truth" which brings with it confusion and pitfalls – otherwise we are no better off than we are today.

[P0380R1] noted the following alternatives all of which seem to meet both of our criteria:

- annotation
- comment
- compile time
- not-runtime
- static

Alternatives that have been proposed on the reflector or in private discussions include the following:

- abstract
- conjecture
- hope
- hypothesis
- hypothetical

- lemma
- nocheck
- posit
- speculate
- theorem
- unavailable
- unchecked
- unevaluated

Or these alternatives that seem to overly convey the "assumed to be true" meaning and would be functionally too synonymous to axiom:

- faith
- ideal
- transcendent

The following seem, in our opinion, like the best of the proposed choices, and we believe any of these would take us away from the confusion we have today:

- nocheck
- unchecked
- hope

One more alternative that should be considered if no consensus can be reached on a less misleading name for this contract level is to just remove the level entirely. Formal comments can serve the purposes of static analyzers as they always have until a better compromise can be reached.

4 References

[P0380R1] G. Dos Reis, J. D. Garcia, J. Lakos, A. Meredith, N. Myers, B. Stroustrup, *Support for contract based programming in C++*
http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2018/p0542r5.html