"Defined and Continuous" or "Discontinuous" To Be or Not To Be

Nate Hayes
Sunfish Studio, LLC

October 3, 2010

Abstract

The definition of a decoration bit that verifies continuity is currently on the table for debate and discussion in the P1788 forum. This paper proposes that P1788 should standardize this decoration bit in the affirmative, i.e., "defined and continuous," as opposed to the counter-affirmative, i.e., "discontinuous."

1 Introduction

This paper is a proposal to amend a proposal. Currently on the table for debate and discussion in the P1788 forum is a motion to propose a "discontinuous" decoration bit [4].

It is hereby respectfully submitted that P1788 should instead standardize the bit in the affirmative as a "defined and continuous" decoration.

2 Proposed Amendment

The motion [4] shall be amended to propose standardization of a "defined and continuous" decoration bit instead of the currently proposed "discontinuous" decoration bit as follows:

- Clause 3.1.1 of [4] provides a "defined and continuous" predicate $C(f; X)$ for a real function f and interval box X, and this shall not be changed except to note the "defined and continuous" bit aims to track this predicate.
- Line 5 of Clause 3.1.2 shall be amended to remove the logical negation ($\neg$) sign before the C predicate.
- Line 7 of Clause 3.1.2 shall be amended to change the logical disjunction ($\vee$) signs to logical conjunction ($\wedge$) signs.

- Line 6 of Clause 3.1.3 shall be amended to change the logical disjunction ($\vee$) sign to a logical conjunction ($\wedge$) sign.
- The remainder of the motion text shall be amended as deemed necessary by the good judgement of the author so as to be consistent and logically coherent with the above changes.

3 Rationale

It is acknowledged that individuals have their own personal taste and style. For example, when writing a software program it may be largely a matter of taste wether one should write

```
if ( isDiscontinuous( X ) ) // Handle "bad thing" here
```

or

```
if ( !isDefinedAndContinuous( X ) ) // Handle "bad thing" here
```

It should also be recognized that both the proposed motion and respective amendments are entirely Level 1 and Level 2 definitions, so regardless of the outcome vendors and implementors will likely have the freedom at Level 3 and Level 4 to choose any internal representation they wish, so long as the overall behavior ultimately conforms to the Level 1 and Level 2 definitions and requirements.

It is therefore suggested that the focus of the debate should be centered on principles of *uniformity* and *consistency* in the "public interface." The purpose to do this is to present in the end a standard that exposes to the rest of the world a design that has rational and consistent justifications for its choices.

When examining issues of consistency, the view of this paper is that "defined and continuous" is the choice that can be best supported.

3.1 Reasons for "discontinuous"

One of the good arguments I've heard in favor of "discontinuous" is that the decoration bit is set to "true" when the "bad thing," i.e., a discontinuity, occurs, and that this methodology is consistent with practices in IEEE 754 wherein exceptional conditions such as "underflow" or "overflow" are flagged in the affirmative.

By itself, this concept has merit. However, when taken into consideration alongside all the reasons of consistency in favor of "defined and continuous" (explained next), the merit of this choice is overshadowed and superseded, in my opinion.

3.2 Reasons for "defined and continuous"

One of the primary points of emphasis in [4] is that of simplicity and consistency. This is emphatically underscored (in all its appropriateness) by the quote from George Corliss:

> If folks of OUR experience have trouble understanding, God help the casual user! We have been, and we must continue to be, sensitive to KISS. I guess one path to simplicity is a very carefully worked-out, **consistent**, and coherent level model. That is WE work very hard so that the result is easy. I'm OK with that, as long as the result is easy.

The bolded emphasis is mine. In my view, these are some good reasons why considering if a decoration bit is to be "defined and continuous" or "discontinuous" should not be left up to matters of chance or personal taste. It is also why I don't see it should be viewed as useless "nit-picking" to raise such a fuss about what might otherwise be seen by some outsider as quite a minor or insignificant topic. So it is in this spirit of George's comments why the whole question of what this decoration bit represents is a serious subject to me, and why the proposed amendment is part of my own contribution to what I see as us working "very hard so that the result is easy."

As illustrated already in section 3.1.1 of the motion, the "natural" Level 1 predicate of the decoration bit in question is "defined and continuous," wherein propagations or "accumulations" of this bit are performed by logical conjunction ($\wedge$). The Level 2 definition proposed in Line 5 of section 3.1.2 introduces an extra (unnecessary) logical negation ($\neg$), and this in turn requires propagation of the decoration bit by logical disjunction ($\vee$). These facts alone are evidence the definitions are not the "most simple" and that KISS is not being followed. From a mathematical perspective there is no reason of necessity or justification for any of this, so in my view it can lead the uninitiated to scratch their heads and ask "why did they define it that way?" or "am I missing something important here?" or "is there really some important reason for that?" Even as an initiate, this was *my* initial response. I took away from this experience a belief that it will be enough to confound other users, implementors, and authors alike.

The proposed motion is not entirely consistent about the use of "discontinuous" and switches back-and-forth at times, which I think is further evidence that "discontinuous" is not truly the "most simple" or "natural" choice even for purposes of pedagogy.

The "defined and continuous" property is an important concept in mathematics, and many fundamental theorems are predicated on the existence of this property. Students are taught this in Calculus. For example, the following is a definition of the Mean Value Theorem from a typical college textbook [3]:

Theorem 1 (Mean Value Theorem) *If f is **defined and continuous** on $[a, b]$ and differentiable on (a, b), then there exists a number c, with $a < c < b$,*

such that

$$f'(c) = \frac{f(b) - f(x)}{b - a}.$$

In other words, $f(b) - f(a) = f'(c)(b - a)$.

The bolded emphasis is mine. Note that the theorem is not given in terms of double negatives, e.g., "if f is not discontinuous..." What, therefore, is the motivation for P1788 to not follow these pedagogical conventions? In my view, this is an important choice of consistency. I think it would be inconsistent for P1788 to promulgate "discontinuous" as a pedagogical tool when just the opposite is already the norm. It seems this would require further rationale, explanation, and justification (which is not present in the current motion). But this would stray even further from the KISS principle.

Another relevant concern relates to exception-handling semantics found in many popular programming languages. In C++, for example, there is the "try-catch" exception-handling mechanism. The "natural" form of these statements is that the user will "try" some lengthy computation and code in the "catch" clause will automatically be invoked by the run-time system if and only if some exceptional condition is encountered, i.e.:

```
try {
    // User begins some lengthy computation...
    // ...and the end of the "try" block gets reached
    // only if there are no exceptional conditions
} catch ( ... ) {
    // This code will get invoked by the run-time system
    // if and only if an exceptional condition occurs
}
```

Implicit in this "try-catch" mechanism is an assumption that the "good thing" occurs in the "try" block and the "bad thing" occurs in the "catch" block. If P1788 standardized "discontinuous," this mechanism will still work. However, users and implementors should then think in terms of double-negatives akin to the "if f is not discontinuous..." example given above. In other words, users and implementors will have to change their thinking and consider that "not the bad thing" occurs in the "try" block. This is still functional, but not the "most simple," and almost certainly not what users are used to. The reason I believe this particular example doesn't fall into the category of a "personal preference" is that like the Mean Value Theorem example described above, these types of exception-handling mechanisms (and all the assumptions that go along with them) are already institutionalized and pervasive.

Perhaps the biggest argument in favor of consistency, however, is that P1788 should be consistent with itself. Viewed in terms of the multi-valued logic of [1], the domain tetrit in [2] propagates via logical conjunction ($\wedge$), or *infimum*. The proposed "discontinuous" bit, however, would propagate via logical disjunction ($\vee$), or *supremum* (as indicated in sections 3.1.2 and 3.1.3).

This inconsistent state of affairs would be the result of the proposed definition for "discontinuous," which tracks a "bad thing," as opposed to the definition of the domain tetrit, which tracks a "good thing," e.g., "is the function defined for its operands." The net effect would be a P1788 standard that uses opposite logical conventions to define and propagate two different decorations... the ultimate sin of inconsistency, in my view.

Some might ask the question: "why not simply negate the definition of the domain tetrit, then, so it will be consistent with the discontinuous bit?" This is a very good question, and I have also considered it. But my answer is simple: because it makes a bad situation worse. There are many arguments of consistency against the "discontinuous" bit already, and the same arguments apply to negation of the domain tetrit as well.

The advanced reader might also have noticed that simply "negating" the Level 1 definition of a domain tetrit does not really change its definition! If propagating a tetrit via logical disjunction ($\vee$) or *supremum* is really desired, this might require using universally-qualified propositions in the Level 1 definition instead. When originally writing Motion 18, it was an intentional choice for me not to do this. The reason to avoid universally-qualified propositions was because they can cause vacuously true statements when an empty set is involved, and discussions in the P1788 forum have shown that the subject of vacuous truth can be tricky even for experts, lest the uninitiated or less experienced. So it was adherence to the KISS principle that made me choose existential quantifications for the domain tetrit in the first place.

A final argument that has been made along these lines is to change the tetrit ranking priority value R to $3 - R$ and then take the supremum of two tetrit values. The domain tetrit will then be consistent with the discontinuous bit (which uses disjunction). The reason not to do this is because the IEEE 1788 definition of a tetrit will then no longer be compatible with the proposed C++ bool_set [1] standard. For example, the supremum of two tetrits is:

sup	0	1	2	3
0	0	1	2	3
1	1	1	2	3
2	2	2	2	3
3	3	3	3	3

but the disjunction ($\vee$) of two bool_set values is:

$\vee$	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	2	3
3	0	3	3	3

Only the infimum between two tetrits is the same as the conjunction ($\wedge$) between two bool_set values, i.e., in both cases the result is:

i/c	0	1	2	3
0	0	0	0	0
1	0	1	1	1
2	0	1	2	2
3	0	1	2	3

So if P1788 wishes the following to be true:

- The propagation of all decorations shall be handled uniformly, i.e., with conjunction/infimum or disjunction/supremum (but not some wild combination of both)
- The IEEE 1788 definition of a tetrit shall be compatible with the proposed C++ bool_set standard [1]

Then the only choice is to accept the current definition of a tetrit [2] and standardize "defined and continuous" as opposed to "discontinuous."

References

[1] Bronnimann, H. et. al., "Bool_set: multi-valued logic," http://www.open-std.org/JTC1/sc22/wg21/docs/papers/2006/n2046.pdf

[2] Hayes, N., "Trits to Tetrits," P1788 Motion 18, May 28, 2010.

[3] Hughes-Hallet, Gleason, McCallum, et. al., "Single Variable Calculus, Third Edition," John Wiley & Sons, Inc., 2002.

[4] Pryce, J., "Motion to propose a "discontinuous" decoration bit, Version 3," Sept. 11, 2010.