1. Comparison Predicates

The IEEE 754 Standard [1,2,3] defines multiple comparison predicates for floating point numbers. The six conventional comparison predicates are <, >, <=, >=, ==, and **!=**. These are familiar to those who use one of the many programming languages designed with readability in mind. To those who use Fortran, they would be more familiar with the mneumonics **.lt.**, **.gt.**, **.le.**, **.ge.**, **.eq.**, and **.ne.**. Except for those last two predicates, equality and inequality, i.e. == and **!=** respectively , these predicates signal (or raise) an IEEE 754 *INVALID* exception when a **Not-a-Number** or **NaN** is compared with itself or anything else. Where a predicate (such as == or **!=**) does signal (or raise) an exception, it is termed **signalling**, and when it does not, it is termed **silent**. Further predicates are defined by IEEE 754, many of which are silent when working with NaNs. In C, they are provided through the far more clumsy mechanism of macros, like **islessgreater**(3) or **isless**(3). Some languages provide none of these extra predicates.

These predicates have been discussed at length [4,5] after the standard was produced by the driving force behind the IEEE 754 standard, Professor William Kahan. In a 2002 paper [5], he tabulated a subset of the standard's predicates on the basis that he had *"yet to need more than the fourteen tabulated"* therein. A close replica of that follows.

In Kahan(2002)	yes	yes	yes	yes	yes	**NO**	yes	yes	yes	yes
Math:	=	≠	?	!?	<>	! <>	<	≤	≥	>
C signalling:	-	-	-	-	<>	**...**	<	<=	>=	>
C silent:	==	!=	**?**	!?	!=?	**...**	!>=	!>	!<	!<=

Some may argue that the subset is too strict. To expand this subset somewhat, an extra column is added to the table for the *neither-less-nor-greater-than* predicate, this addition also making it more obvious from where Kahan got the **!=?** symbol seen above

2. Chapel Implementation

Chapel is ideally placed to provide the first complete, symbolic, implementation of this Kahan subset of this most practically useful subset of IEEE 754 comparison predicates. To achieve this, the above table has been replicated and augmented as follows:

1. The IEEE 754 standard usage a single question mark (**?**) to query whether two numbers were *unordered* with respect to each other has been replaced with the **??** (double question mark) symbol instead. This resolves the conflict presented by Chapel's own usage of a single **?** for other purposes. The net result of this choice is that the **unordered/ordered** predicate pair of

   ?? / !?

   is now perfectly **anti**-symmetric with the **equality/inequality** predicate pair of

   == / !=

   This would seem like an improvement or at the very least, a justification of **??**.

2. The extra column added for *being neither less than nor greater than* will be populated with symbols from the IEEE 754 standards of 1985 [1] and 2008 [2]. These symbols have disappeared in the latest release [3] of that standard for reasons which are currently being investigated.

But this one change and one addition are the only variations to Kahan's list of predicates.

Note that during the 1990s, Kahan used the symbol **!>=<** as the **unordered** predicate, although by 2002 he had reverted back to using a single question mark. The negation of that, i.e. an ordered predicate using the symbol >=< seems hard to find, but its existence is assumed - somewhere.

Using the two points mentioned earlier, and incorporating Kahan's own variation on the (un)ordered predicates, the earlier table can now be written as:

New Column:	no	no	no	no	no	**yes**	no	no	no	no
Math:	=	≠	?	!?	<>	! <>	<	≤	≥	>
Chapel signalling:	-	-	-	>=<	<>	**=?**	<	<=	>=	>
Chapel silent:	==	!=	??	!?	!=?	**!<>**	!>=	!>	!<	!<=
Alternative silent:	-	?<>	!>=<	-	-	-	?>=	?>	?<	?<=

The additional column is the *simply equal* or *neither less than nor greater than* predicate. On the rule that no signalling predicate should begin with an exclamation mark, it uses the IEEE 754 suggested **=?** symbol for the signalling version of that predicate, and proposes the Math symbol itself for its silent negation. This also gives a obvious hint as to where Kahan came up with the symbol **!=?** for the silent negation of <>.

Solely for completeness of documentation, some alternative symbols are shown for the quiet predicates, their origin being IEEE 754 standards documents except for >=< and **!>=<** which come from Kahan. However, at least to the author of this document, the former group have too many question marks and Kahan's pair looks too bulky to be practical, making them, in the author's opinion, impractical for implementation in Chapel. Included in that list is the predicate **?<>** that the standard suggests could be an alternative to the **!=** inequality predicate, something the author considers of no benefit.

3. Optimization

Note that Chapel must never optimize away expressions like

```
x != x
x == x
```

and should rely on the underlying predicate to do its job as per the IEEE 754 standard.

4. References

[1] *IEEE Standard for Binary Floating-Point Arithmetic*, (1985), in ANSI/IEEE Std 754-1985, pp.1-20, 12 October 1985,

[2] *IEEE Standard for Floating-Point Arithmetic*, (2008), in IEEE Std 754-2008, pp.1-70, 29 August 2008,

[3] *IEEE Standard for Floating-Point Arithmetic*, (2019), in IEEE Std 754-2019 (Revision of IEEE 754-2008), pp.1-84, 22 July 2019,

[4] W Kahan (1997), *Lecture Notes on the Status of IEEE Standard 754 for Binary Floating-Point Arithmetic*, EECS, University of California at Berkeley,
See: *http://www.cs.berkeley.edu/~wkahan/ieee754status/ieee754.ps*

[5] W Kahan (2002), *Fclass: a Proposed Classification of Standard Floating-Point Operands*, EECS, University of California at Berkeley,
See: *http://www.cs.berkeley.edu/~wkahan/ieee754status/Fclass.pdf*