Here are some initial comments and questions:

Quote:

"**4.3.5 Static Function Calls**
The expanded QName and number of arguments in the static function call must match the name and arity of a function signature in the static context; if there is no match, a static error is raised [err:XPST0017]. For this purpose the number of arguments in the static function call is the number of positional arguments, plus one if there are one or more keyword arguments."

Question:
This is true for array-variadic and record-variadic functions.
For bounded variadic functions shouldn't the number of arguments in the function call be M + N, where M is the number of positional arguments and N is the number of (all declared) keyword arguments?

Quote:

"`my:keyword-function(x:=3, y:=4)` denotes a static function call with one argument: the effective value of the argument is `map{'x':3,'y':4}`."

Questions:
1. Isn't this statement true only for record-variadic functions?

2. `keyword-function()` may even be non-variadic, just a function with two positional/required parameters and in this call the two positional arguments are supplied by keyword. In this case, shouldn't the number of arguments be 2?

3. Shouldn't the number of arguments be 2 also for a bounded-variadic function? It is not record-variadic or an array-variadic function.

Quote:

"4.3.6 Static Functions

`%variadic("no")` indicates that the function is not variadic. In a function call, an argument must be supplied for every parameter in the function signature. It

can be supplied either positionally or by keyword, but there must be a one-to-one mapping between arguments and parameter declarations. The minimum and maximum arity are both defined by the number of parameter declarations."

Question:

Isn't it more precise and clear to say that in this case both the minimum and maximum arity of the function have the same value, equal to the number of parameter declarations?

Quote:

"`%variadic("bounded")` indicates that the function is bounded-variadic. A bounded-variadic function declares zero or more required parameters and one or more optional parameters. In a function call, an argument must be supplied for every required parameter, and arguments may be supplied for optional parameters: in both cases, the argument may be supplied either positionally or by keyword"

Question:

Why is there no definition of what is the arity (minimum and maximum) of the function in this case?"

Quote:

"`%variadic("sequence")` indicates that the function is sequence-variadic. A sequence-variadic function declares one or more parameters, of which the last typically has an occurrence indicator of `*` or `+` to indicate that a sequence may be supplied.

Questions:

1. Why sequence-variadic and not array-variadic?

If an array is used, then it can hold for example:

```
[1, (), 2, 3]
```

and all 4 elements of the array will be accessible from the function call, not just 3 as in the case when a sequence is passed:

```
(1, (), 2, 3)
```

Thus having array-variadic functions (function calls) is more precise and expressive, as shown in this example.

2. If the only reason is that `fn:concat()` cannot be expressed/called as an array-variadic function, can we have both: array-variadic and sequence-variadic (the latter just for fn:concat()) defined?

Quote:
"`%variadic("map")` indicates that the function is map-variadic. A map-variadic function declares one or more parameters, of which the last must be a type that accepts a map.

Questions:
1. Why not "record-variadic"?
Unlike the map type, the new record type allows for static typing (and if dynamic typing is really necessary, an "extended" record type may be used). Thus, using "record-variadic" will be an improvement over "map-variadic".

2. Is there any example of an existing standard function that cannot be expressed/called as a record-variadic function, but can be called as a map-variadic function?

General question:
In the case when a function call can be array-variadic or record-variadic, the caller may prefer to pass just one array (or a record) containing the variadic arguments. In this case wouldn't it be good to allow the ability to specify as a keyword argument this array (or record) and have a standard name for these (for example "varargs")? I believe this will improve the readability of the code.

Thanks in advance for any answers and explanations,

Dimitre

On Fri, Dec 11, 2020 at 9:52 AM Michael Kay <mike@saxonica.com> wrote:
> I've posted a new draft for XPath 4.0 at
>
> https://qt4cg.org/branch/master/xquery-40/xpath-40-diff.html

> This include first attempts to draft spec prose for variadic function calls, and for applying the
> lookup operator to atomic values as suggested by Liam (current-date()?year syntax).
>
> Michael Kay
> Saxonica