Subject: Re: _overloadMinus: what to do with invalid input?
Posted by penteado on Mon, 29 Dec 2014 22:27:00 GMT
View Forum Message <> Reply to Message

On Monday, December 29, 2014 5:41:49 PM UTC-2, Mike Galloy wrote:
> I would think an error and halting, like IDL would do if you tried to use
> an invalid operator with the native types. What happens if you try to add
> two pointers? (not in front of my computer right now)

To answer Mike's question:

```
IDL> a=ptr_new(1)
IDL> b=ptr_new(2)
IDL> c=a+b
% Operation illegal with pointer types.
% Execution halted at: $MAIN$
```

Regarding Paul's question, I would say that the answer depends on how one envisions the object's usage. If one decides it makes no sense to do the subtraction, like with pointers, it should throw an error. An error should also be raised, instead of returning a value, if such a return value could be confused with a valid result. For instance, taking IDL's list:

```
IDL> l=list(1,2)
IDL> l+3
% LIST::_OVERLOADPLUS: Arguments must both be lists.
% Execution halted at: $MAIN$
IDL> l-1
% Unable to convert variable to type object reference.
% Execution halted at: $MAIN$
```

It throws an error if I try to add or subtract an integer to a list. If it returned something, like !null, 0 or an empty list, such a return could be confused with something that is a valid result, obtained from different operations.

One could argue that adding/subtracting a scalar to a list should mean applying that operation to each element in the list, so that l-1 above should be equivalent to l.map(lambda(x:x-1)). But that is not what currently happens (as of IDL 8.4).