Ruby master - Feature #16990

Sets: operators compatibility with Array

06/26/2020 08:27 PM - marcandre (Marc-Andre Lafortune)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	

Description

We currently have set <operator> array work fine:

```
Set[1] + [2] # => Set[1, 2]
```

Nothing works in the reverse order:

```
[1] + Set[2] # => no implicit conversion of Set into Array
# should be:
[1] + Set[2] # => [1, 2]
```

set-like operators

Note that the situation is particularly frustrating for &, | and -.
If someone wants to do ary - set, one **has** to do ary - set.to_a which will, internally, do a to_set, so what is happening is set.to_a.to_set!! (assuming ary is over SMALL_ARRAY_LEN == 16 size, otherwise it's still doing in O(ary * set) instead of O(ary)).

The same holds with & and |; see order issue as to why this can *not* (officially) be done any other way.

Reminder:

```
ary & ary.reverse # => ary
Set[*ary] & Set[*ary.reverse]  # => Set[*ary.reverse], officially order is indeterminate
```

Related issues:

Related to Ruby master - Feature #16989: Sets: need ♥		**Assigned**

History

#1 - 06/26/2020 08:47 PM - marcandre (Marc-Andre Lafortune)

- Related to Feature #16989: Sets: need ♥ added

#2 - 09/02/2020 08:59 PM - Eregon (Benoit Daloze)

Isn't [1].to_set + Set[2] a good workaround here?

#3 - 09/03/2020 12:09 AM - marcandre (Marc-Andre Lafortune)

Eregon (Benoit Daloze) wrote in #note-2:

> Isn't [1].to_set + Set[2] a good workaround here?

Did you mean only for +, or for all operators? Take - for example... ary.to_set - set the to_set is wasteful, and I don't want to write it in the first place

An important question is: should the result be a Set, or an Array? In most cases, if there is interoperability, it won't matter that much. I think that array <op> set should return an array. In my understanding, array & set should be a great way to say I have this array, I want to keep only those elements that match the set. It should just work.

I'll repeat a usecase which was a bunch of constants in RuboCop that were arrays. Changing them to Sets would break a bunch of code that does my_list_of_stuff + OtherClass::STUFF, for example.

#4 - 09/03/2020 08:47 AM - knu (Akinori MUSHA)

We can probably define Set#to_ary if it's OK, and Array#+ will be able to deal with a set. Let us think about the downsides...

#5 - 09/03/2020 08:50 AM - knu (Akinori MUSHA)

As for the result type, I think Array operators should return arrays. Otherwise array += set would turn the variable array to a Set and that would be a surprise.

#6 - 09/03/2020 02:00 PM - Eregon (Benoit Daloze)

Because Array and Set are fundamentally different, I think ensuring both operands have the same type, explicitly or internally, is completely reasonable.
I don't expect Array "set" operations to magically know about the Set representation.

Having something that is both fast for include? and + means it needs to keep both a set-like representation and an array-like representation, which is a memory trade-off that FastArray in your PRs makes.

> my_list_of_stuff + OtherClass::STUFF

Which is what type?
set + array already does set + array.to_set and returns a Set (dedup'd elements)

array + set is not so well defined.
Does it do array + set.to_a with duplicated elements? And then that makes the return type inconsistent with set + array.
Doing array.to_set + set implicitly doesn't seem nice either as knu (Akinori MUSHA) said.

I think a few practical examples from RuboCop would help to figure what makes most sense.
Maybe having a specialized abstraction like FastArray is what makes most sense for RuboCop if non-set operations are used frequently.

#7 - 09/03/2020 11:29 PM - marcandre (Marc-Andre Lafortune)

knu (Akinori MUSHA) wrote in #note-4:

> We can probably define Set#to_ary if it's OK, and Array#+ will be able to deal with a set. Let us think about the downsides...

While this may be a good thing, and at least make them interoperable, it is still quite inefficient... For example, Array#- would call to_ary, which would create a temporary array from the hash, only to create a temporary hash/set...

> As for the result type, I think Array operators should return arrays. Otherwise array += set would turn the variable array to a Set and that would be a surprise.

Indeed. I'm glad we agree

#8 - 09/03/2020 11:35 PM - marcandre (Marc-Andre Lafortune)

Eregon (Benoit Daloze) wrote in #note-6:

> I don't expect Array "set" operations to magically know about the Set representation.

I would like to expect it

> array + set is not so well defined.
> Does it do array + set.to_a

Yes

> And then that makes the return type inconsistent with set + array.

Yes. Note that it is already inconsistent (Set vs raising an error).

#9 - 09/04/2020 12:25 AM - mame (Yusuke Endoh)

I expect that ary + set return a Set, not an Array, unless it raises an exception.

> Otherwise array += set would turn the variable array to a Set and that would be a surprise.

It is a surprise if ary + set returns a collection object that is ordered and that has multiple instances in its elements.
To me. ary += set looks like int_val += float. It is not a surprise to me that it changes the type of int_val.

#10 - 09/04/2020 01:13 AM - knu (Akinori MUSHA)

mame (Yusuke Endoh) wrote in #note-9:

I expect that ary + set return a Set, not an Array, unless it raises an exception.

> Otherwise array += set would turn the variable array to a Set and that would be a surprise.

It is a surprise if ary + set returns a collection object that is ordered and that has multiple instances in its elements.

I will use array | something if I mean to deduplicate the result, so array + something to me is the way to explicitly say I want to simply concatenate two lists. (should array + set be defined)

> To me. ary += set looks like int_val += float. It is not a surprise to me that it changes the type of int_val.

The coercion protocol in Numeric classes works like that not to lose precision. In that sense, I think Array is to Set as Float is to Integer because a set can be converted to an array without losing information and not the other way around, so you could argue that set + array and array + set should both return an array when int + float and float + int both result in a float.