

Ruby - Feature #12593

Allow compound assignments to work when destructuring arrays

07/18/2016 12:59 PM - najamelan (Naja Melan)

Status:Rejected Priority:Normal Assignee: Target version:	
Description <pre>a = ['a', 'b'] b = ['c', 'd'] def c return ['A', 'B'], ['C', 'D'] end a, b += c # -> would be awesome, but gives syntax error a, b = a + c.first, b + c.last # clunky and will call method twice... # current realistic use: t, tt = c a += t b += tt # desired result # p a == ['a', 'c', 'A', 'B'] #-> true p b == ['b', 'd', 'C', 'D'] #-> true I would propose that as a, b = [c, d] # is equivalent to: a = c b = d a, b += [c, d] # would be equivalent to: a += c b += d This not working surprised me. It could work with all compound assignment operators I think. Maybe even with some other operators.</pre>	

History

#1 - 07/19/2016 01:08 AM - shyouhei (Shyouhei Urabe)

The problem is, parallel assignment can go ultra complex. Its left hand and right hand side not necessarily are arrays, or not always come in same count. a, b = 1, [2], "3", :4' is a valid ruby code; there seems no reason to forbid a, b += 1, [2], "3", :4', or even, a, b += 1.

One could think of behaviours of such assignments, but I doubt if there are reasonable definitions for all pitfall-ish situations.

#2 - 07/19/2016 09:05 AM - najamelan (Naja Melan)

What I propose is to translate the construct:

```
a = 3
b = [ 2 ]

a, b = 2, 3, 4

a == 2
b == 3
```

```
# 4 is ignored
```

```
a, b = 2
```

```
a == 2  
b == nil
```

That's current behaviour if the number of elements is not the same on both sides.

```
a,b += 2, 3, 4
```

```
# translates to:
```

```
a = a + 2  
b = b + 3 # `+': no implicit conversion of Fixnum into Array (TypeError)
```

```
# 4 is ignored.
```

```
a, b += 2
```

```
a = a + 2  
b = b + nil
```

If you just translate the statement, There is no special case to take into account, the interpreter will just do it's job on the translated statement.

Obviously you have to make special accomodation for everything that you want to work:

```
if a, b < c, d # would become:
```

```
if a < c && b < d # then && is arbitrary, matter of convention, that someone has to decide on.
```

but whatever wouldn't work because nobody has wanted to implement it, well it already doesn't work today, so that's no change then.

#3 - 08/11/2016 07:23 AM - matz (Yukihiro Matsumoto)

- Status changed from Open to Rejected

I am not going to make multiple assignments even more complex.
The behavior of OP_ASGN for multiple assignments is not obvious.

Matz.