When pattern matching arrays, often times we want to assert that a given array’s elements all match a pattern.

For example:

```ruby
class Robot; end
class User; end

case [User.new, User.new]
in [User, '*'] => u if u.all? { _1 in User }
  puts 'array of users!' end
# => array of users!

case [User.new, User.new, Robot.new]
in [User, '*'] => u if u.all? { _1 in User }
  puts 'array of users!' end
# => Error: guard clause does not return true (NoMatchingPatternError)
```

I propose an improved syntax for matching an array's elements:

```ruby
case [User.new, User.new]
in [User*] => puts 'array of users!' end
```

Putting the `*` on the right-hand side differentiates the syntax from the array Splat operator. Another option would be `**`, or perhaps using a range `User...`

Not only is the pattern more readable, it also allows these constraints to be expressed in single line pattern matching:

```ruby
[User.new, User.new] => [User*] => arr
  #=> [#{<User:0x00000001074f5828>, <User:0x00000001074f56e8>}]  

[User.new, Robot.new] => [User*] => arr
  #=> [#{<User:0x00000001074f56e8>, <Robot:0x00000001074f5580>}] does not return true (NoMatchingPatternError)

[User.new, User.new, Robot.new] => [User*, Robot] => arr
  #=> [#{<User:0x00000001074f5580>, <User:0x00000001074f56e8>, <Robot:0x00000001074f5580>}]  

[User.new, User.new] in [User*]
  #=> true

[User.new, Robot.new] in [User*]
  #=> false
```

It allows single line pattern matching to express similar constraints to `case in`, since guard clauses are not available for single line pattern matching:

```ruby
[User.new, User.new] in [User, '*'] => u if u.all? { _1 in User }
  #=> undefined method `all?' for nil:NilClass (NoMethodError)
```