Ruby master - Feature #7939

Alternative curry function creation

02/24/2013 08:38 PM - drKreso (Kresimir Bojcic)

Status:	Feedback	
Priority:	Normal	
Assignee:	matz (Yukihiro Matsumoto)	
Target version:		

Description

=begin
I really like the new "(({assuming}))" method used for currying in Perl 6.

For example if I loose my mind and implement ((%fizzbuzz%)) via currying I can do it like this:

```
fb = ->(modulo_number, message, x) { message if x % modulo_number == 0 }
fizzbuzz = fb.curry[15,"FizzBuzz"]
fizz = fb.curry[3, "Fizz"]
buzz = fb.curry5, "Buzz".each { |i| puts fizzbuzz[i] || fizz[i] || buzz[i] || i }
```

Here the first hurdle is that curry is somewhat mathematical, and the secons is that you need to use (({[]})) for function invoking...
If we had something similar to this:

```
class Proc
def assuming(*args)
curry.call *args
end
end
```

It could be written more naturally IMO:

```
fb = ->(modulo_number, message, x) { message if x % modulo_number == 0 }
fizzbuzz = fb.assuming(15,"FizzBuzz")
buzz = fb.assuming(5, "Buzz")
fizz = fb.assuming(3,"Fizz")

(1..100).each { |i| puts fizzbuzz[i] || fizz[i] || buzz[i] || i }
```
=end

Related issues:

Related to Ruby master - Feature #6817: Partial application	**Open**
Related to Ruby master - Feature #13765: Add Proc#bind	**Open**

History

#1 - 02/24/2013 10:30 PM - nobu (Nobuyoshi Nakada)

It's Partial Application, not currying.

#2 - 02/24/2013 11:19 PM - nobu (Nobuyoshi Nakada)

- Description updated

- Status changed from Open to Rejected

=begin
```
fb = ->(modulo_number, message, x) { message if x % modulo_number == 0 }.curry(3)
fizzbuzz = fb[15,"FizzBuzz"]
fizz = fb[3, "Fizz"]
buzz = fb5, "Buzz".each { |i| puts fizzbuzz[i] || fizz[i] || buzz[i] || i }
```
=end

#3 - 02/24/2013 11:23 PM - naruse (Yui NARUSE)

- Category set to core

- Status changed from Rejected to Assigned

- Assignee set to matz (Yukihiro Matsumoto)

- Target version set to 2.6

> nobu
> It can't be reason to reject this.
> And describe the reason by English

#4 - 02/25/2013 06:41 AM - marcandre (Marc-Andre Lafortune)

From http://bugs.ruby-lang.org/projects/ruby/wiki/HowToRequestFeatures, I believe this FR fails point (1) "Ensure it's a meaningful improvement" and (2) "what's a good name".

It's pretty clear Nobu's example is more concise and nicer than what assuming would provide. BTW, the .curry(3) part can simply be .curry.

I second Nobu in rejecting this FR, but Matz can do it if you prefer.

#5 - 02/25/2013 09:17 AM - phluid61 (Matthew Kerwin)

marcandre (Marc-Andre Lafortune) wrote:

> It's pretty clear Nobu's example is more concise and nicer than what assuming would provide. BTW, the .curry(3) part can simply be .curry.

As a philosophical question, would a name like #curry_with or #apply be more appropriate? There is a small amount of utility added by such a method: at first glance I didn't see the .curry on the end of the first line of Nobu's example, and it took a bit of effort on my part to parse the code (or its intent); however the call to .assuming made it clear that something different was intended and happening at that point.

The square-bracket invocation falls somewhere in between, in terms of readability. It may just be me, but sometimes I temporarily fail to grasp that foo.bar[baz] is actually two chained method calls. In this case it was counfounded by the fact that I forgot that Proc#curry has an optional arity parameter. I know that those are my issues, that it is up to me to overcome, but having such hints in the code can help.

#6 - 02/25/2013 10:31 AM - marcandre (Marc-Andre Lafortune)

phluid61 (Matthew Kerwin) wrote:

> As a philosophical question, would a name like #curry_with or #apply be more appropriate? There is a small amount of utility added by such a method: at first glance I didn't see the .curry on the end of the first line of Nobu's example, and it took a bit of effort on my part to parse the code (or its intent); however the call to .assuming made it clear that something different was intended and happening at that point.
>
> The square-bracket invocation falls somewhere in between, in terms of readability. It may just be me, but sometimes I temporarily fail to grasp that foo.bar[baz] is actually two chained method calls. In this case it was counfounded by the fact that I forgot that Proc#curry has an optional arity parameter. I know that those are my issues, that it is up to me to overcome, but having such hints in the code can help.

There are many different ways to write the given example. You seem to be willing to use [] since your last line also has [] in it. But you can use .call or even .():

```
fb = ->(modulo_number, message, x) { message if x % modulo_number == 0 }
.curry  # <---- difficult to miss if it's at the beginning of a line
fizzbuzz = fb.(15,"FizzBuzz")
fizz = fb.(3, "Fizz")
buzz = fb.(5, "Buzz")
(1..100).each { |i| puts fizzbuzz.(i) || fizz.(i) || buzz.(i) || i }
```

You can use all of them if you want...

```
fb = ->(modulo_number, message, x) { message if x % modulo_number == 0 }
fizzbuzz = fb.curry.(15,"FizzBuzz")
fizz = fb.curry.call(3, "Fizz")
buzz = fb.curry5, "Buzz".each { |i| puts fizzbuzz.(i) || fizz.call(i) || buzz[i] || i }
```

I feel that each of these variations is easier to understand than your proposition. Moreover, I use curry very rarely, so I see no use in adding a new method.

As for the name, I feel the one proposed was quite bad, but I don't think there is a really good one. The only name that would really fit is "curry_call", which is exactly what it is doing. I see no point in exchanging "curry.call" with "curry_call"...

#7 - 02/21/2014 06:54 PM - Anonymous

Thanks to Yui Naruse for reopening this issue. As a matter of fact, I hate the current behavior of #curry method with passion. Current #curry is nearly

utterly useless. In line with the OP suggestion, I propose to redefine #curry as follows:

```
class Proc
  def curry *args
    hash = args[0]
    ordered_curry, named_curry = hash.each_with_object [{}, {}] do |(k, v), m|
      case k
      when Integer then m[0][k] = v
      when Symbol then m[1][k] = v
      else fail ArgumentError end
    end
    ordered_min_size = ordered_curry.keys.max
    o = -> *args {
      loop.with_index.with_object [] do |(_, i), ary|
        break ary if args.empty? and i > ordered_min_size
        ary << ( ordered_curry[ i ] || args.shift )
      end
    }
    n = -> **named_args { named_curry.update named_args }
    -> *ordered_args, **named_args {
      self.( *o.( *ordered_args ), **n.( **named_args ) )
    }
  end
end
```

And then, the usage would be:

```
meal = -> main_course, soup, appetizer: nil, dessert: nil, drink: "beer" do
  { main_course: main_course,
    soup: soup,
    appetizer: appetizer,
    dessert: dessert,
    drink: drink }
end

a_la_carte = meal.curry( 1 => "chicken broth",
                         dessert: "creme brule",
                         drink: "mineral water" )

a_la_carte.( "coq au vin" )
#=> {:main_course=>"coq au vin", :soup=>"chicken broth", :appetizer=>nil, :dessert=>"creme brule", :drink=>"mi
neral water"}

a_la_carte.( "coq au vin", dessert: "blueberry cake" )
#=> {:main_course=>"coq au vin", :soup=>"chicken broth", :appetizer=>nil, :dessert=>"blueberry cake", :drink=>
"mineral water"}
```

There are some unresolved issues lurking in the above prototype code (such as how to curry methods and what to do with blocks), but since I took the pain to write it, I'm gonna use it in my personal library. But seriously, this is the only form of #curry which I can understand and memorize. Current #curry with its n argument is absolutely awful. To the heaven with theory. Curry is curry, you divide both ordered and named arguments into the "core" and "curry" part. Phew. I came out and actually said what I think this time.

#8 - 02/24/2014 02:18 AM - matz (Yukihiro Matsumoto)

- Status changed from Assigned to Feedback

We are not going to change the behavior of 'curry' with respect to the mathematical term currying, which origin is the famous mathematician Haskel Curry.

But we don't reject your opinion that proposed API is intuitive. All we say is that we need new name.

Matz.

#9 - 02/24/2014 03:04 AM - shugo (Shugo Maeda)

Yukihiro Matsumoto wrote:

> We are not going to change the behavior of 'curry' with respect to the mathematical term currying, which origin is the famous mathematician Haskel Curry.
>
> But we don't reject your opinion that proposed API is intuitive. All we say is that we need new name.

Which version of API do you have in mind, Kresimir's or Boris'?
I prefer Kresimir's because it's simple and reasonable, but it might be too simple to add it as a new built-in method.

Speaking of the name, I prefer Proc#bind for partial application, but it might be confusing with UnboundMethod#bind.

#10 - 07/26/2017 03:35 PM - k0kubun (Takashi Kokubun)

- Related to Feature #6817: Partial application added

#11 - 07/26/2017 03:36 PM - k0kubun (Takashi Kokubun)

- Related to Feature #13765: Add Proc#bind added

#12 - 12/25/2017 06:15 PM - naruse (Yui NARUSE)

- Target version deleted (2.6)