Ruby - Feature #7314

Convert Proc to Lambda doesn't work in MRI

11/09/2012 01:30 AM - schneems (Richard Schneeman)

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

Description

I have code where I need to convert a proc to a lambda (i need to be able to return out of the block). I would expect that passing a proc into a lambda to return a lambda. When I run this code on MRI i do not get the result I would expect

```
my_proc = proc { |x| x }
my_lambda = lambda &my_proc
my_lambda.lambda?
```

The result is false but I would expect it to be true

There is currently a way to turn a proc into a lambda in MRI like this:

```
def convert_to_lambda &block
  obj = Object.new
  obj.define_singleton_method(:_, &block)
  return obj.method(:_).to_proc
end
```

But this feels like a hack, and is not supported across other implementations. I would expect that passing a proc into a lambda to return a lambda, I believe it is a bug.

Related issues:

Related to Ruby - Feature #12957: A more OO way to create lambda Procs	**Feedback**
Related to Ruby - Feature #15973: Let Kernel#lambda always return a lambda	**Closed**

History

#1 - 11/09/2012 02:03 AM - shyouhei (Shyouhei Urabe)

- Tracker changed from Bug to Feature

Moved this to feature tracker. I think you are feeling this like a hack because, you are in fact doing something hacky (return from a block).

Anyway I'm not against that kind of feature, though I'm not sure if that can be achieved by lambda(&proc). So I feel a needs of discussion.

#2 - 11/10/2012 07:58 AM - schneems (Richard Schneeman)

I would like a standard way to turn a Proc into a lambda even if it cannot be achieved through lambda(&proc). I don't know if it will affect the outcome, but jRuby correctly returns a lambda from lambda(&proc) in 1.9 mode.

#3 - 11/24/2012 10:29 AM - mame (Yusuke Endoh)

- Status changed from Open to Assigned

- Assignee set to matz (Yukihiro Matsumoto)

- Target version set to 2.6

Matz will require a use case, I guess.

--
Yusuke Endoh mame@tsg.ne.jp

#4 - 01/28/2016 09:51 PM - avit (Andrew Vit)

Use case: a stored block gets called with a "target" as first parameter, plus optional arguments. If the target object is an array, then the behaviour is unpredictable:

```
lm = lambda { |target, *options| puts target.inspect, options.inspect }
```

```
pr = proc   { |target, *options| puts target.inspect, options.inspect }

lm.call([1,2], 'a')  #=> [1, 2], ["a"]
pr.call([1,2], 'a')  #=> [1, 2], ["a"]

lm.call([1,2])  #=> [1, 2], []
pr.call([1,2])  #=> 1, [2]
```

Note how calling the block with only a single array destructures the "target" argument.
If we need to avoid destructuring, a lambda does the right thing, so converting it would be helpful.

#5 - 05/17/2016 07:53 AM - nobu (Nobuyoshi Nakada)

- Description updated

#6 - 05/18/2016 01:02 AM - shyouhei (Shyouhei Urabe)

We looked at this issue in yesterday's developer meeting.

The use case Andrew showed is (in spite of his intention) not unpredictable. Rather, if we allowed proc->lambda transformation, it gets unpredictable for a proc body what to expect for its first argument.

When writing a proc literal, its programmer's intention is clear that they wanted proc-style behaviour in mind. Currently if a block starts with "lambda {", it behaves like a lambda. All others are proc. This is consistent. But if we introduce proc conversion, this intension breaks. You can't predict what would happen when a proc is actually called during you write that proc; because it can be converted later. This is bad.

#7 - 05/18/2016 10:34 PM - headius (Charles Nutter)

Maybe this should warn or error when you attempt to turn a proc into a lambda?

The JRuby behavior is not intentional and we'll align with whatever MRI does. I'd also note that next always exits the block exactly like lambda's return, for all of procs, lambdas, and uncaptured blocks.

#8 - 05/21/2016 09:15 AM - shyouhei (Shyouhei Urabe)

Shyouhei Urabe wrote:

> Currently if a block starts with "lambda {", it behaves like a lambda. All others are proc.

I have to apology that I forgot about stabby lambdas here. They behave like lamndas too.

Anyways this doesn't change the fact that the way a Proc instance behaves, is statically determined when they are literally written. I think it is a good property. I'd like to +1 to Charles' proposal to add warning when people try converting a proc into lambda.

#9 - 11/20/2016 01:41 PM - shyouhei (Shyouhei Urabe)

- Related to Feature #12957: A more OO way to create lambda Procs added

#10 - 07/03/2019 01:59 AM - shyouhei (Shyouhei Urabe)

- Related to Feature #15973: Let Kernel#lambda always return a lambda added