Backport200 - Backport #8872

Case statements do not honor a refinement of the '===' method

09/07/2013 02:17 PM - jconley88 (Jon Conley)

Status:	Closed	
Priority:	Normal	
Assignee:	nagachika (Tomoyuki Chikanaga)	

Description

=begin
Below, I've redefined the ((|===|)) method of symbol to always return true. In ((|RefineTest#uses_refinement|)), I call ((|===|)) directly and the refined method is called. In ((|RefineTest#does_not_use_refinement|)), the ((|===|)) method is called indirectly through a case statement. If the refined ((|===|)) method was called, the result should be (('*The refinement was used*')), but this code currently returns (('*The refinement was not used*')).

```
module RefineSymbol
refine Symbol do
def ===(other)
true
end
end
end

using RefineSymbol

class RefineTest
def uses_refinement
:a === :b
end

def does_not_use_refinement
case :a
when :b
'The refinement was used'
else
'The refinement was not used'
end
end
end

rt = RefineTest.new
rt.uses_refinement   # => true
rt.does_not_use_refinement  # => expected 'The refinement was used' but got 'The refinement was not used'
```
=end

Related issues:

Related to Backport200 - Backport #9175: Backport r43913	**Closed**	**11/29/2013**
Has duplicate Ruby trunk - Bug #9150: Segfault in case statement execution, p...	**Closed**	**11/25/2013**

Associated revisions

Revision 66915c50 - 09/07/2013 06:44 AM - charliesome (Charlie Somerville)

- vm_eval.c (vm_call0): fix prototype, the id parameter should be of type ID, not VALUE
- vm_insnhelper.c (check_match): the rb_funcall family of functions does not care about refinements. We need to use rb_method_entry_with_refinements instead to call === with refinements. Thanks to Jon Conley for reporting this bug. [ruby-core:57051] [Bug #8872]
- test/ruby/test_refinement.rb: add test

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@42869 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 42869 - 09/07/2013 06:44 AM - charliesome (Charlie Somerville)

- vm_eval.c (vm_call0): fix prototype, the id parameter should be of type ID, not VALUE
- vm_insnhelper.c (check_match): the rb_funcall family of functions does not care about refinements. We need to use rb_method_entry_with_refinements instead to call === with refinements. Thanks to Jon Conley for reporting this bug. [ruby-core:57051] [Bug #8872]
- test/ruby/test_refinement.rb: add test

Revision 42869 - 09/07/2013 06:44 AM - charliesome (Charlie Somerville)

- vm_eval.c (vm_call0): fix prototype, the id parameter should be of type ID, not VALUE
- vm_insnhelper.c (check_match): the rb_funcall family of functions does not care about refinements. We need to use rb_method_entry_with_refinements instead to call === with refinements. Thanks to Jon Conley for reporting this bug. [ruby-core:57051] [Bug #8872]
- test/ruby/test_refinement.rb: add test

Revision 42869 - 09/07/2013 06:44 AM - charliesome (Charlie Somerville)

- vm_eval.c (vm_call0): fix prototype, the id parameter should be of type ID, not VALUE
- vm_insnhelper.c (check_match): the rb_funcall family of functions does not care about refinements. We need to use rb_method_entry_with_refinements instead to call === with refinements. Thanks to Jon Conley for reporting this bug. [ruby-core:57051] [Bug #8872]
- test/ruby/test_refinement.rb: add test

Revision 42869 - 09/07/2013 06:44 AM - charliesome (Charlie Somerville)

- vm_eval.c (vm_call0): fix prototype, the id parameter should be of type ID, not VALUE
- vm_insnhelper.c (check_match): the rb_funcall family of functions does not care about refinements. We need to use rb_method_entry_with_refinements instead to call === with refinements. Thanks to Jon Conley for reporting this bug. [ruby-core:57051] [Bug #8872]
- test/ruby/test_refinement.rb: add test

Revision 42869 - 09/07/2013 06:44 AM - charliesome (Charlie Somerville)

- vm_eval.c (vm_call0): fix prototype, the id parameter should be of type ID, not VALUE
- vm_insnhelper.c (check_match): the rb_funcall family of functions does not care about refinements. We need to use rb_method_entry_with_refinements instead to call === with refinements. Thanks to Jon Conley for reporting this bug. [ruby-core:57051] [Bug #8872]
- test/ruby/test_refinement.rb: add test

Revision 42869 - 09/07/2013 06:44 AM - charliesome (Charlie Somerville)

- vm_eval.c (vm_call0): fix prototype, the id parameter should be of type ID, not VALUE
- vm_insnhelper.c (check_match): the rb_funcall family of functions does not care about refinements. We need to use rb_method_entry_with_refinements instead to call === with refinements. Thanks to Jon Conley for reporting this bug. [ruby-core:57051] [Bug #8872]
- test/ruby/test_refinement.rb: add test

Revision 3041c43d - 09/12/2013 03:54 PM - nagachika (Tomoyuki Chikanaga)

merge revision(s) 42869: [Backport #8872]

```
    * vm_eval.c (vm_call0): fix prototype, the id parameter should be of
      type ID, not VALUE

    * vm_insnhelper.c (check_match): the rb_funcall family of functions
      does not care about refinements. We need to use
      rb_method_entry_with_refinements instead to call === with
      refinements. Thanks to Jon Conley for reporting this bug.
      [ruby-core:57051] [Bug #8872]

    * test/ruby/test_refinement.rb: add test
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_0_0@42923 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 42923 - 09/12/2013 03:54 PM - nagachika (Tomoyuki Chikanaga)

merge revision(s) 42869: [Backport #8872]

```
* vm_eval.c (vm_call0): fix prototype, the id parameter should be of
  type ID, not VALUE

* vm_insnhelper.c (check_match): the rb_funcall family of functions
  does not care about refinements. We need to use
  rb_method_entry_with_refinements instead to call === with
  refinements. Thanks to Jon Conley for reporting this bug.
  [ruby-core:57051] [Bug #8872]

* test/ruby/test_refinement.rb: add test
```

Revision 6136b992 - 01/23/2014 12:40 PM - nagachika (Tomoyuki Chikanaga)

merge revision(s) 43913: [Backport #8872] [Backport #9175]

```
    * vm_insnhelper.c (check_match): Fix SEGV with VM_CHECKMATCH_TYPE_CASE
      and class of `pattern` has `method_missing`
      [Bug #8882] [ruby-core:58606]
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_0_0@44695 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 44695 - 01/23/2014 12:40 PM - nagachika (Tomoyuki Chikanaga)

merge revision(s) 43913: [Backport #8872] [Backport #9175]

```
* vm_insnhelper.c (check_match): Fix SEGV with VM_CHECKMATCH_TYPE_CASE
  and class of `pattern` has `method_missing`
  [Bug #8882] [ruby-core:58606]
```

History

#1 - 09/07/2013 03:44 PM - charliesome (Charlie Somerville)

- Status changed from Open to Closed

- % Done changed from 0 to 100

This issue was solved with changeset r42869.
Jon, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

- vm_eval.c (vm_call0): fix prototype, the id parameter should be of type ID, not VALUE
- vm_insnhelper.c (check_match): the rb_funcall family of functions does not care about refinements. We need to use rb_method_entry_with_refinements instead to call === with refinements. Thanks to Jon Conley for reporting this bug. [ruby-core:57051] [Bug #8872]
- test/ruby/test_refinement.rb: add test

#2 - 09/07/2013 03:48 PM - charliesome (Charlie Somerville)

- Backport changed from 1.9.3: UNKNOWN, 2.0.0: UNKNOWN to 1.9.3: DONTNEED, 2.0.0: REQUIRED

#3 - 09/11/2013 11:55 PM - nagachika (Tomoyuki Chikanaga)

- Tracker changed from Bug to Backport

- Project changed from Ruby trunk to Backport200

- Status changed from Closed to Assigned

- Assignee set to nagachika (Tomoyuki Chikanaga)

#4 - 09/13/2013 12:54 AM - nagachika (Tomoyuki Chikanaga)

- Status changed from Assigned to Closed

This issue was solved with changeset r42923.
Jon, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

---

merge revision(s) 42869: [Backport #8872]

```
* vm_eval.c (vm_call0): fix prototype, the id parameter should be of
  type ID, not VALUE

* vm_insnhelper.c (check_match): the rb_funcall family of functions
  does not care about refinements. We need to use
  rb_method_entry_with_refinements instead to call === with
  refinements. Thanks to Jon Conley for reporting this bug.
  [ruby-core:57051] [Bug #8872]

* test/ruby/test_refinement.rb: add test
```

#5 - 11/27/2013 01:48 AM - pixeltrix (Andrew White)

This has caused a regression in Rails where === isn't being sent via method_missing:

https://github.com/rails/rails/pull/13055

This results in a segmentation fault.

#6 - 11/29/2013 05:58 PM - sorah (Sorah Fukumori)

pixeltrix (Andrew White) Fixed at r43913, Thank you for your reporting

#7 - 11/29/2013 06:12 PM - shugo (Shugo Maeda)

- Status changed from Closed to Assigned

- Assignee changed from nagachika (Tomoyuki Chikanaga) to matz (Yukihiro Matsumoto)

As I stated in #9150, I doubt this feature conforms to the design policy of refinements.
r42869 should be reverted, shouldn't it, Matz?

#8 - 11/29/2013 06:24 PM - shugo (Shugo Maeda)

I strongly request committers not to change the behavior of Refinements without Matz's permission, except when there's an obvious bug such as SEGV in Refinements, because the current feature set of Refinements are restricted to keep compatibility with other implementations such as JRuby.

In general, refinements should be activated only in a certain lexical scope.

#9 - 11/29/2013 06:38 PM - sorah (Sorah Fukumori)

shugo (Shugo Maeda) wrote:

> I strongly request committers not to change the behavior of Refinements without Matz's permission, except when there's an obvious bug such as SEGV in Refinements, because the current feature set of Refinements are restricted to keep compatibility with other implementations such as JRuby.
>
> In general, refinements should be activated only in a certain lexical scope.

Agreed.

Note that my commit (r43913) just fixes bug in the current behavior, please revert it too with r42839, if we revert r42839.

#10 - 11/29/2013 07:10 PM - matz (Yukihiro Matsumoto)

The basic principle of refinements is that refinement do work in local scope.
So methods called from the scope will be refined but method called from methods called will not.

And very importantly, "===" called from case statement is controversial.
It can be viewed as a method called from case statement, or case statement itself is just a syntax sugar wrapping "===" calls. From the former point of view, refine should not affect "===", but from the latter POV, refinement should change the behavior.

Right now, we took the former POV, and OP expected the latter.
So I would like to hear from others.

Matz.

#11 - 11/29/2013 09:07 PM - shugo (Shugo Maeda)

matz (Yukihiro Matsumoto) wrote:

> And very importantly, "===" called from case statement is controversial.
> It can be viewed as a method called from case statement, or case statement itself is just a syntax sugar wrapping "===" calls. From the former point of view, refine should not affect "===", but from the latter POV, refinement should change the behavior.

Ruby has other features which call methods implicitly. For example, Range#to_a is called by the following code:

a = *[1..10]

Should refinements be honored in all such features?

#12 - 11/29/2013 10:28 PM - shugo (Shugo Maeda)

I've found that for expressions honor refinements.

module RefineEach

```
refine Array do
def each
yield "refined"
end
end
end

using RefineEach

for i in [1, 2, 3]
p i
end
```

for expressions seem to be more closely related to method calls, but I'm not sure how refinements should be handled in these cases....

#13 - 11/29/2013 10:32 PM - Eregon (Benoit Daloze)

shugo (Shugo Maeda) wrote:

> a = *[1..10]

You probably meant

```
a = [*1..10]
```

#14 - 11/29/2013 11:09 PM - shugo (Shugo Maeda)

Eregon (Benoit Daloze) wrote:

> > a = *[1..10]
>
> You probably meant
>
> ```
> a = [*1..10]
> ```

Ah, I meant:

a = *1..10

Thanks anyway.

#15 - 11/30/2013 10:48 AM - shugo (Shugo Maeda)

Could anyone tell me use cases of this feature?

I think refinements are for object-oriented ways using dynamic dispatch, but case expressions are not for such object-oriented ways.

#16 - 11/30/2013 07:07 PM - jballanc (Joshua Ballanco)

To play devil's advocate, I think the opposing question to be asked is "how would you alter the behavior of case evaluation using refinements"?

It may be, as shugo states, that refinements are not meant to be used for altering the behavior of expressions. In this case, the answer to the question I posed is, simply, "you don't."

That said, I think it is fairly well understood in the Ruby community that "===" is related to case. I've even heard the operator described as "case equality". Playing devil's advocate again, what is the use case for refining "===" if it doesn't affect the evaluation of a case...when clause?

#17 - 12/04/2013 11:54 AM - shugo (Shugo Maeda)

jballanc (Joshua Ballanco) wrote:

> To play devil's advocate, I think the opposing question to be asked is "how would you alter the behavior of case evaluation using refinements"?
>
> It may be, as shugo states, that refinements are not meant to be used for altering the behavior of expressions. In this case, the answer to the question I posed is, simply, "you don't."
>
> That said, I think it is fairly well understood in the Ruby community that "===" is related to case. I've even heard the operator described as "case equality". Playing devil's advocate again, what is the use case for refining "===" if it doesn't affect the evaluation of a case...when clause?

I don't come up with any use case for refining "===" regardless of whether it affects the evaluation of a case expression or not.
I'm neutral about this issue, and really want to know use cases.

#18 - 12/18/2013 10:40 AM - nagachika (Tomoyuki Chikanaga)

hi,

Is there any progress? Does anyone has usecase of refinements & case statement?

#19 - 12/18/2013 03:11 PM - matz (Yukihiro Matsumoto)

After consideration, I think for and case statement should honor refinement since they are fundamentally syntax sugar using each/=== methods.

Matz.

#20 - 12/22/2013 12:04 AM - nagachika (Tomoyuki Chikanaga)

How about 2.0.0? Can I change the behavior at 2.0.0 to same with 2.1?
If so I'll backport r43913 to fix SEGV.

#21 - 01/15/2014 04:29 PM - matz (Yukihiro Matsumoto)

- ruby -v set to 2.0.0

I don't think changing the behavior in the middle of a release is a good idea.
So I don't encourage backporting. Thanks.

Matz.

#22 - 01/15/2014 04:37 PM - matz (Yukihiro Matsumoto)

- ruby -v changed from 2.0.0 to -

Issue #8872 has been updated by Yukihiro Matsumoto.

ruby -v set to 2.0.0

I don't think changing the behavior in the middle of a release is a good idea.
So I don't encourage backporting. Thanks.

Matz.

Backport #8872: Case statements do not honor a refinement of the '===' method
https://bugs.ruby-lang.org/issues/8872#change-44355

- Author: Jon Conley
- Status: Assigned
- Priority: Normal
- Assignee: Yukihiro Matsumoto
- Category:
- Target version:
- ruby -v: 2.0.0 ---------- =begin Below, I've redefined the ((|===|)) method of symbol to always return true. In ((|RefineTest#uses_refinement|)), I call ((|===|)) directly and the refined method is called. In ((|RefineTest#does_not_use_refinement|)), the ((|===|)) method is called indirectly through a case statement. If the refined ((|===|)) method was called, the result should be (('*The refinement was used*')), but this code currently returns (('*The refinement was not used*')).

module RefineSymbol
refine Symbol do
def ===(other)
true
end
end
end

using RefineSymbol

class RefineTest
def uses_refinement
:a === :b
end

```
def does_not_use_refinement
  case :a
  when :b
    'The refinement was used'
  else
```

```
    'The refinement was not used'
  end
end
```

end

rt = RefineTest.new
rt.uses_refinement   # => true
rt.does_not_use_refinement # => expected 'The refinement was used' but got 'The refinement was not used'
=end

--
http://bugs.ruby-lang.org/

#23 - 01/23/2014 12:04 PM - nagachika (Tomoyuki Chikanaga)

- Assignee changed from matz (Yukihiro Matsumoto) to nagachika (Tomoyuki Chikanaga)

Hi,

I've told matz that this change introduced by r42923 was already released at 2.0.0p353 and matz decided to not to revert the changeset. (see https://twitter.com/yukihiro_matz/status/423500823788662784)

I will backport r43913 to fix SEGV introduced at r42923.

Thanks.

#24 - 01/23/2014 12:41 PM - nagachika (Tomoyuki Chikanaga)

- Status changed from Assigned to Closed

Applied in changeset r44695.

---

merge revision(s) 43913: [Backport #8872] [Backport #9175]

```
* vm_insnhelper.c (check_match): Fix SEGV with VM_CHECKMATCH_TYPE_CASE
  and class of `pattern` has `method_missing`
  [Bug #8882] [ruby-core:58606]
```