Ruby master - Bug #10293

splatting an empty hash in a method invocation sends an argument to the method (should send nothing)

09/24/2014 11:57 PM - rits (First Last)

Status: Closed
Priority: Normal
Assignee: nobu (Nobuyoshi Nakada)
Target version: ruby -v: ruby 2.1.3p242 (2014-09-19 revision 47630) [x86_64-linux]
Backport: 2.0.0: UNKNOWN, 2.1: REQUIRED

Description

```irb(main):001:0> def foo; end; foo **{}``
ArgumentError: wrong number of arguments (1 for 0)

note that splatting an empty array results in no arguments

this behavior is particularly problematic with send (dispatching to different methods)

```send method_name, *possibly_empty_arg_list, **possibly_empty_keyarg_hash```

one should not have to do

```if possibly_empty_keyarg_hash.empty?
receiver.send method_name, *possibly_empty_arg_list
else
receiver.send method_name, *possibly_empty_arg_list, **possibly_empty_keyarg_hash
end```

or

```possibly_empty_arg_list << possibly_empty_keyarg_hash unless possibly_empty_keyarg_hash.empty?
receiver.send method_name, *possibly_empty_arg_list```

Related issues:

Related to Ruby master - Feature #14183: "Real" keyword argument

Closed

History

#1 - 09/25/2014 12:18 AM - nagachika (Tomoyuki Chikanaga)
- Category set to core
- Status changed from Open to Assigned
- Assignee set to nobu (Nobuyoshi Nakada)
- Target version set to 2.2.0
- Backport changed from 2.0.0: UNKNOWN, 2.1: UNKNOWN to 2.0.0: UNKNOWN, 2.1: REQUIRED

#2 - 09/25/2014 01:44 AM - rits (First Last)

https://bugs.ruby-lang.org/issues/9291 is somewhat related
foo(**nil) should not raise since foo(*nil) does not and it's the more useful behavior

#3 - 01/05/2018 09:01 PM - naruse (Yui NARUSE)
- Target version deleted (2.2.0)

#4 - 08/20/2019 08:21 PM - jeremyevans0 (Jeremy Evans)
- Related to Feature #14183: "Real" keyword argument added

#5 - 09/02/2019 04:33 AM - jeremyevans0 (Jeremy Evans)
- Status changed from Assigned to Closed

With the acceptance of #14183, double splatting an empty hash no longer sends an empty positional hash to the method.