Double splat does not work on empty hash assigned via variable

When an empty hash is given as a literal, the double splat operates on it, and leaves nothing, which is expected.

```ruby
class String
def foo; end
end

[**(v)] => []
/foo*.foo(**v) => nil
/foo*.send("foo", **v) => nil
```

However, when an empty hash is given via variable, the double splat retains an empty hash in place.

```ruby
h = {}

[**h] => [{}]
/foo*.foo(**h) => wrong number of arguments (given 1, expected 0)
/foo*.send("foo", **h) => wrong number of arguments (given 1, expected 0)
```

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

History

#1 - 12/22/2015 10:59 AM - sawa (Tsuyoshi Sawada)
I found more cases (other than via variable) that the double splat does not work on an empty hash. While parenthesis or nesting does not cause the issue:

```ruby
[**(v)] => []
[**begin {} end] => []
[**[**v]] => []
```

more complicated expressions seem to cause the issue:

```ruby
[**(v if true)} => [{]}
[**(if true; {} end)} => [{]}
[**(false ? :foo : {})} => [{]}
```

#2 - 02/14/2016 04:48 PM - justcolin (Colin Fulton)
A similar bug happens when using the double splat operator in the parameter list of a method definition. See #12022.

#3 - 03/08/2016 04:32 AM - sawa (Tsuyoshi Sawada)
This bug is related to #10708.

#4 - 05/17/2016 08:24 AM - nobu (Nobuyoshi Nakada)
- Description updated

#5 - 03/15/2019 09:23 AM - sawa (Tsuyoshi Sawada)
- Description updated

#6 - 06/27/2019 10:12 PM - jeremyevans0 (Jeremy Evans)
With recent commits to the master branch, keyword splats of empty hashes do not pass positional arguments to methods or cause an empty hash to be added to an array.