inconsistent behavior using `**` vs hash as method parameter

Using ruby 2.2.2 I see this work:

```
[18] pry(main)> def test(cmd, opts={})
[18] pry(main)*   puts cmd,opts
[18] pry(main)* end

=> :test

[19] pry(main)> test(:ads, :d => 6, :f => :gah, {a:3,b:4} => 3)
ads
{d=>6, f=>gah, {a=>3, b=>4}=>3}
=> nil
```

But this fails:

```
[2] pry(main)> def test(cmd, **opts)
[2] pry(main)*   puts cmd, opts
[2] pry(main)* end

[9] pry(main)> test(:ads, :d => 6, :f => :gah, {a:3,b:4} => 3)
ArgumentError: wrong number of arguments (2 for 1)
from ( pry):2:in 'test'
```

This is expected behavior, keyword argument hashes only support symbol keys, not hash keys. I suppose the error message could be better, though.

What message would be better?

Unsuppored key type?

btw what is the reason to have that limitation? Not that I need this functionality, I just happened to notice the difference...

When last argument is hash-like but has keys that are not symbols, we could output instead "wrong number of arguments (2 for 1); note that the last argument has keys that are not symbols and thus was not considered as keyword parameters"
With the changes in #14183, keyword splats can support arbitrary keys, and the second example now passes:

```ruby
test(:ads, :d => 6, :f => :gah, {a:3,b:4} => 3)

# ads
# {:d=>6, :f=>:gah, {:a=>3, :b=>4}=>3}
```