unknown keyword when passing an hash to a method that accepts a default argument and a named argument

Description

e.g.

```ruby
>> def a(b = {}, c: 1); p [b, c]; end; a(a: 1)
Traceback (most recent call last):
3: from (irb):3
2: from (irb):3:in `rescue in irb_binding'
1: from (irb):3:in `a'
ArgumentError (unknown keyword: a)
```

Prettier version:

```ruby
def do_something(thing = {}, force: true)
  # doing things
end
do_something({ user: { id: 1 } })
# => ArgumentError (unknown keyword: user)
```

Tested on ruby-head from ~1 month ago and ruby 2.6.2

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

History

#1 - 04/08/2019 02:35 AM - jeremyevans0 (Jeremy Evans)

I believe this is the same issue as #12717. We are trying to address this issue in #14183.

#2 - 09/02/2019 04:35 AM - jeremyevans0 (Jeremy Evans)

- Related to Feature #14183: "Real" keyword argument added

#3 - 09/02/2019 04:39 AM - jeremyevans0 (Jeremy Evans)

- Status changed from Open to Closed

The first example should fail, because you are passing an unsupported keyword to a method that accepts a keyword.

With the acceptance of #14183, the second example now submits a warning:

```ruby
do_something({ user: { id: 1 } })
# (irb):85: warning: The last argument is used as the keyword parameter
# (irb):82: warning: for `do_something' defined here
# ArgumentError (unknown keyword: :user)
```

In Ruby 3, the second example will pass the hash as the first argument. To get the Ruby 3 behavior with the master branch:

```ruby
do_something({ user: { id: 1 } }, **{});
```

# => nil