If Ruby classes could provide some means for redefining operator precedence, it would provide the flexibility useful to some DSL use-cases.

My particular application, for instance, is in an SI units system gem that could use \(^\) to mean power of the unit (e.g. 1.meter\(^3\) would mean cubic meters). But to do that right the operator needs a higher precedence. I don't expect it to be something commonly used, obviously, but it certain use cases like mine it is practically essential.

I first suggested that \(((#)^{\text{ii}})\) be given a higher precedence and XOR get another operator in \#6678. I was not surprised that it was rejected, but I figured it was the proper first step, before proposing this much broader feature request.

As for notation, I suppose the simplest means if to create class method that can move the precedence to a position relative to another, e.g.

```ruby
class Unit
  precedence :^, :**
end
```

Which is to say, move \(((#)^{\text{ii}})\) operator to a precedence above \(((#)^{\text{ii}})\).

Pretty interesting idea, that reminds me Prolog or Haskell. But changing precedence may confuse readers. It is as dangerous as macros.

So I reject the idea (for Ruby language) for same reason.

Matz.

Interesting. Can you tell us how "dangerous" it is, apart from being confusing?

"dangerous" here of course means "confusing" or "error prone". I don't think changing operator precedence would cause your computer explode.

Matz.
I know, but you said "as dangerous as macros". So my question.

Matz, I know you don't like macros.

But look out, there are tons of DSLs written in Ruby. Which means everybody, except you, are willing to have macros.

So I agree with you, macros are dangerous. But to meet our needs, Ruby should have "something macro-like, but more elegant and safe". Right?

So going back to this story. It has actual user. It might be good to have. Why you think this is "as dangerous as" macro?

Matz.

So it's about taste, not about dangers? If so it makes much more sense to me.

I have similar opinion with Matz: i think it is nice to have in language something about the syntax that never changes. In this case, it is to be able to look at a line of code and to know in which order it will be executed, without having to go through the entire program.

The operator precedence in my opinion is such a basic thing as parentheses parsing. I think it has to be defined in a reasonable way, but i do not think it is necessary to be able to redefine it, as you can always use parentheses if needed. I think this is similar to natural languages.

"you can always use parentheses if needed"

Unfortunately it is very unnatural for a unit system. Let me give an example to explain. In Stick, the obvious DSL is:

10.meters**2 => 100m
10.meters^2 => 10m²

So, ** is power on the value and ^ is used to indicate power of the unit.

But the precedence of ^ is a problem.

10.meters^2 / 2 => 10m

Rather then the 5m^2 expected. While parenthesis can be used, it so unnatural to this common notation that it puts a rather ugly black mark on creating a nice SI units system for Ruby at all.

This particular problem could easily be solved by raising the precedence of ^ to that of **, which is why I suggested #6678 first --although that feels more like a band-aid. Controllable precedence feels like the right solution to me b/c others may ultimately have different needs.

I am not sure what to do if both are rejected.

matz (Yukihiro Matsumoto) I understand your cautionary take. It certainly could be dangerous if someone where to horse around with precedence, say in the Object class. But I doubt it would prove such a problem in practice b/c developers would be well aware of the potential "dangers" of such a thing. It would only end up being used for very specific cases like my own. From a more general principle, that's always been my favorite thing about Ruby --it trusts developers to shoot themselves in the foot at their own discretion :)

If it would help, I could see a warning be issued stating "precedence has been changed for class XYZ". Also, if refinements are here to stay, maybe it could be localized by them?

I realize this isn't a minor feature request, and it needs to be considered carefully, both in its effects to the language and it's implementation. But it could prove fairly powerful, especially in academic areas. Maybe we could give the issue some time to collect feedback at least? Perhaps there are some other strong use cases out there we are not aware.

trans (Thomas Sawyer) wrote:
(10.meters)**2 should be 100.meters² in your notation.

Rather then the 5m² expected. While parenthesis can be used, it so unnatural to this common notation that it puts a rather ugly black mark on creating a nice SI units system for Ruby at all.

I think it is a choice between simple rules and complex context-dependent rules. Is the user expected to know the particular context he/she is working with, or should it be enough to know the standard Ruby parsing rules?

This particular problem could easily be solved by raising the precedence of ^ to that of **, which is why I suggested #6678 first --although that feels more like a band-aid. Controllable precedence feels like the right solution to me b/c others may ultimately have different needs.

If it was up to me, and if there were no compatibility problems, i would vote for using ^ everywhere in the sense of exponentiation, with the highest precedence, perhaps except for the unary minus (see my comment 19 in #7326). Related: #7322.

Update: Please correct me if i am wrong, but it seems to me that this proposal is about allowing Ruby to dynamically change its parsing rules.

#12 - 11/15/2012 09:46 AM - matz (Yukihiro Matsumoto)

trans (Thomas Sawyer) I don't deny the idea itself. But not for Ruby, because

- syntax should be stable.
- Ruby is not a language for academic research of programming languages.
- we cannot use yacc (or bison) any longer, if we adopt variable operator precedence.

each one of above is enough to reject.

Matz.

#13 - 11/15/2012 01:57 PM - trans (Thomas Sawyer)

"Ruby is not a language for academic research of programming languages."

I was referring to academic application, as in the sciences, using DSLs. Not experimenting with programming languages.

I understand though, the yacc point is a killer.

#14 - 11/21/2012 11:58 PM - nobu (Nobuyoshi Nakada)

Why don't you define meter² method? ;)

#15 - 11/22/2012 01:29 AM - judofyr (Magnus Holm)

On Tue, Nov 13, 2012 at 3:25 PM, trans (Thomas Sawyer) transfire@gmail.com wrote:

Issue #7336 has been updated by trans (Thomas Sawyer).

"you can always use parentheses if needed"

Unfortunately it is very unnatural for a unit system. Let me give an example to explain. In Stick, the obvious DSL is:

10.meters**2 => 100m
10.meters² => 10m²

So, ** is power on the value and ^ is used to indicate power of the unit.

It doesn't make any sense (mathematically) to only do power on the quantity. The unit is a part of the value.

(10*10).meters => 100m
10.meters * 2 => 100m²

Or, if you actually have 10 meters and want to square the quantity, make it explicit that it's a weird thing to do:

a = 10.meters
(a.quantity ** 10).to_unit(a.unit)

#16 - 11/22/2012 01:36 AM - trans (Thomas Sawyer)
As Alexey pointed out that was a mistake on my part. Yes, it should be:

10\text{meters}^2 \Rightarrow 100\text{m}^2

LOL! I would if I could! Actually, I've thought more than once that progress in keyboard input totally stagnated in the late 80s. Why can't we type superscripts and subscripts and make good use of it!

And use MS Word as IDE, please ;).