Ruby trunk - Feature #8437

custom operators, unicode

05/23/2013 09:20 AM - eike.rb (Eike Dierks)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	

Description

The ruby language does currently one support a predefined set of operators.
It would be nice to add custom operators.
A lot of people ask about the elvis operator aka (?:) to cope with nil, aka rails try()

This probably is a problem with the parser at first,
because introducing new operators makes parsing a lot more complicated.

Maybe we could allow symbols from the unicode space
to be used as new operators?
That would be compatible with all before code written in ascii.

So we could allow all the symbols from the unicode Math operators plane
to be available as operators in Ruby.

While few of you might have tried that,
Unicode is fully available for naming identifiers.

We should also extend the set of operators to the unicode space.

While we are still not used to it now,
to have some unicode characters in our codes,
could really add to the expressivness.

So as of today, you can already name your variable like
delta_t or Δt

Or if you're working with angles,
you might call your variables α or Θ
This is completely legal in Ruby.

I'm asking for:
this_set ∪ other_set

or maybe we could even do this for prefix like:
√2

I'd believe the math operator plane of unicode should be removed
from the allowable names of identifiers,
but should instead be reserved for operators in the parser, like + nowdays is.

History

#1 - 05/23/2013 10:10 AM - phluid61 (Matthew Kerwin)

eike.rb (Eike Dierks) wrote:

> I'd believe the math operator plane of unicode should be removed
> from the allowable names of identifiers,
> but should instead be reserved for operators in the parser, like +
> nowdays is.

Neither for nor against this idea at the moment, but which "math operator plane" do you mean? A quick read on wikipedia (PDF) lists the following potential Unicode 6 blocks:

- Mathematical Operators (U+2200–U+22FF) ⇒ PDF

- Miscellaneous Mathematical Symbols-A (U+27C0–U+27EF) ⇒ PDF
- Miscellaneous Mathematical Symbols-B (U+2980–U+29FF) ⇒ PDF
- Supplemental Mathematical Operators (U+2A00–U+2AFF) ⇒ PDF
- Letterlike Symbols (U+2100–U+214F) ⇒ PDF*I'd leave this block for identifiers, personally*
- Miscellaneous Technical (U+2308–U+230B) ⇒ PDF
- Geometric Shapes (U+25A0–U+25FF) ⇒ PDF
- Miscellaneous Symbols and Arrows (U+2B30–U+2B4C) ⇒ PDF
- Mathematical Alphanumeric Symbols (1D400–1D7FF) ⇒ PDF*Again, probably identifiers*

#2 - 05/25/2013 02:56 PM - eike.rb (Eike Dierks)

Hi Mathew,
thanks for the list.

This is a good definition.

All of the symbols from that lists,
as defined by Mathew above

Shall not be allowed as part of an identifier in the ruby language
but shall be reserved for other use.

Would you agree with this?
We need to name the planes to make this a bit more formal.

Is it ok if we define this by exclusion?
would every other symbol be ok?

there must be some definiton in unicode like 'letters'

But anyway,
deltaT

This is so beauttiful to write it with a greek delta.

I"m really looking forward to have some code
with variable names written in japanse letters.
(but please add a comment so that we can understand it)

#3 - 05/30/2013 11:42 PM - Anonymous

-1

Who the heaven told you that people write code in ASCII? The Unicode idea is quite corny. See https://github.com/collectiveidea/unicode_math, also as unicode_math gem. I find the present operator assortment varied enough to build the internal DSLs I want, such as:

SO_3 + H_2O >> H_2SO_4 | { ΔH: -146.kJ.mol⁻¹, E_a: 0.kJ.mol⁻¹ }

(Not functional yet, but you can already try units with Unicode exponents, gem install sy.)

Precedence table is the main feature and strength of operators. Connected to this is the less important feature of operators, the one that appeals to you, which is the possibility to drop the dot in their call:

a.+(b) #--> a + b

In my opinion, it is good to have the operators that we already have, but I would not like to have more. As for unary operators, they can always be replaced by unary method definitions:

def Δ(t); ... end

Precedence table already takes months to learn (for me). My memory is not below average, so I guess there must be some psychological snag, why even today, I have to check the Ruby book for precedence table all the time.

EDIT: .... Can't take my mind of this ... that unsightly dot, as in

require 'unicode_math'
5.× 5 #=> 25

... so, not really more operators, but ... the ability to change the

appearance of existing operators ... such as use × or ⋅ instead of * if
the user asks for it ... or ⊕ for +, ≡ for ===, ⊂ for <, but otherwise
they would be same operators, with the same precedence table, same operator
methods... And only for those cases where the user explicitly turns it on
on a per-character basis...

#4 - 08/27/2013 10:21 AM - phluid61 (Matthew Kerwin)

boris_stitnicky (Boris Stitnicky) wrote:

> EDIT: .... Can't take my mind of this ... that unsightly dot, as in
>
> require 'unicode_math'
> 5.× 5 #=> 25
>
> ... so, not really more operators, but ... the ability to change the
> appearance of existing operators ... such as use × or ⋅ instead of * if
> the user asks for it ... or ⊕ for +, ≡ for ===, ⊂ for <, but otherwise
> they would be same operators, with the same precedence table, same operator
> methods... And only for those cases where the user explicitly turns it on
> on a per-character basis...

Sounds like you want a preprocessor, or even a syntax interpreter/translator. ;)

#5 - 08/26/2015 10:55 PM - eike.rb (Eike Dierks)

There have been comments on this feature request.

I'd like to suggest that the unicode symbols from:
Mathematical Operators (U+2200–U+22FF) ⇒ (())
should be informally reserved for future use.

While this should not apply to the other planes of the unicode space as mentioned Mathew's post.

Informally reserving the code would not break any code.
Only very few codes would use the operators a method names,
but even that would not break existing code (at least I believe -- need to test)

It;s more up to add symbols to the parser,
so that these should be recognized as infix operators.

I believe this can easily be done and would not even break existing code.

```
x ∈ set
seta ⊂ setb
seta ⊃ setb
a ⊙ b
```

you get the idea

This is not yet possible,
because the parser limits the infix operators to a few ascii symbols.

I suggest to modify the parser to accept all operators from the unicode operator plane,
as to be allowed as infix operator symbols.

#6 - 08/27/2015 04:36 AM - duerst (Martin Dürst)

Matthew Kerwin wrote:

> Neither for nor against this idea at the moment,

Same here.

> but which "math operator plane" do you mean? A quick read on wikipedia ((())) lists the following potential Unicode 6 blocks:

[shortened]

The right way to do this is not by using blocks, but by using character properties. Unicode General Category (see http://www.unicode.org/reports/tr44/#General_Category_Values) would provide a starting point.

#7 - 08/30/2015 03:05 AM - nobu (Nobuyoshi Nakada)

Eike Dierks wrote:

> I suggest to modify the parser to accept all operators from the unicode operator plane,
> as to be allowed as infix operator symbols.

That does not make sense, all symbols are not infix operators.