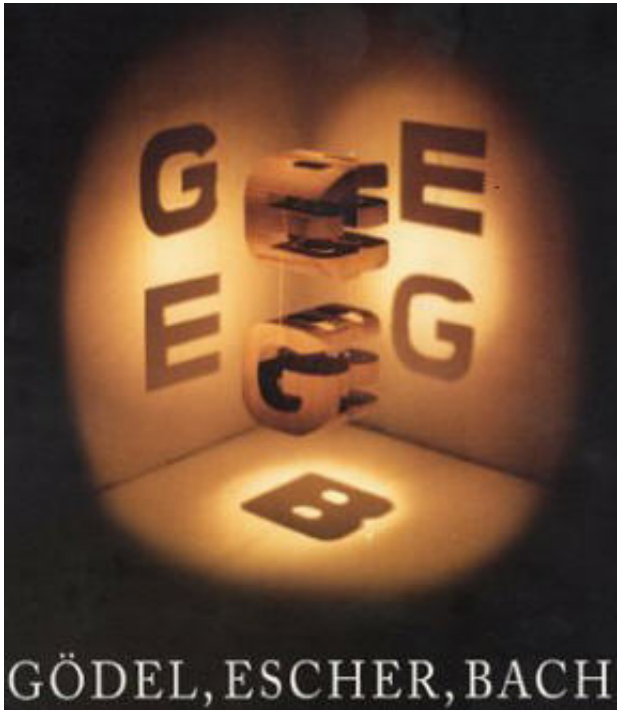
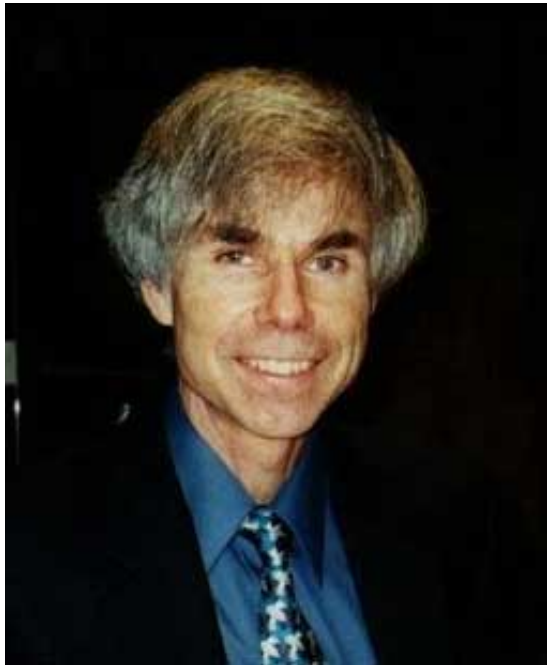
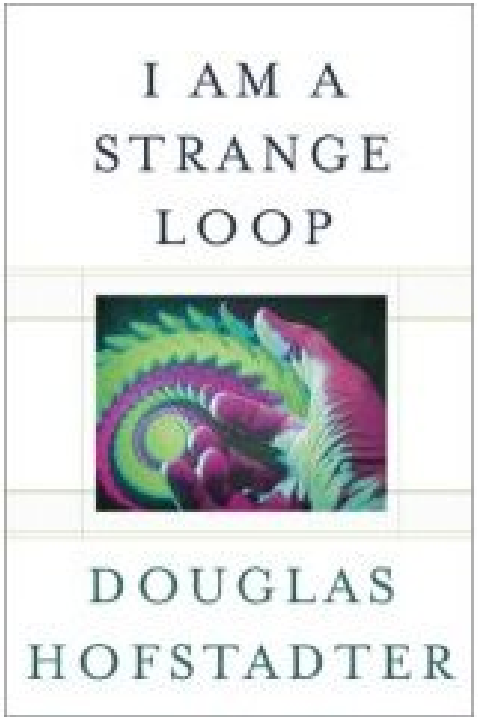
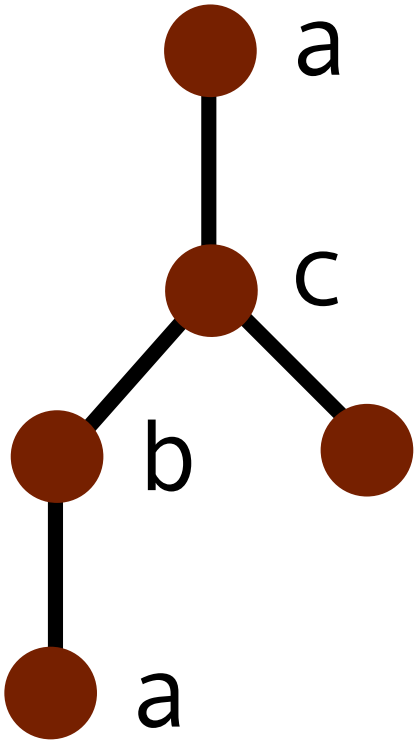
I am a strange loop

Carl Mäsak
2009-11-21


GÖDEL, ESCHER, BACH

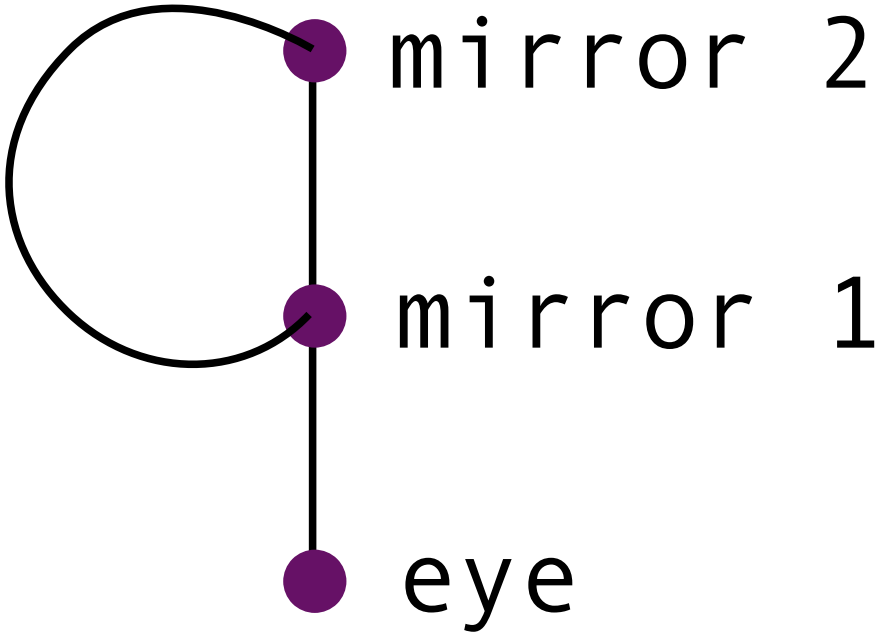


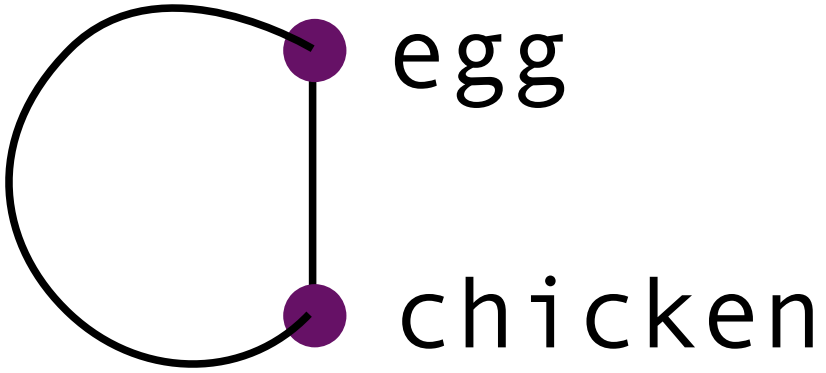

I AM A
STRANGE
LOOP
DOUGLAS
HOFSTADTER

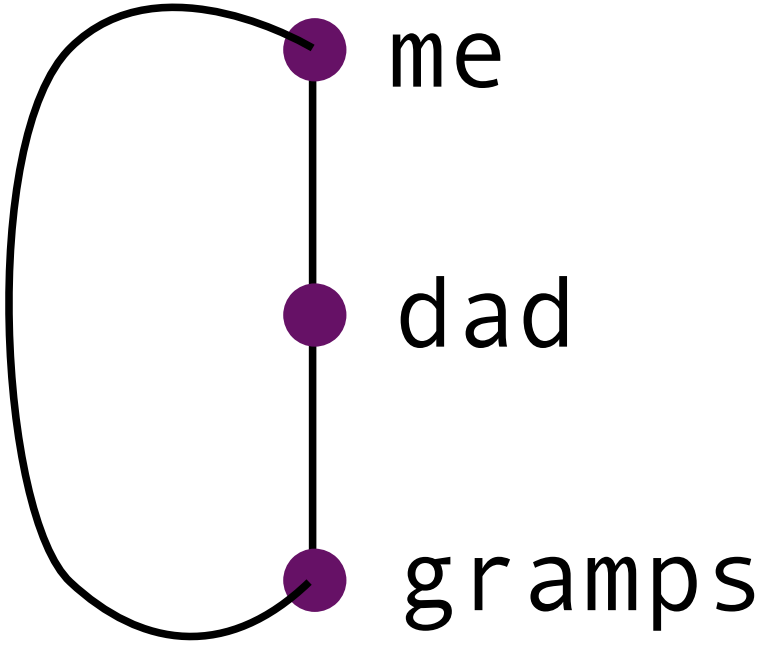


Strange Loop


١٠٥
١٠٥
١٠٥
١٠٥
١٠٥
٦٠١
٦٠١
٦٠١
٦٠١


mirror 2
mirror 1
eye


egg
chicken


me
dad
gramps

Perl 6 regexes

S05

many things are the same as in Perl 5

\w, \s, \d, \W, \S, \D

`a?!`	`a??`	`a?:`
`a*!`	`a*?`	`a*:`
`a+!`	`a+?`	`a+:`

greedy	~~eager~~ frugal	no backtracking

`a?:!`	`a?:?`	`a?:`
`a*:!`	`a*:?`	`a*:`
`a+:!`	`a+:?`	`a+:`

greedy	~~eager~~ frugal	no backtracking

(capturing)

$a \mid b$

(?:non-capturing)

[non-capturing]

'non-capturing'

moose*, 'moose'*

alnums can go unquoted

everything else needs either 'quotes' or \\

[aeiou]

<[aeiou]>

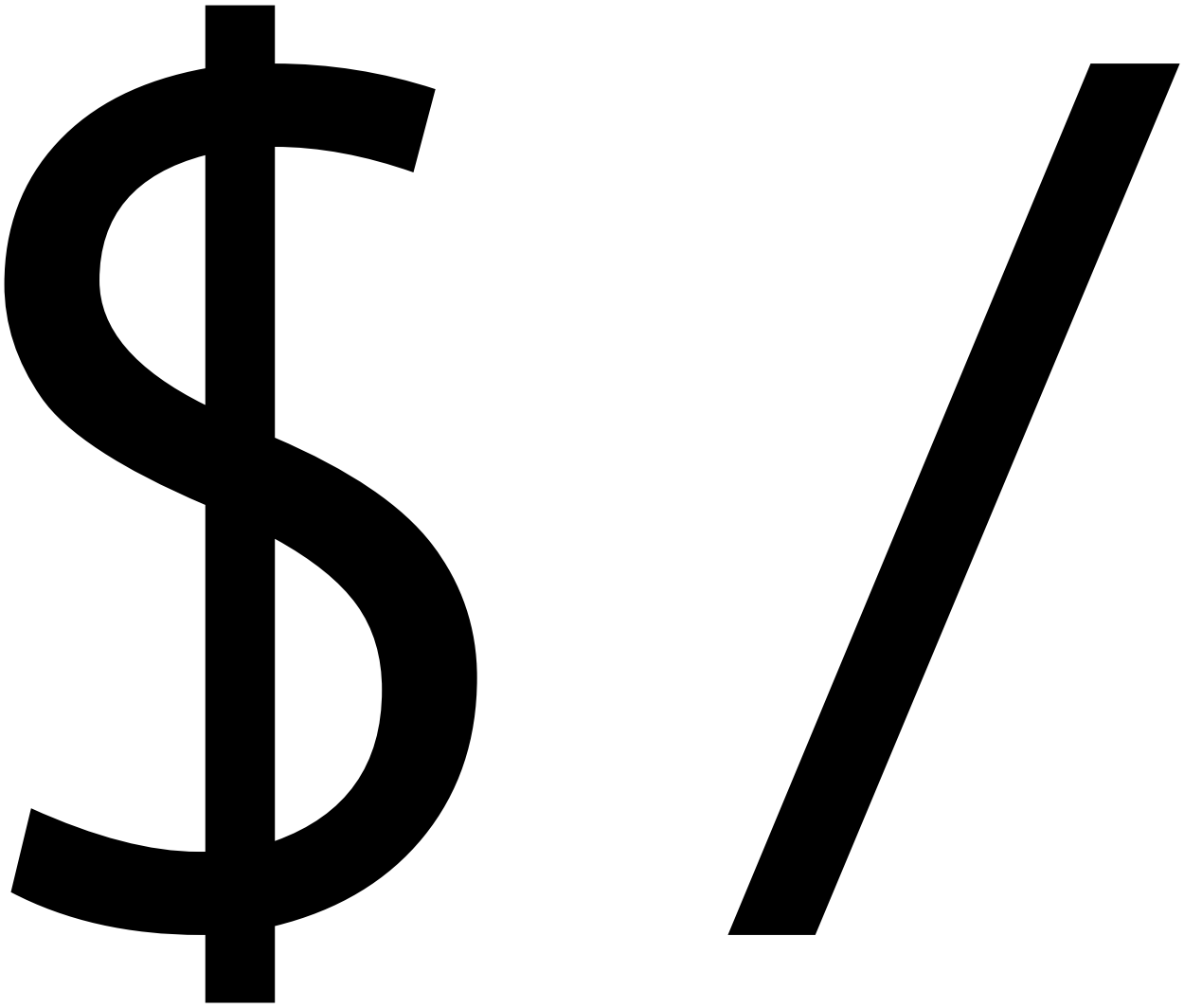
[a-z]

<[a..z]>

^ and $

^^ and $$

. matches anything



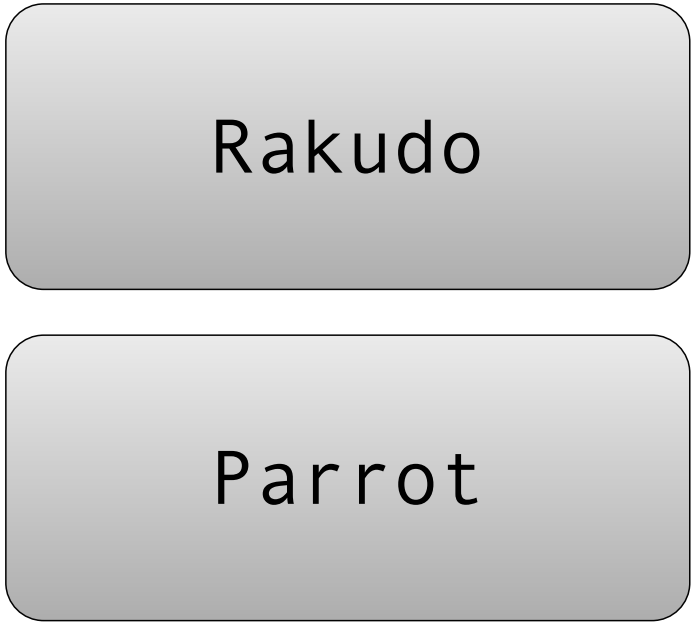
```
$/[0]    $/[1]    $/[2]

$0       $1       $2

$0[1][2]     $1['foo']
```

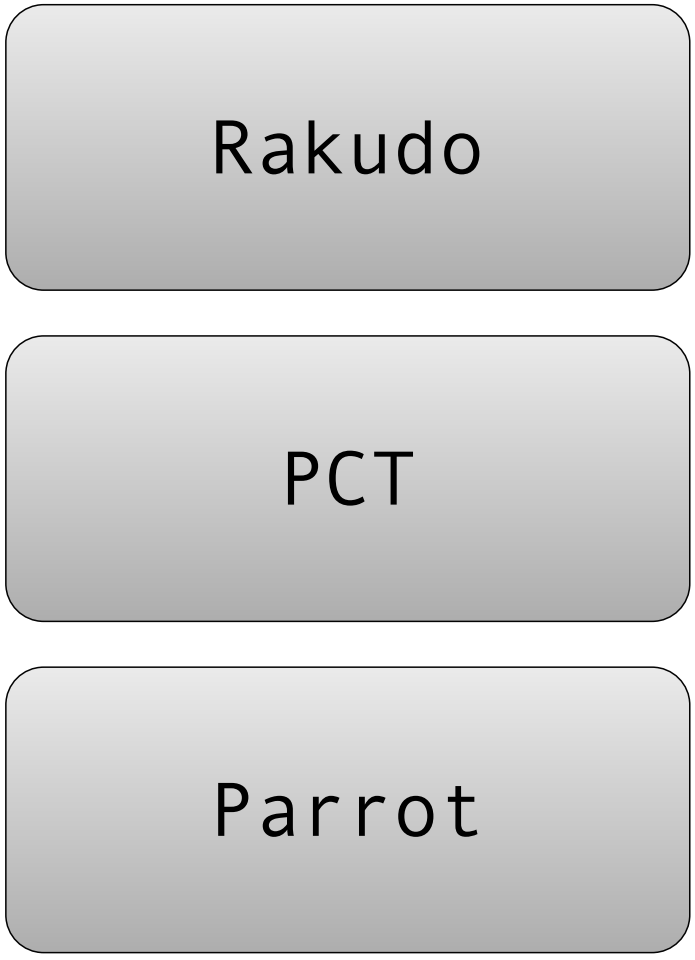
```
grammar G {

  token a { <b> }

  token b { <c> }

  token c { 'OH HAI' }

}
```

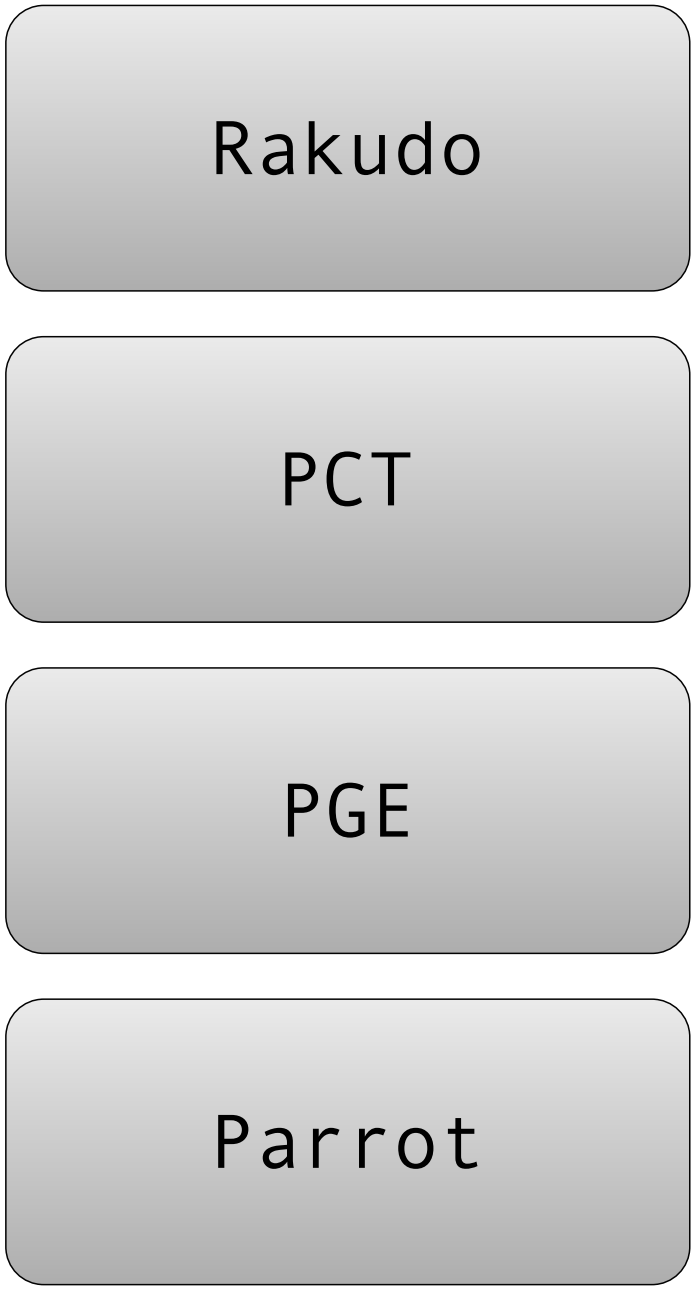
```
class C {

  method a { self.b }

  method b { seld.c }

  method c { say 'OH HAI' }

}
```

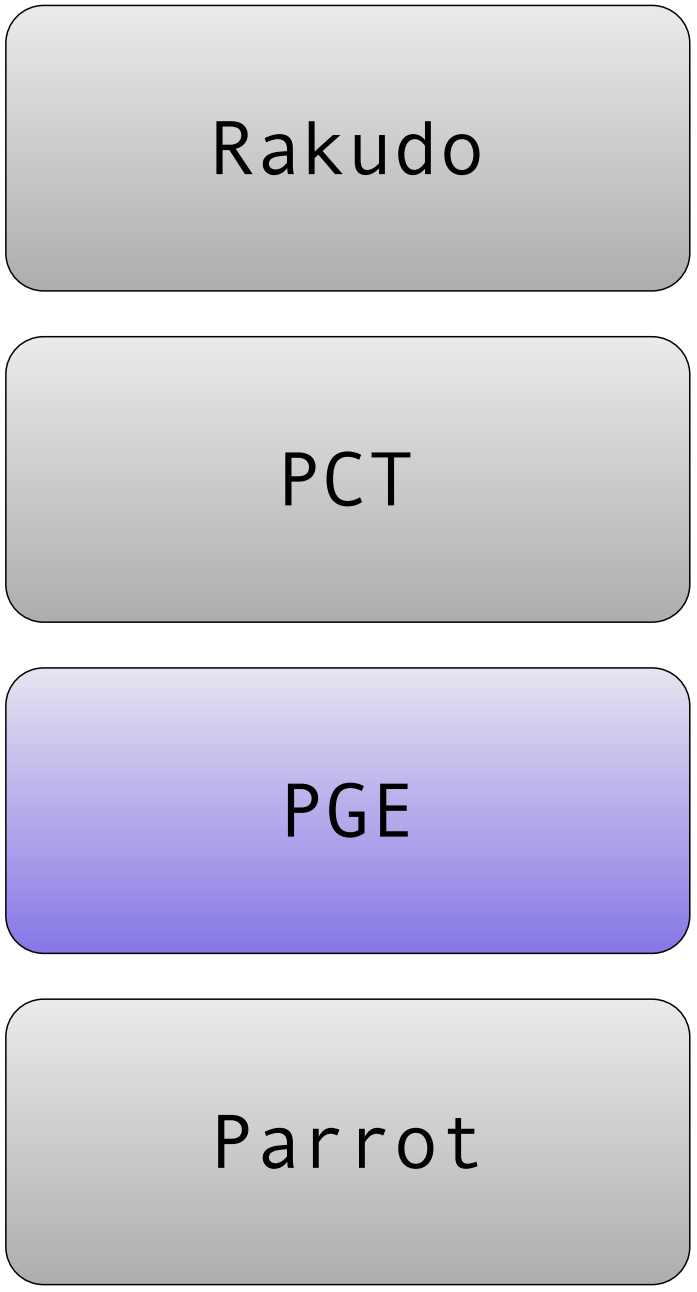
"Any grammar regex...

...is really just a kind of method" - S05

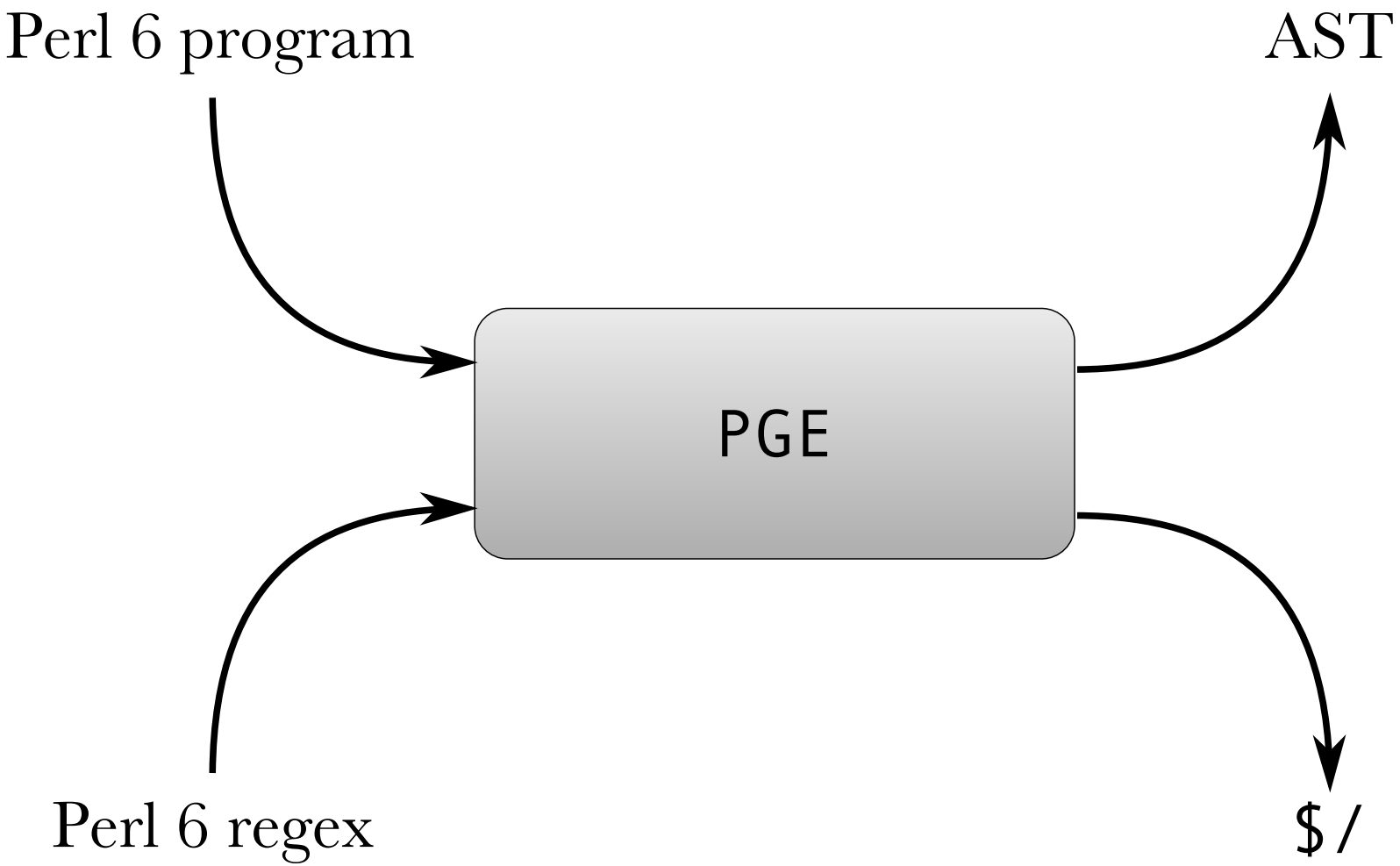

Rakudo
Parrot


Rakudo
PCT
Parrot


Rakudo
PCT
PGE
Parrot


Rakudo
PCT
PGE
Parrot

PGE always sort of fascinated me


Perl 6 program
AST
PGE
Perl 6 regex
$/

code reviews

like a book review

comment

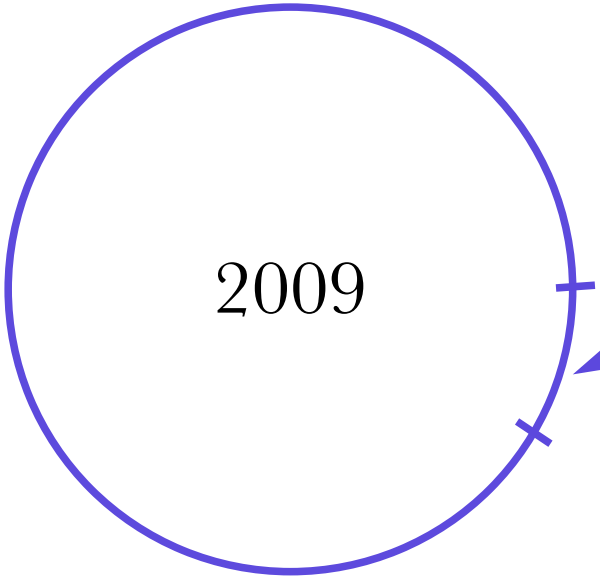
contribute

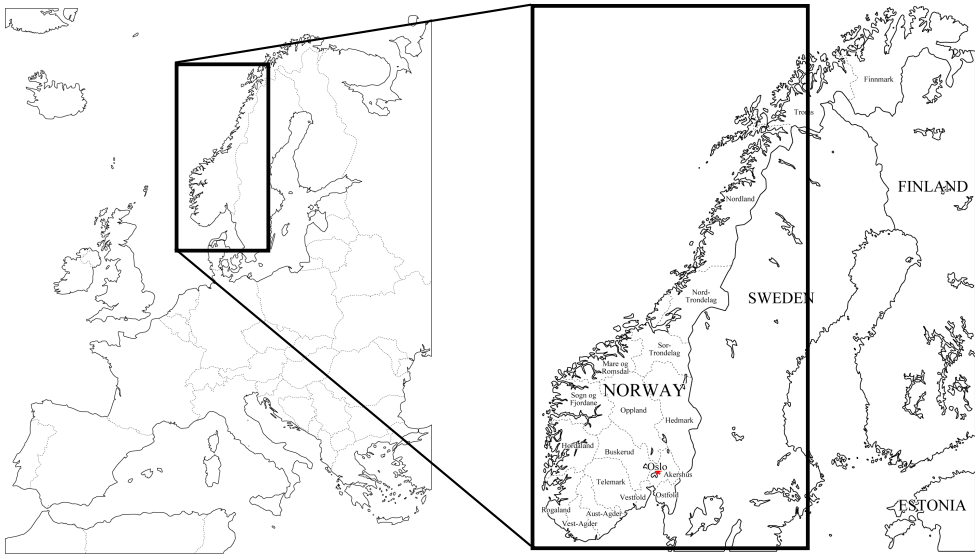
grok

increase bus number

rewrite of the community

might as well get good habits





Nordic Perl Workshop

"Building compilers
with the
Parrot Compiler Toolkit"

"PGE is actually
quite small –
about 5.4 k
lines of code."

— Patrick Michaud

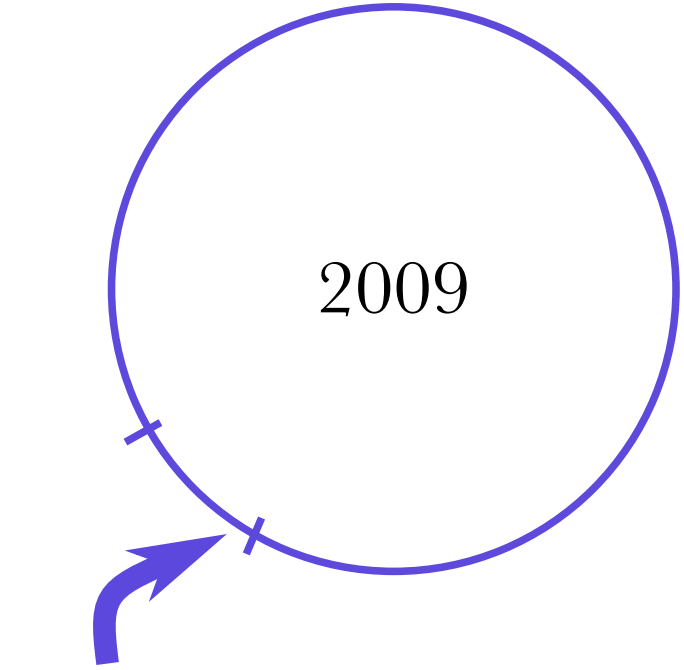

Wow!

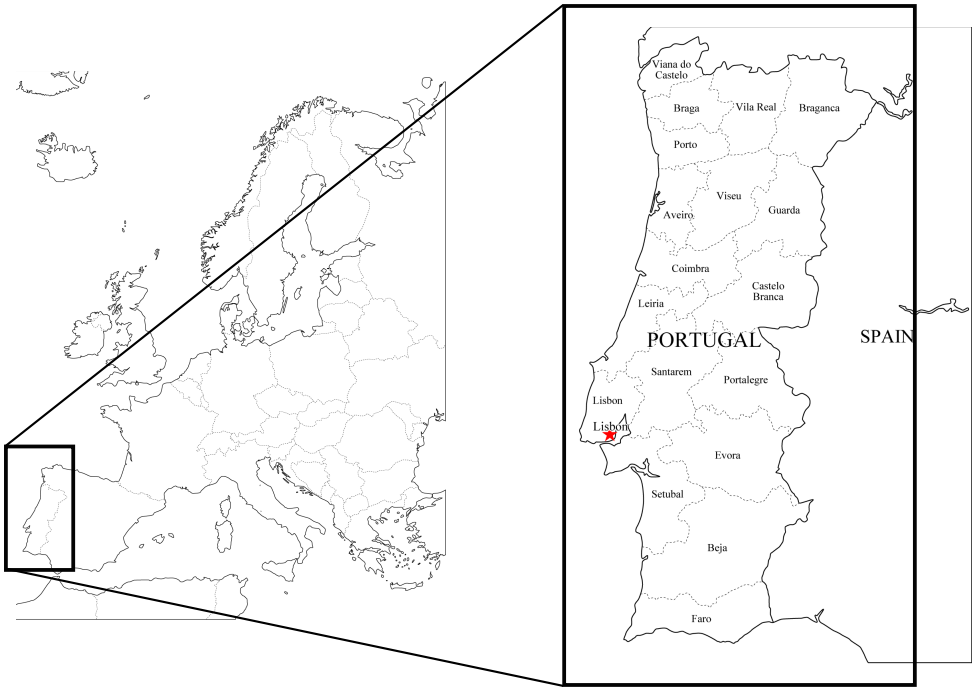
I've got to do a code review of that!

PIR

looks like punchline

Perl 5 & asm





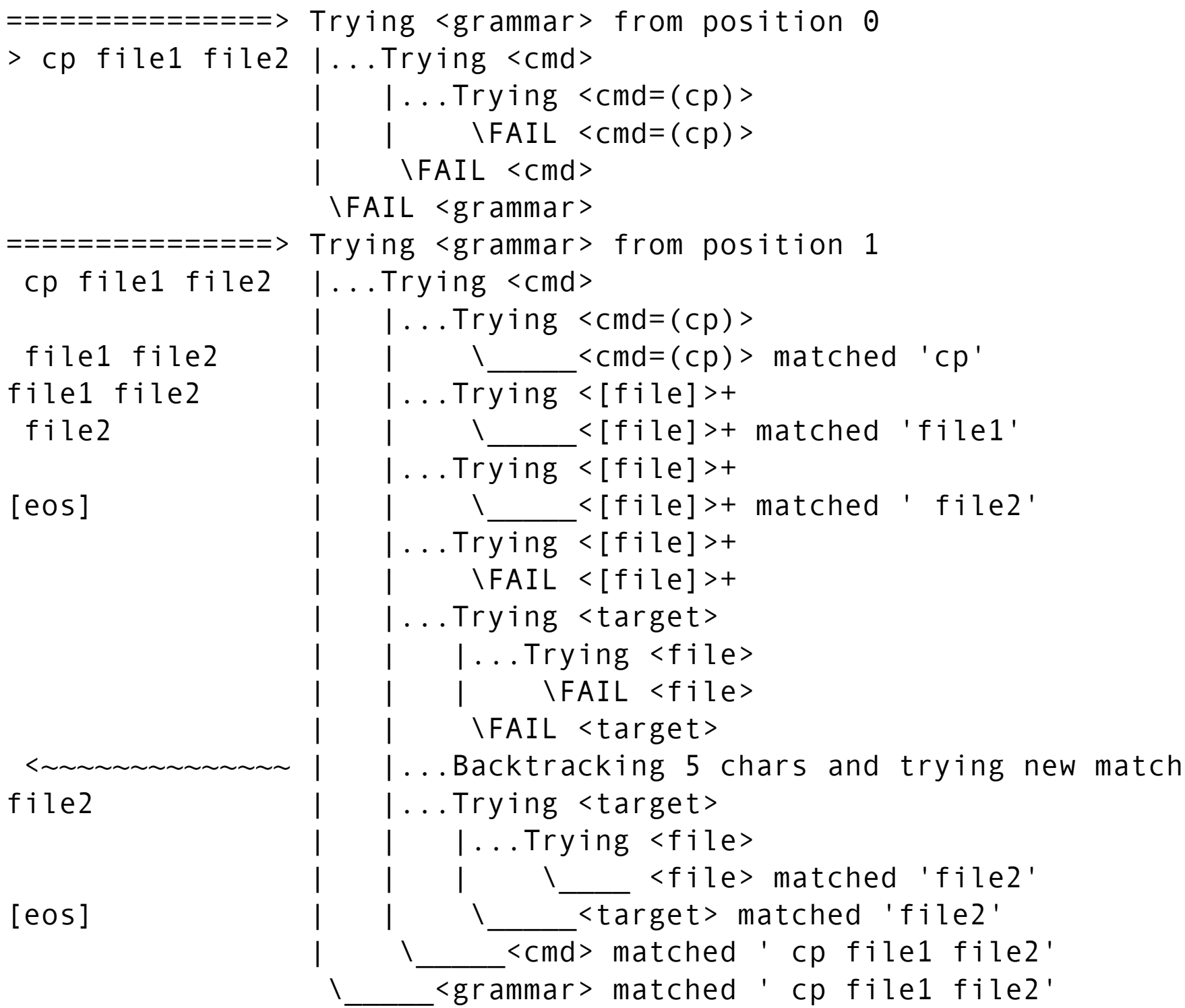
YAPC::EU

"The Missing Link"

He had reinvented Perl 6 grammars!

In Perl 5!

Not only that...


==========> Trying <grammar> from position 0
> cp file1 file2 |...Trying <cmd>
|...Trying <cmd=(cp)>
\FAIL <cmd=(cp)>
\FAIL <cmd>
\FAIL <grammar>
==========> Trying <grammar> from position 1
cp file1 file2 |...Trying <cmd>
|...Trying <cmd=(cp)>
file1 file2 _____<cmd=(cp)> matched 'cp'
file1 file2 |...Trying <[file]>+
file2 _____<[file]>+ matched 'file1'
|...Trying <[file]>+
[eos] _____<[file]>+ matched ' file2'
|...Trying <[file]>+
\FAIL <[file]>+
|...Trying <target>
|...Trying <file>
\FAIL <file>
\FAIL <target>
<~~~~~~~~~~~~~~~ |...Backtracking 5 chars and trying new match
file2 |...Trying <target>
|...Trying <file>
____ <file> matched 'file2'
[eos] _____<target> matched 'file2'
_____<cmd> matched ' cp file1 file2'
_____<grammar> matched ' cp file1 file2'

So jealous!

Perl 6 is supposed to have the coolest tech!

started looking at PGE

Reading compilers/pge/PGE/Exp.pir -- there's something slightly unnerving about PIR code that produces PIR code. It's also exquisite. ♥

2:46 PM Aug 24th from Tweetie

Delete

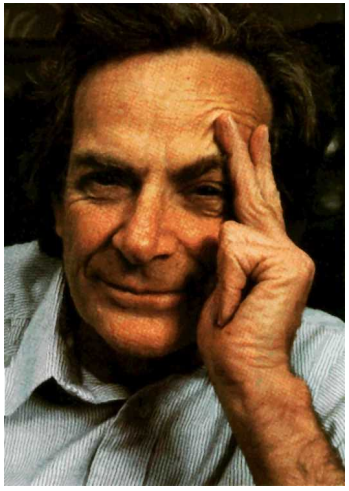
carlmasak

Carl Mäsak

realized I didn't understand it

not much parser experience

What I cannot create,
I do not understand.



— Richard Feynman

maybe the problem was

that I hadn't created PGE

decided to write PGE

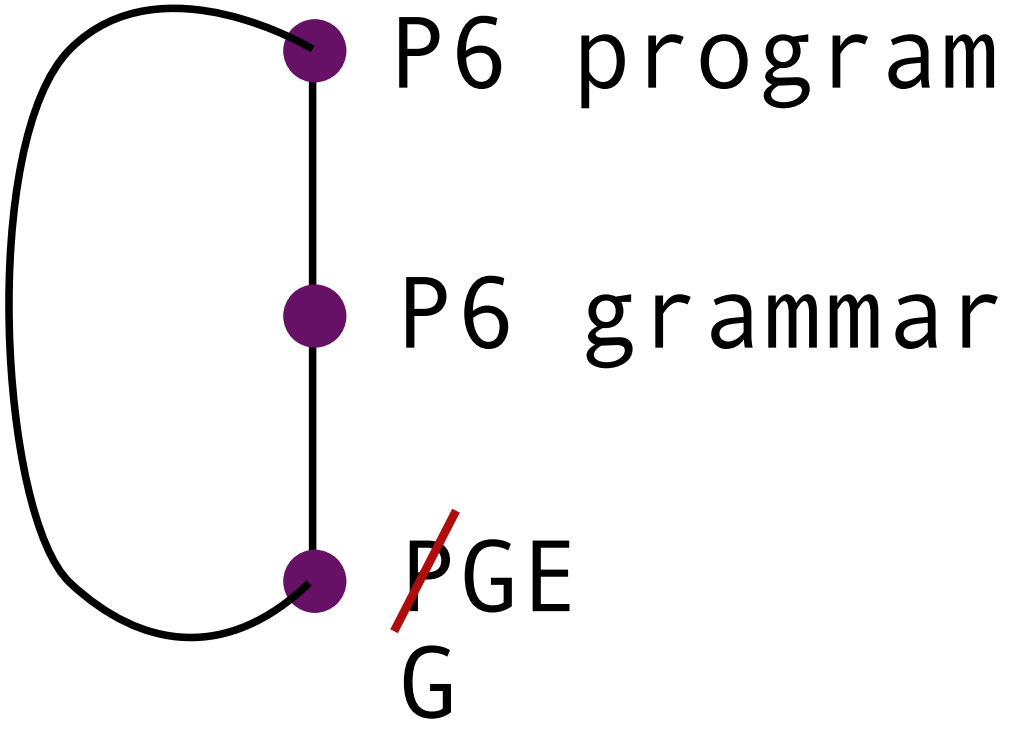
...in Perl 6

deranged

maniacal

insane

had to try it


P6 program
P6 grammar
PGE
G

also

try out the debugger idea

web application debugger

parse Perl 6 with Perl 6

static Perl 6 analyzer

macro preprocessor

Perl 6 can't parse Perl 6! (yet)

PGE and STD.pm

piggybacking Rakudo/PGE

no can do

parsing 'say "hello world"' works

class def doesn't work

tangled

leveraging STD

fails on technicality

STD outputs YAML

no YAML parser

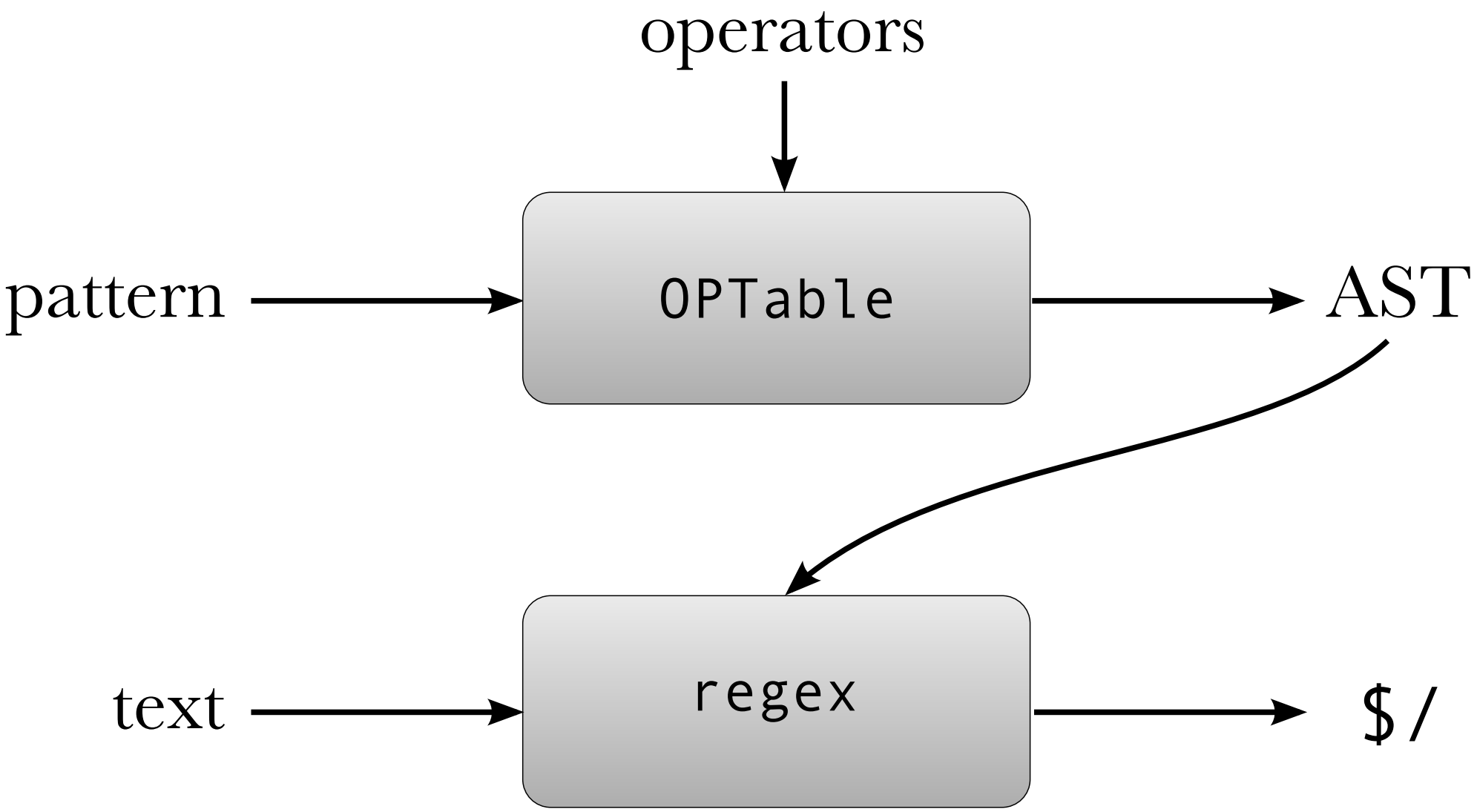
Exp, Match, OPTable (?), Perl6Regex

started out simple

soon pretty hairy

turns out

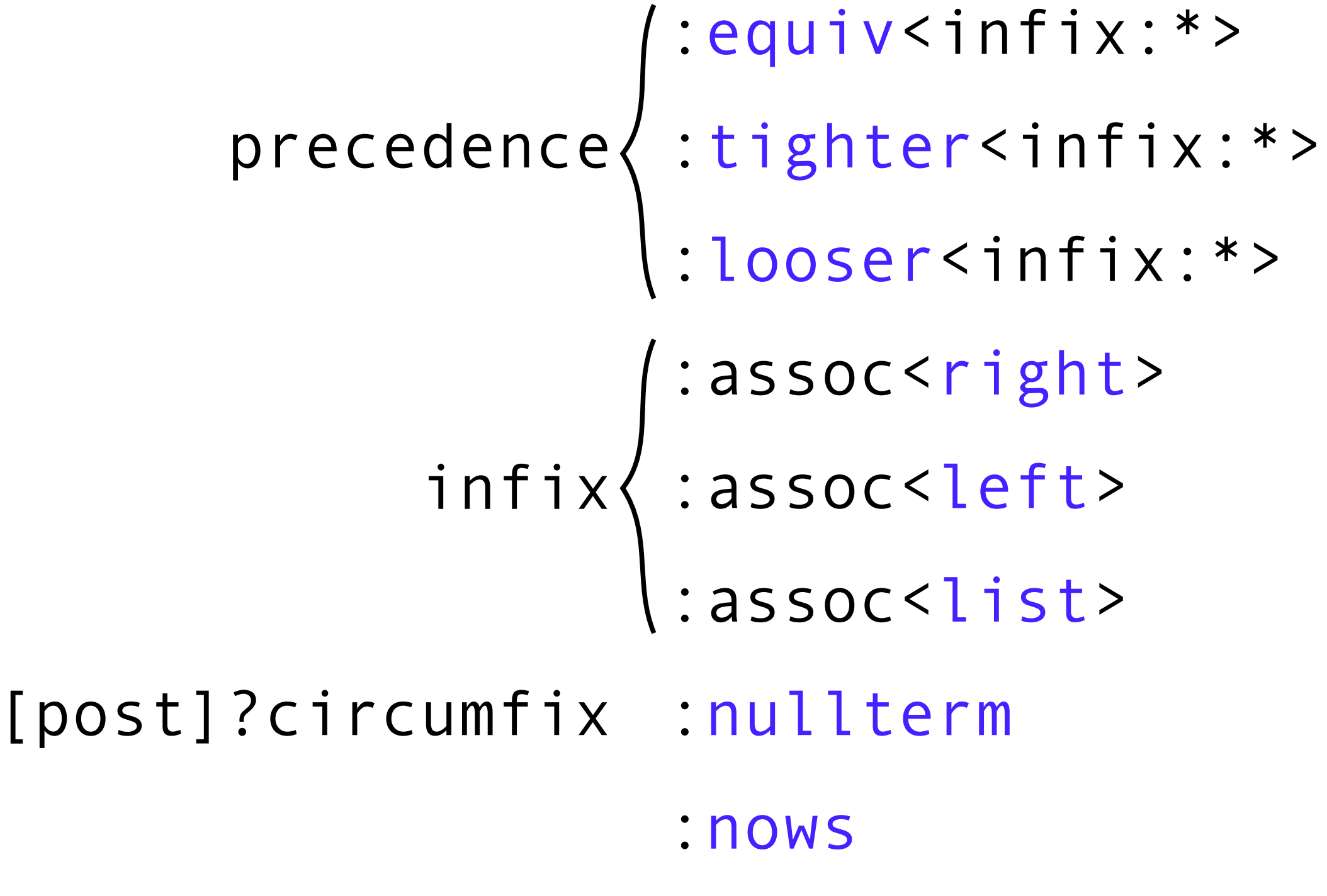
there are two parsers


operators
pattern
OPTable
AST
text
regex
$/

term	a

---

infix	a * b
prefix	--a
postfix	a++
circumfix	(a+b) * c
postcircumfix	a[42]


precedence
:equiv<infix:*>
:tighter<infix:*>
:looser<infix:*>
infix
:assoc<right>
:assoc<left>
:assoc<list>
[post]?circumfix
:nullterm
:nows

Op stack:

Expression: a + b

a

Op stack:

Expression: a + b

a

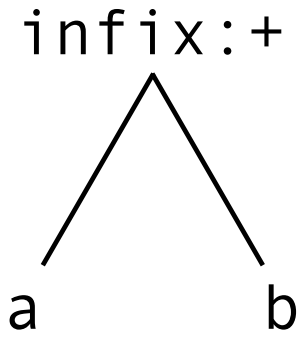
Op stack: infix:<+>

Expression: a + b

a b

Op stack: infix:<+>

Expression: a + b



Op stack:

Expression: a + b

Op stack:

Expression: a + b + c

a

Op stack:

Expression: a + b + c

a

Op stack:

infix:+

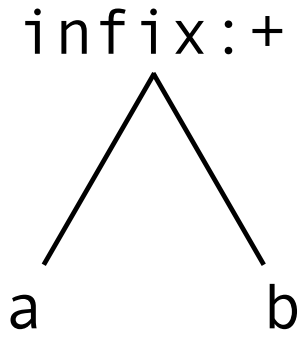
Expression:

a + b + c

a b

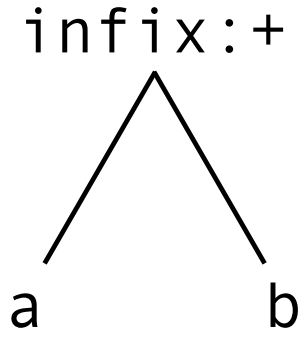
Op stack: infix:+

Expression: a + b + c



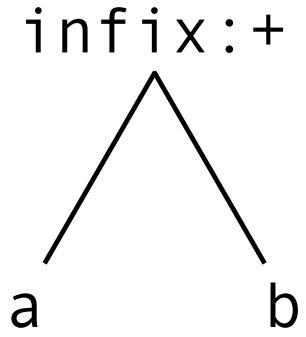
Op stack:

Expression: a + b + c



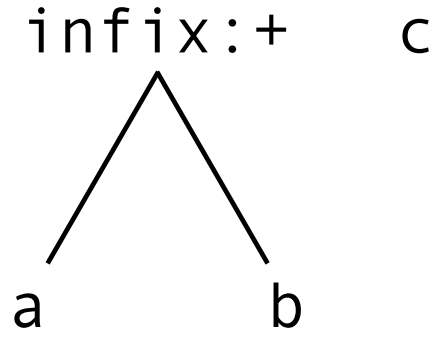
Op stack:

Expression: a + b + c



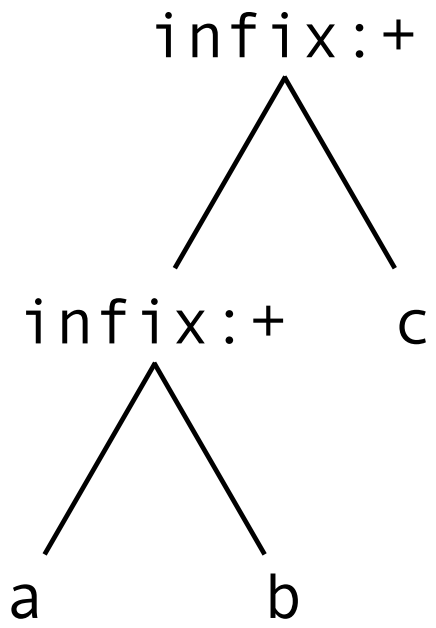
Op stack: infix:+

Expression: a + b + c



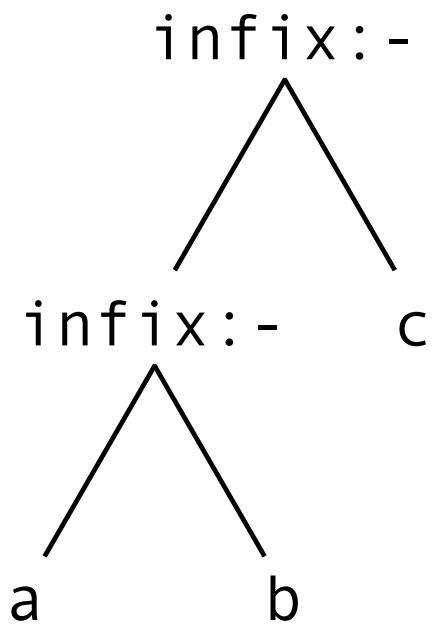
Op stack: infix:+

Expression: a + b + c



Op stack:

Expression: a + b + c



Op stack:

Expression: a - b - c

Op stack:

Expression: a ** b ** c

a

Op stack:

Expression: a ** b ** c

a

Op stack: infix:**

Expression: a ** b ** c

a b

Op stack: infix:**

Expression: a ** b ** c

a b

infix:**

Op stack: infix:**

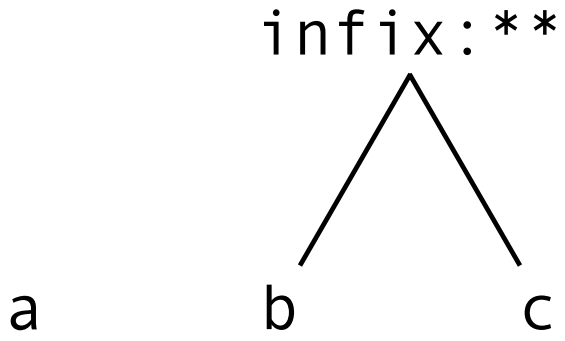
Expression: a ** b ** c

a b c

infix:**

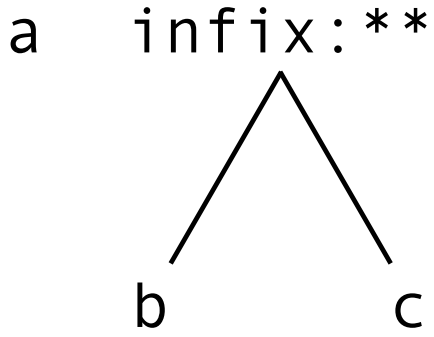
Op stack: infix:**

Expression: a ** b ** c



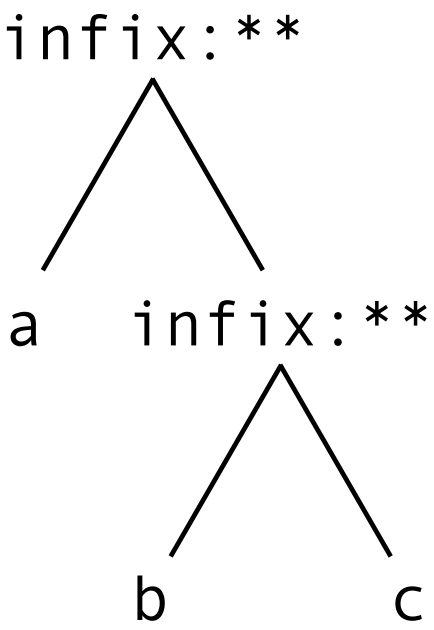
Op stack: infix:**

Expression: a ** b ** c



Op stack: infix:**

Expression: a ** b ** c



Op stack:

Expression: a ** b ** c

stack \ current token	close	infix	prefix	postfix	circfix	postcir
empty	E	S	S	S	S	S
close	R	P	S	P	X	P
infix	R	P/A	S	P	S	P
prefix	R	P	S	P	S	P
postfix	R	R	X	R	S	S
circfix	S	S	S	S	S	S
postcir	S	S	S	S	S	S

translating PIR to Perl 6

dijkstrifying

turning gotos into ifs and whiles

everything expressible as ifs and whiles

...doesn't mean you should!

continuations

Parrot has'em

PGE uses them

```
   <masak> I increasingly long for first-class continuations in Perl 6, and it
           just feels so ironic that Rakudo then is built upon a VM that has'em.
   <masak> and still they're not even planned to be reachable, except awkwardly
           through gather/take!
...
   <masak> a follow-up on my continuations rant from yesterday: S04:627
           specifies C<gather> as being lazy. how will this be pulled off on a
           VM without continuations?
...
   <masak> I'll settle for 'delimited' continuations, even though I don't know
           exactly what they are.
...
<TimToady> masak: on VMs without continuations, laziness must be emulated with
           closures, as STD does on Perl 5.
   <masak> TimToady: I'm running up against some situations where first-class
           continuations would be very useful. that's why I'm thinking of this
           lately.
   <masak> or maybe not even first-class, but some mechanism such as Ruby's
           'yield'.
   <masak> it feels strange that we have gather/take and pipes, but no local way
           to make the same kind of control flow.
<TimToady> that's because I like to keep my laziness orthogonal to sub calls
   <masak> I see.
```

```
<TimToady> and I very much don't like co-routines that change their signature
           depending on whether they're initting or continuing, though making
           yield return new arguments is a sop to fix that
   <masak> hm, yes. co-routines changing signatures feels wrong, I agree.
   <masak> Seaside and Lift are two web applications which make good use of
           continuations. PGE and other parsers use them liberally. I'm still
           trying to figure out which contortions are necessary to port such
           applications to pure Perl 6.
<TimToady> I thought those were just "web continuations"
   <masak> the point is that they're syntactically supported.
<TimToady> well, basically, lazy lists are our continuation syntax primitive
<TimToady> lazy lists are also a poor-man's monad
<TimToady> making sure time happens in the right order :)
   <masak> TimToady: either I'm missing something major from the 'lazy lists are
           our continuation syntax primitive' viewpoint, or I'm simply
           dissatisfied with it.
<TimToady> continuations are like naked singularities, but the event horizon
           is a lazy list
```

"Continuations are like naked singularities, but the event horizon is a lazy list."

— Larry Wall

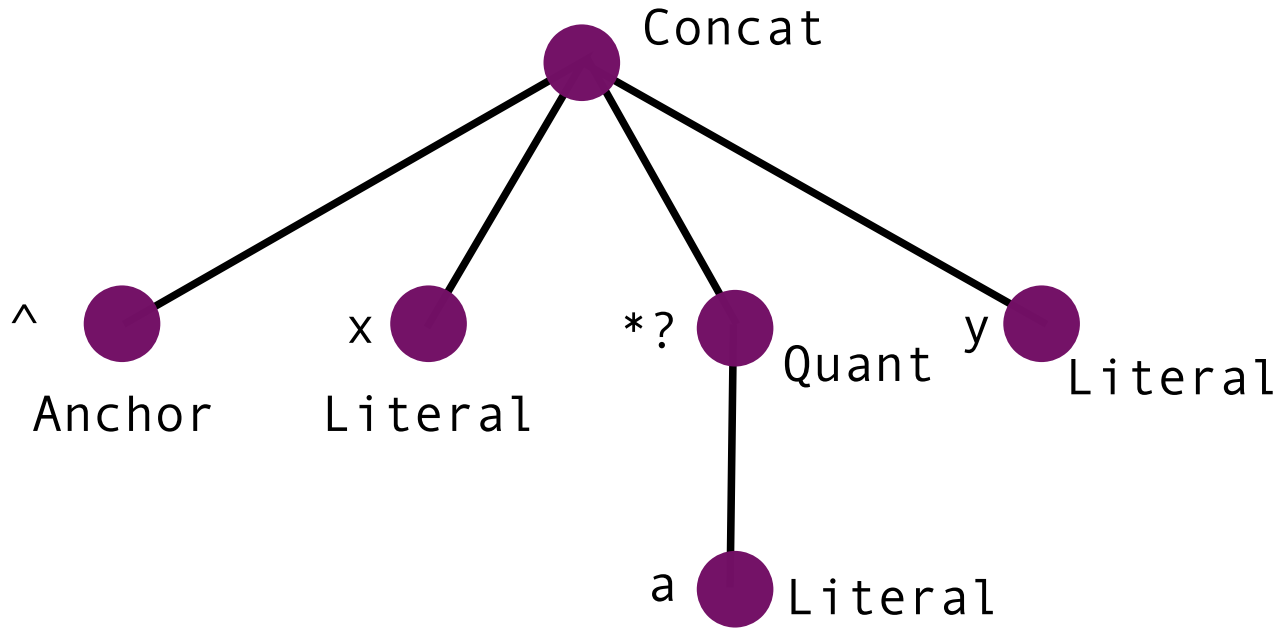
no dice

no continuations

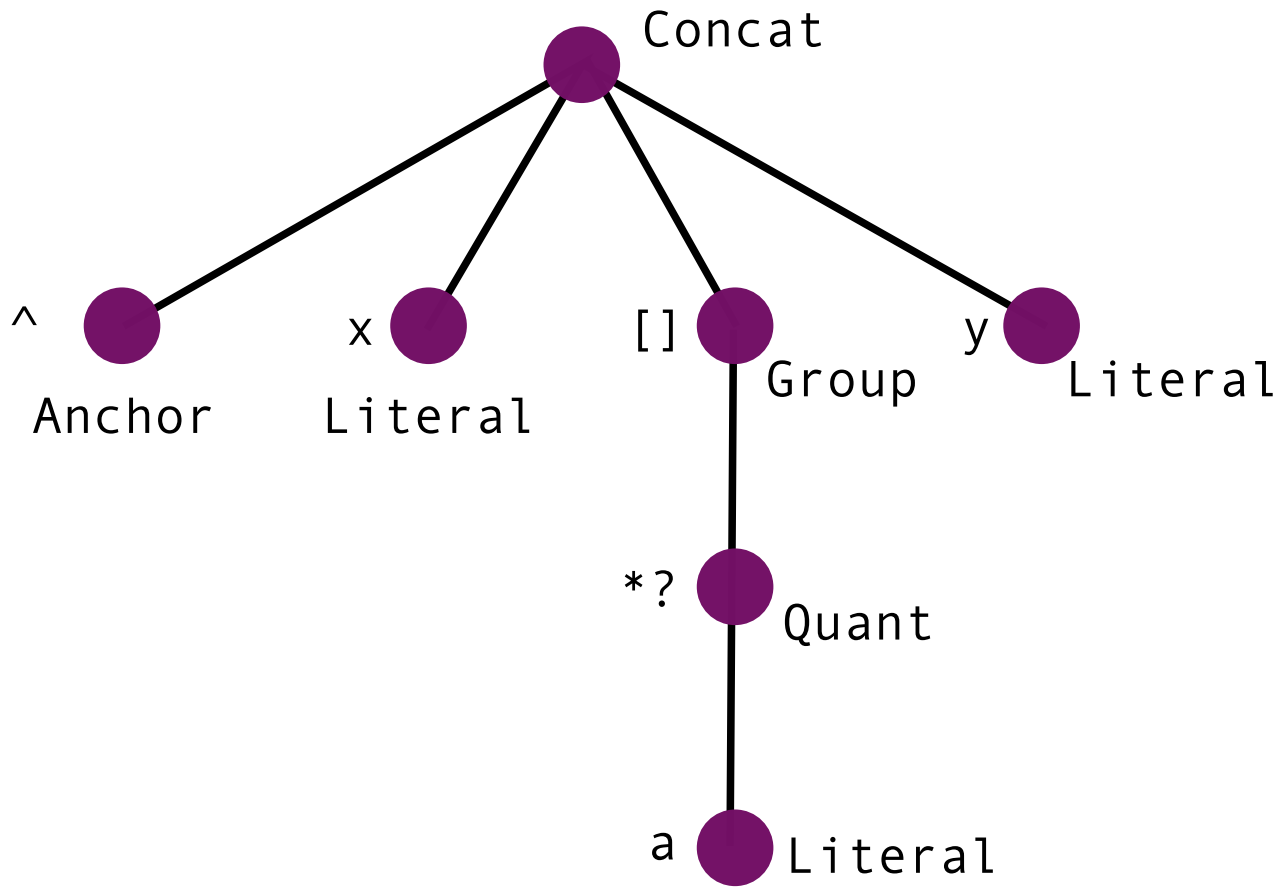
workaround?

yes!

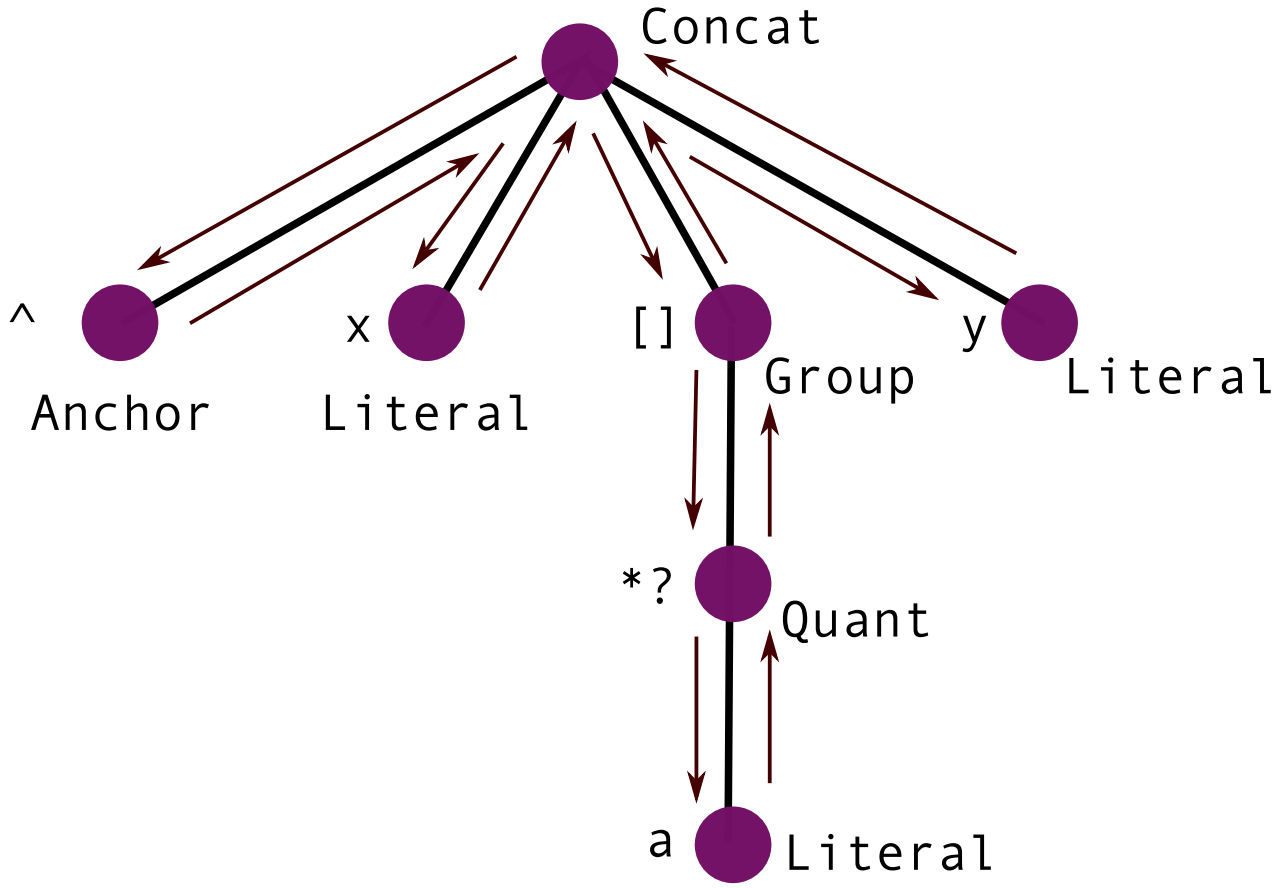
^ x a*? y



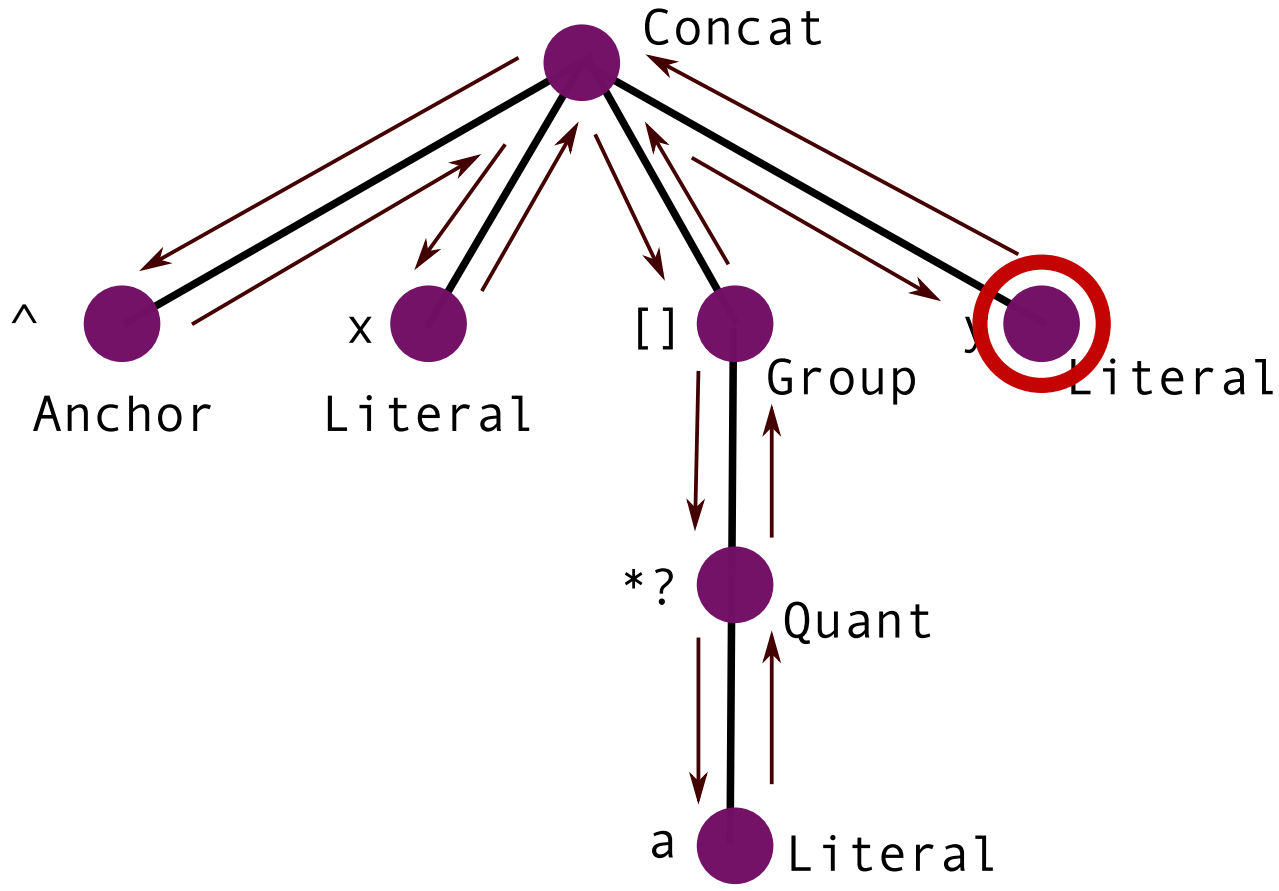
^ x [a*?] y



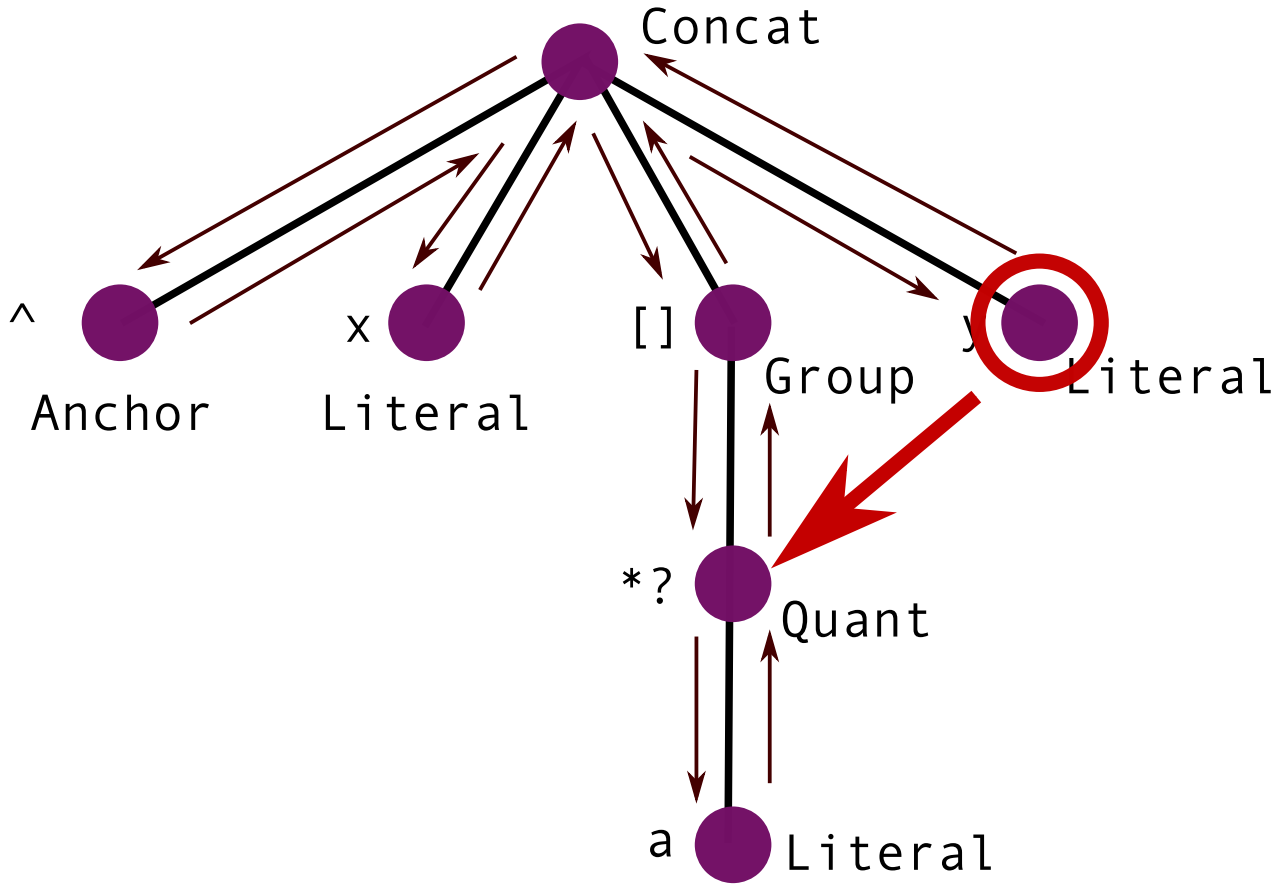
^ x [a*?] y



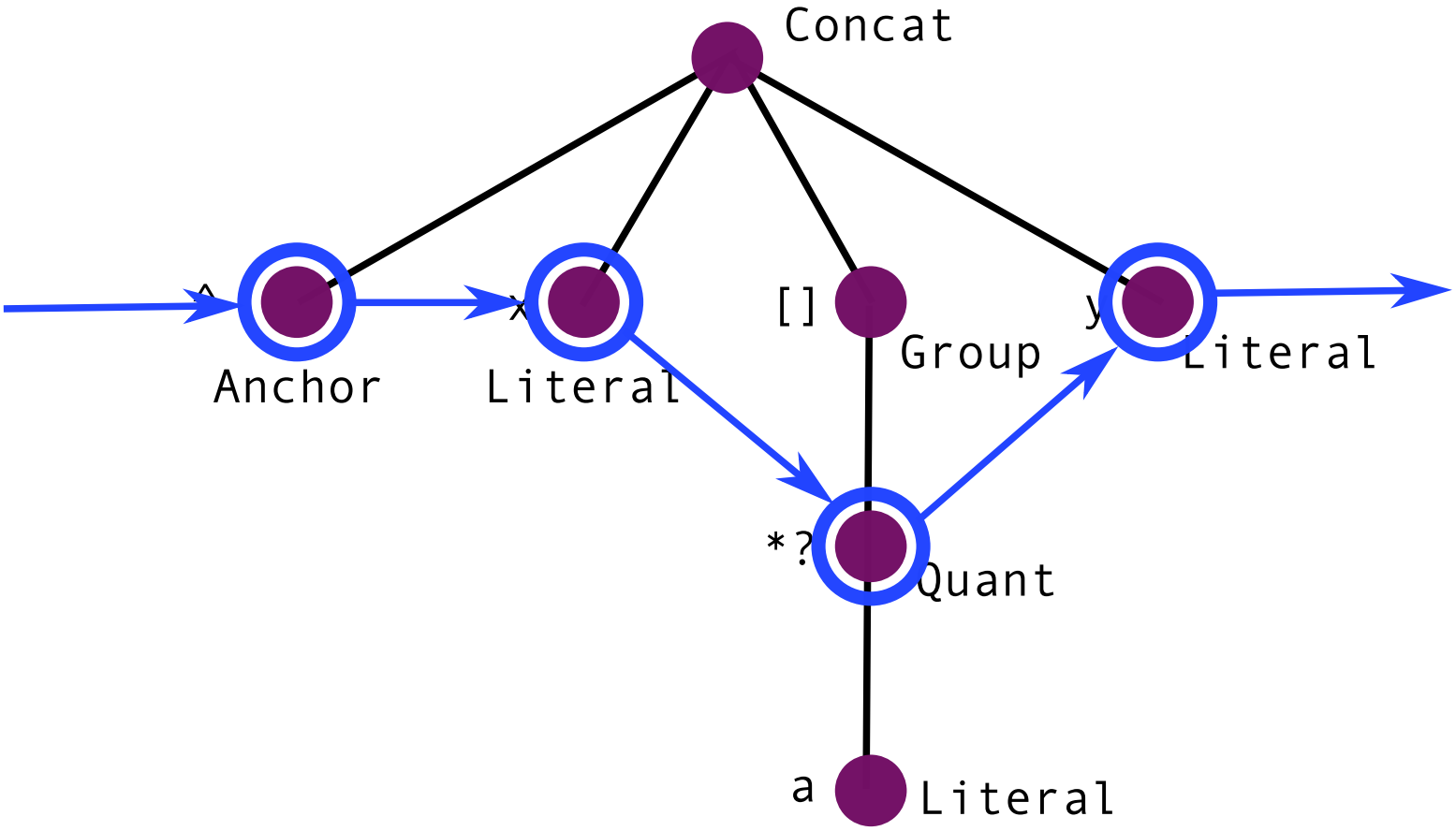
^ x [a*?] y



^ x [a*?] y



^ x [a*?] y



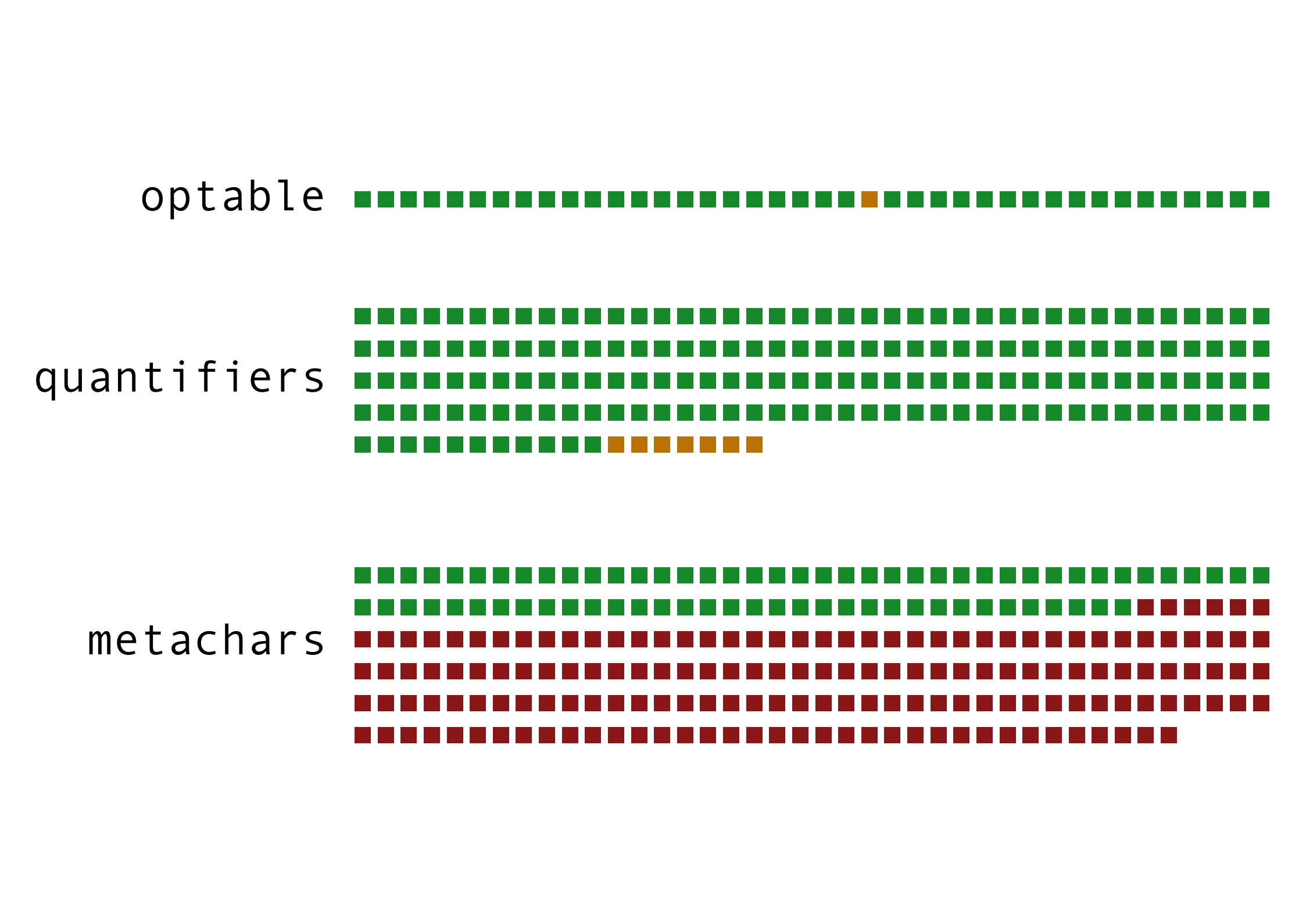
large app

Hitting limits again

```
rc/gc/api.c:248: failed assertion 'PObj_is_PMC_TEST(obj)'
Backtrace - Obtained 31 stack frames (max trace depth is 32).
0   libparrot.dylib                     0x005fc2ad Parrot_do_check_events + 173
1   libparrot.dylib                     0x005fc417 Parrot_confess + 151
2   libparrot.dylib                     0x006097a7 Parrot_gc_mark_PMC_alive_fun + 135
3   libparrot.dylib                     0x007a05f5 Parrot_CodeString_get_isa + 1957
4   libparrot.dylib                     0x0060d4fe Parrot_is_blocked_GC_sweep + 6398
5   libparrot.dylib                     0x006098c7 Parrot_gc_mark_PObj_alive + 183
6   libparrot.dylib                     0x0060eac1 Parrot_is_blocked_GC_sweep + 11969
7   libparrot.dylib                     0x0060eb8e Parrot_is_blocked_GC_sweep + 12174
8   libparrot.dylib                     0x0060ec0c Parrot_is_blocked_GC_sweep + 12300
9   libparrot.dylib                     0x0060d076 Parrot_is_blocked_GC_sweep + 5238
10  libparrot.dylib                     0x0060c213 Parrot_is_blocked_GC_sweep + 1555
11  libparrot.dylib                     0x0060c355 Parrot_is_blocked_GC_sweep + 1877
12  libparrot.dylib                     0x0060a7e3 Parrot_gc_mark_STRING_alive_fun + 3827
13  libparrot.dylib                     0x0060c5e8 Parrot_is_blocked_GC_sweep + 2536
14  libparrot.dylib                     0x0060c750 Parrot_is_blocked_GC_sweep + 2896
15  libparrot.dylib                     0x00609dfd Parrot_gc_mark_STRING_alive_fun + 1293
16  libparrot.dylib                     0x00673156 pmc_reuse + 566
17  libparrot.dylib                     0x00673278 pmc_new + 232
18  libparrot.dylib                     0x0067d52e new_ret_continuation_pmc + 78
19  libparrot.dylib                     0x0061d92d new_runloop_jump_point + 765
20  libparrot.dylib                     0x0061e2a5 Parrot_run_meth_fromc_args_reti + 213
21  libparrot.dylib                     0x007c135a Parrot_NameSpace_get_isa + 30186
22  libparrot.dylib                     0x0057dd2d Parrot_str_from_int + 3469
23  libparrot.dylib                     0x00676037 enable_event_checking + 1991
24  libparrot.dylib                     0x00674f2a Parrot_runcore_switch + 3978
25  libparrot.dylib                     0x0061d7b8 new_runloop_jump_point + 392
26  libparrot.dylib                     0x0061dae6 new_runloop_jump_point + 1206
27  libparrot.dylib                     0x0061e7ea Parrot_runops_fromc_args + 186
28  libparrot.dylib                     0x005f9101 Parrot_runcode + 337
29  perl6                               0x00001ac9 start + 505
30  perl6                               0x00001906 start + 54
Abort trap
```

segfaults

sudden halts


optable
quantifiers
metachars

LIVE DEMO

Attribution

Bamboo boat	http://www.flickr.com/photos/magical-world/1812650937
Infinite mirrors	http://www.flickr.com/photos/snorkel/61787270
Chicken and egg	http://www.flickr.com/photos/kioan/3449402515
Grandfather	http://www.flickr.com/photos/deepblue66/129998451
masak and pmichaud	http://www.flickr.com/photos/szbalint/3812311510
Ice	http://www.flickr.com/photos/ytwhitelight/184253655
Fishing	http://www.flickr.com/photos/magical-world/1813491324

The above images are all licensed under a Creative Commons license on Flickr.
All other images were found on the Web and are considered as falling under "fair use".

Thank you

Any loopy questions?