Athlete Programming

Превед \o/

Carl Mäsak


biking
I like
cooking
writing music

spoken languages

I like Perl 6

programming languages

today:

a certain type of problem

not about 'Perl 6 is awesome'

(well, yes, actually)

(accidentally)

my interest

software that's aware

not "intelligent"

"smart"
phones,
software, etc

really
stupid
software

real
artificial
intelligence

2011 2020 2035 2038

tell the code what's going on

push structure into the code

hard problems

how to solve a hard problem

(in two simple steps)

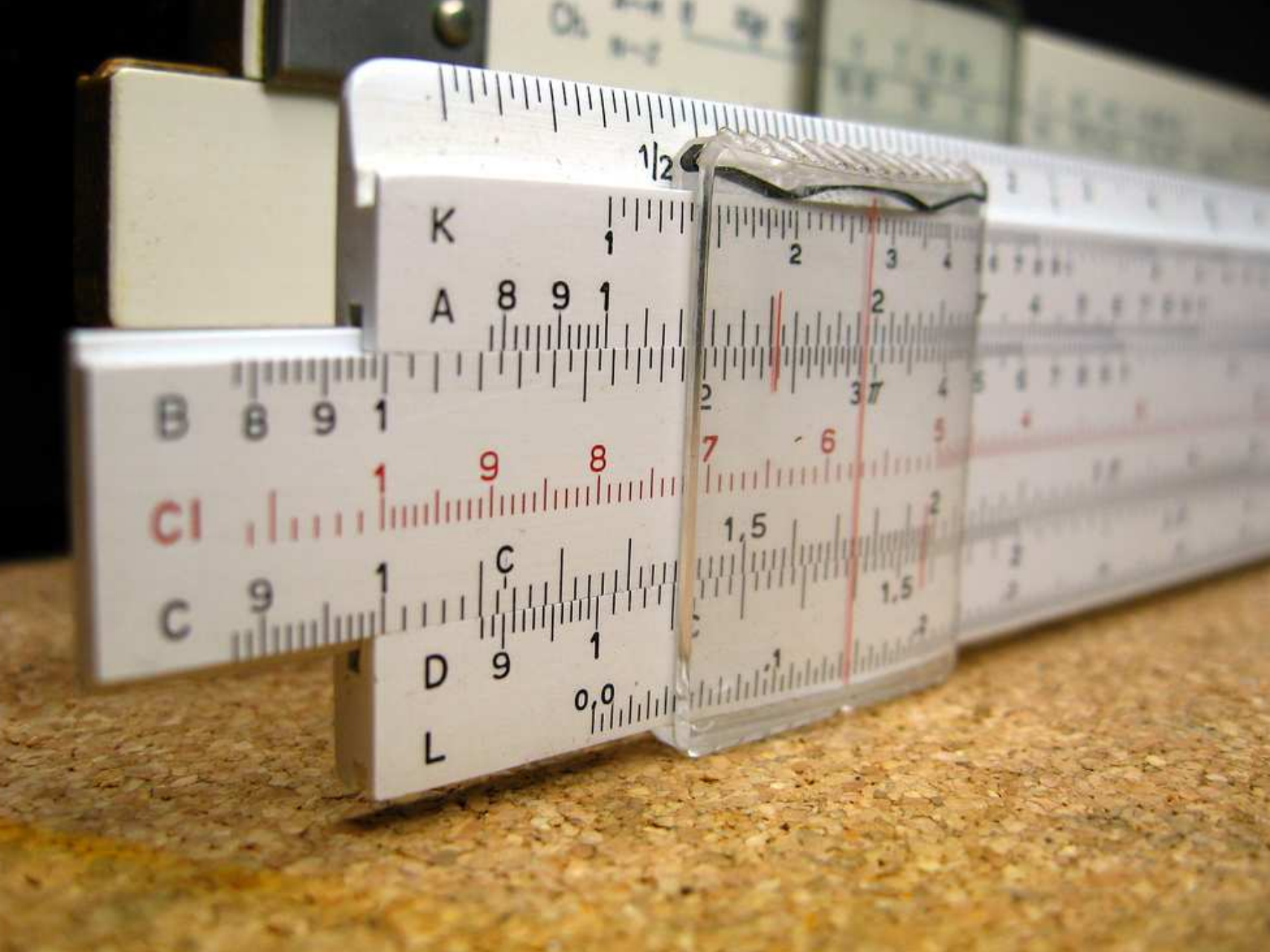
already know the answer

❷

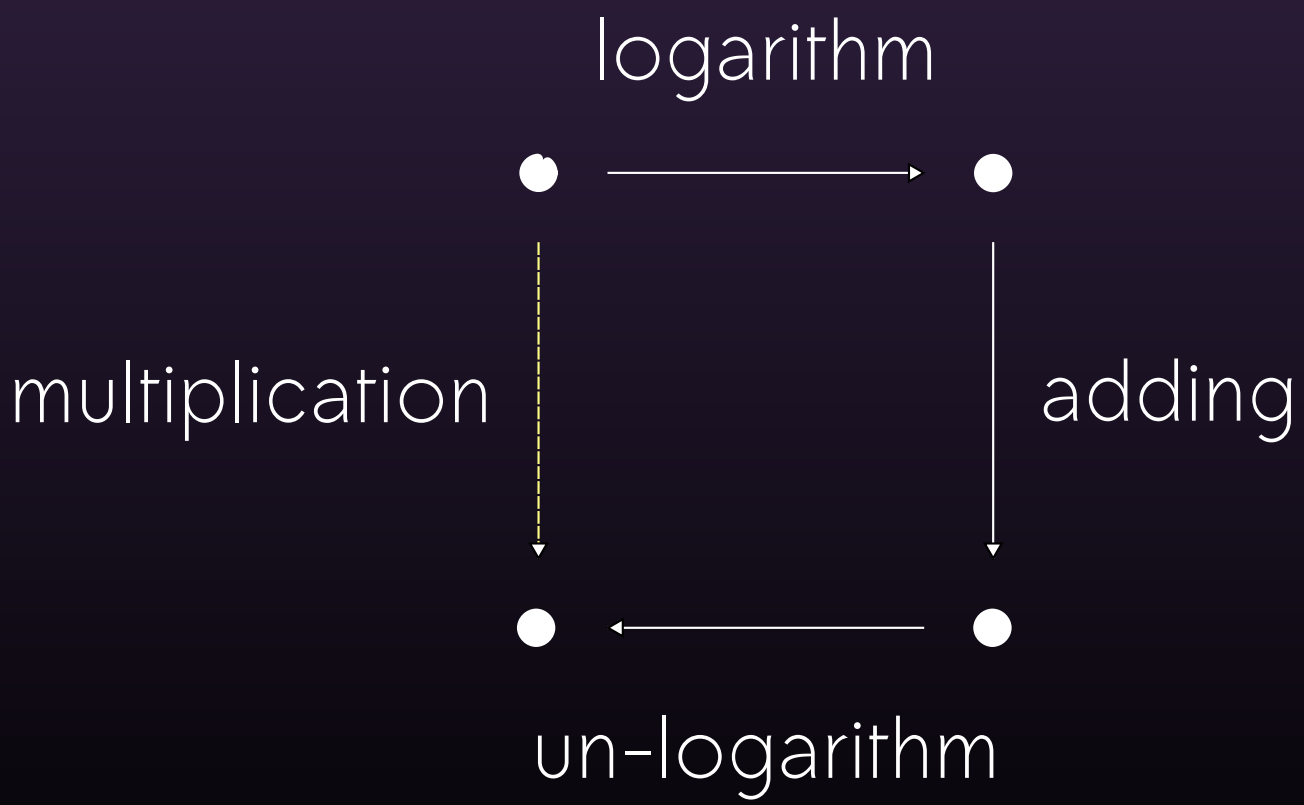
choose a simpler problem...

...and solve that instead

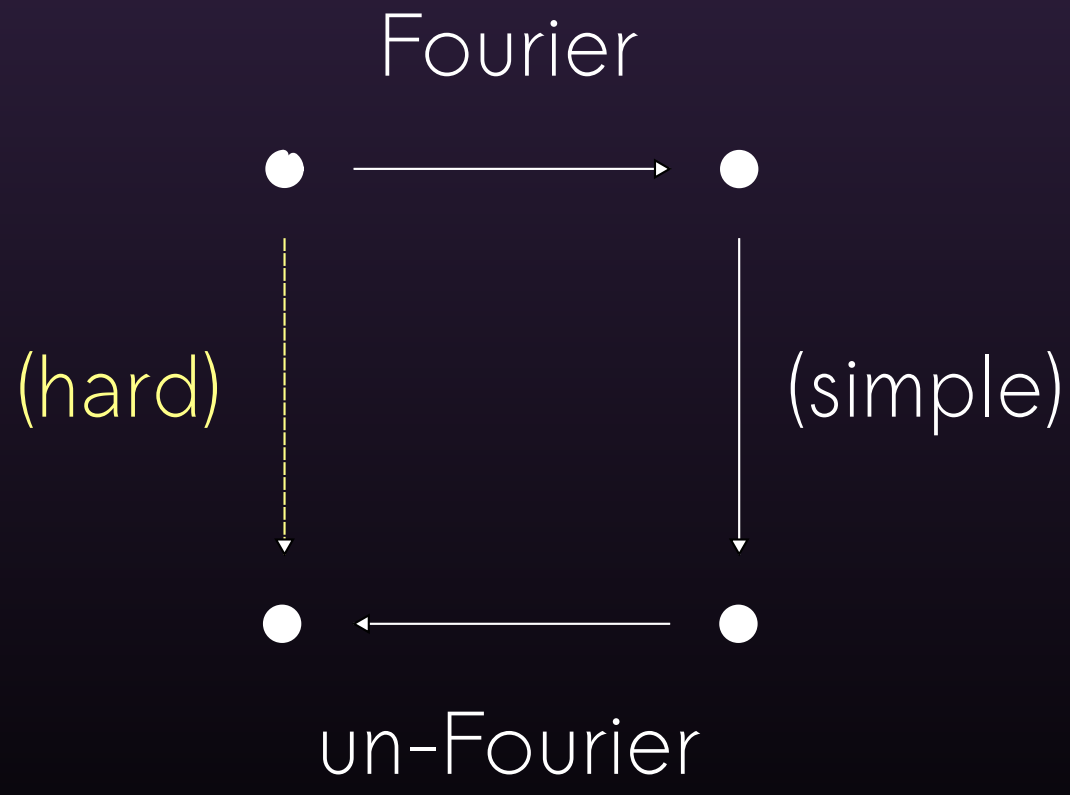
solving under a transform

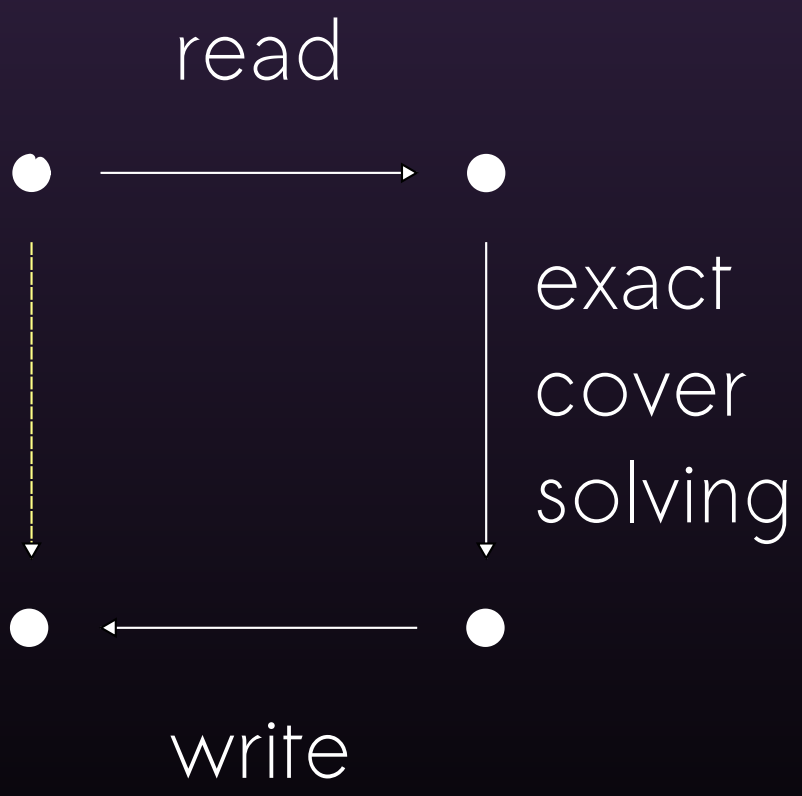

K
A 8 9 1
B 8 9 1
CI
C 9 1
D 9 1
L
1/2
1
2
3
4
1,5
0,0

logarithms


logarithm
multiplication
adding
un-logarithm

Fourier transformation


Fourier
(hard)
(simple)
un-Fourier


read
exact
cover
solving
write

why "athletic programming"?

blog post by Steve Yegge

"Rich programmer food"

7 problems

the compiler pattern


parsing
fiddling
writing


parsing
fiddling
writing

parsing
fiddling
writing

parsing
folding
writing

importing blog posts

-n and -p

```
perl -nE '$sum += $_;
          END { say $sum }'
```

```
perl -nE '$sum += $_;
          }{ say $sum'
```

}{

Yapsi

"Functional programming with Bananas, Lenses, Envelopes, and Barbed Wire"

by Erik Meijer,
Maarten M. Fokkinga,
Ross Paterson

parsing
folding
writing

the parsing step

yacc and lex

top-down

```
sentence = subject, verb, object
subject  = "I" | "we" | "they"
verb     = "can haz" | "enjoy"
object   = "cheezburger" | "Perl 6"
```

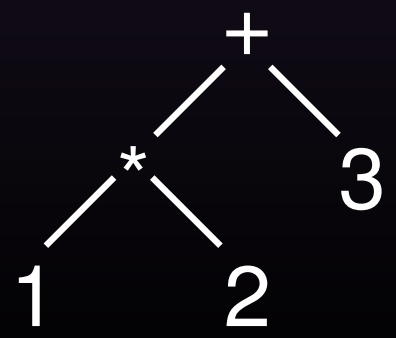
```
grammar Sentence {
  rule TOP {
    ^ <subject> <verb> <object> $
  }
  token subject { 'I' | 'we' | 'they' }
  token verb    { 'can haz' | 'enjoy' }
  token object  { 'cheezeburger'
                | 'Perl 6' }
}
```

bottom-up

Precedence

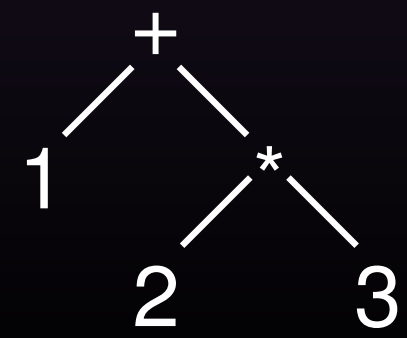
1 * 2 + 3

terms	ops
shift 1	
	shift infix:<*>
shift 2	
reduce △ * △	
	shift infix:<+>
shift 3	
reduce △ + △	



1 + 2 * 3

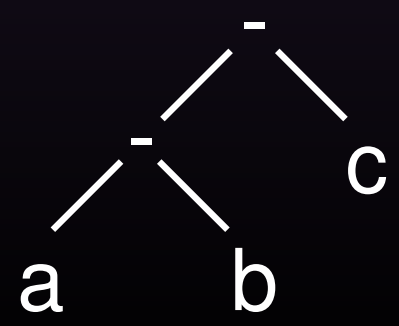
terms	ops
shift 1	
	shift infix:<+>
shift 2	
	shift infix:<*>
shift 3	
reduce △ * △	
reduce △ + △	



Associativity

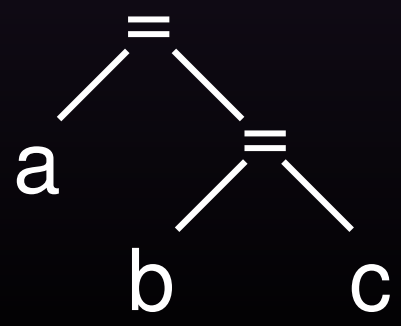
a - b - c

terms	ops
shift a	
	shift infix:<->
shift b	
reduce △ - △	
	shift infix:<->
shift c	
reduce △ - △	



a = b = c

terms	ops
shift a	
	shift infix:<=>
shift b	
	shift infix:<=>
shift c	
reduce △ = △	
reduce △ = △	



so... top-down or bottom-up?

both!

top-down for blocks/statements

bottom-up for expressions

```
for @a Z @b -> $a, $b {
    ...
}
```

backtracking

the Thompson engine

"Regular Expression Matching
Can Be Simple And Fast
(but is slow in Java, Perl,
PHP, Python, Ruby, ...)"

by Russ Cox

'You're doing it wrong!'

declarative subset

procedural superset

managing languages

language introspection divide


grammar
complexity
grammar
awareness

syntax complex

system not self-describing enough

Solution 1:
bring the
complexity
down

grammar
complexity

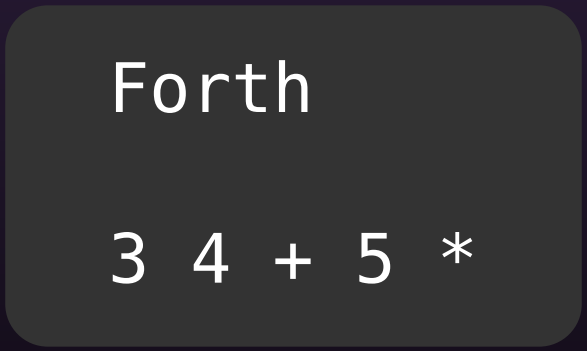
grammar
awareness

Lisp:

```
(* (+ 1 2)
   (+ 3 4))
```

Smalltalk:

```
3 + 4 * 5
```


Forth
3 4 + 5 *

APL:

```
(~R∈R∘.×R)/R←1↓⍳R
```

grammar
complexity

Solution 2:
bring the
awareness
up

grammar
awareness

going meta

talking about the language

in the language

Haskell, C++, Fortress, Perl 5

```
sub postfix:<!>(Int $n) {
    [*] 1..$n
}

say 5!;   # 120
```

precedence levels

Why Perl 6 is awesome

top-down and bottom-up
already built in

declarative and procedural

extensible metacircular parser
(can create sublanguages)

growing the language

DSL

domain-specific language

external DSLs

internal DSLs

weak!

internal DSLs:

should really be called

"nicely named subs/methods"

doing something right

do it all the time!

I propose new terms

growing the language inwards

growing the language outwards

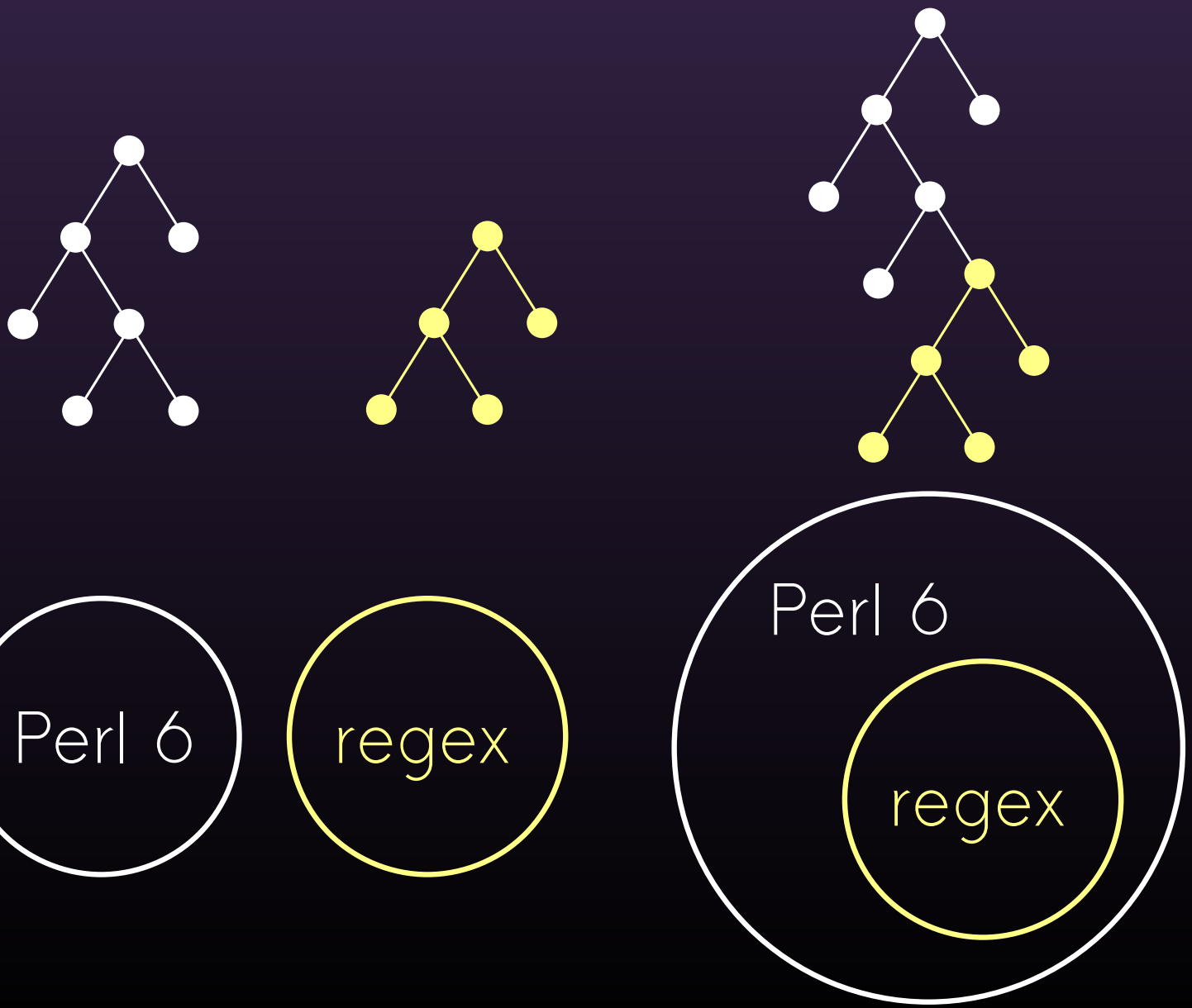
Growing the language inwards

"When programming a component,
the right computation model for
the component is the least expressive
model that results in a natural program."

— Concepts, Techniques,
and Models of
Computer Programming

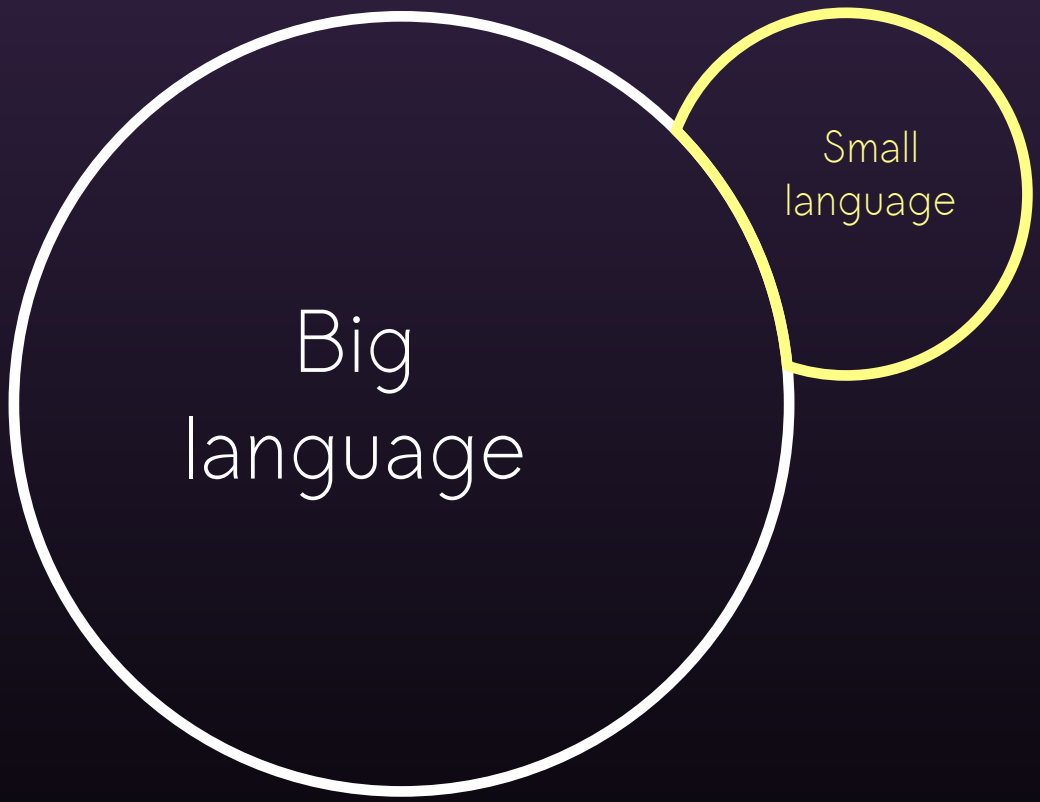
regexes

first-class citizens, not strings


Perl 6
regex
Perl 6
regex

SQL

```
my $query = Q:SQL:to'EOQ';
    SELECT * FROM mytable
      WHERE _id = $id
EOQ
```



Growing the language outwards

Jeff Moser asked Alan Kay...

elegant code ☺

too much "goo" ☹

"You 'avoid the goo'
by avoiding the goo."

— Alan Kay

Wasabi

compiles to VBScript, PHP4, PHP5

introduces closures

active records

anonymous functions

embedded SQL

CoffeeScript

```
# Array comprehensions:
cubes = (math.cube n for n in list)
```

```
cubes = (function() {
  var _i, _len, _results;
  _results = [];
  for (_i = 0, _len = list.length; _i < _len; _i++) {
    n = list[_i];
    _results.push(math.cube(n));
  }
  return _results;
})();
```

STD.pm6

```
token decint {
    \d+ [ _ \d+ ]*
}
```

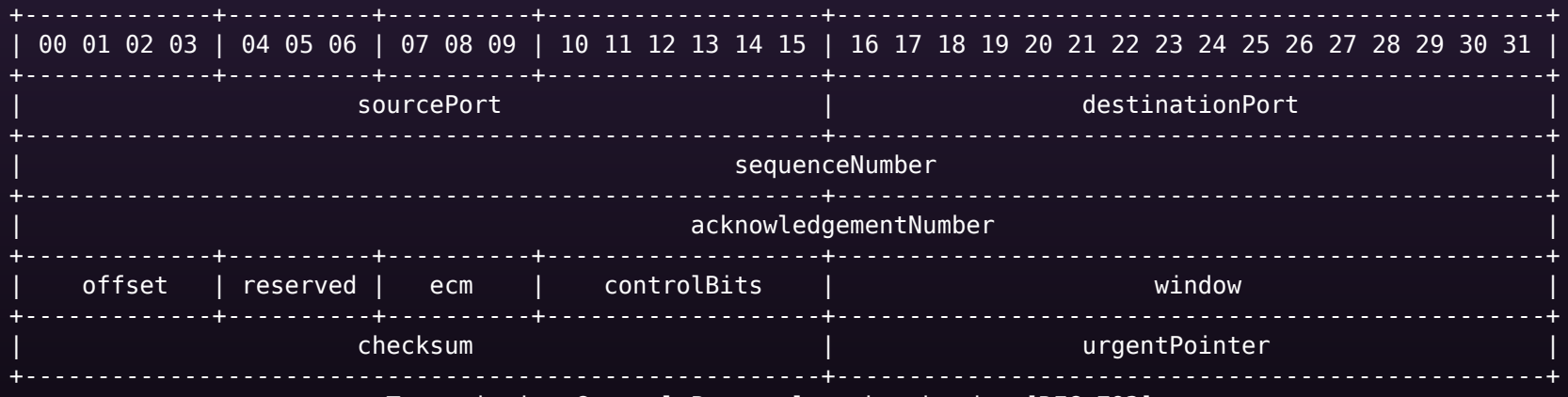
```
## token decint {
sub decint__PEEK { $_[0]->_AUTOLEXpeek('decint', $retree) }
sub decint {
    no warnings 'recursion';
    my $self = shift;


    local $::CTX = $self->callm() if $::DEBUG & DEBUG::trace_call;

    my $C = $self->cursor_xact("RULE decint");
    my $xact = $C->xact;
    my $S = $C->{'_pos'};
    $self->_MATCHIFYr($S, "decint", do {
        if (my ($C) = ($C->_PATTERN(qr/\G\d++/))) {
            $C->_STARr(sub {
                    my $C=shift;
                    if (my ($C) = ($C->_BRACKETr(sub {
                                my $C=shift;
                                $C->_PATTERN(qr/\G_\d++/)
                            }))) { ($C) } else { () }
                })
        } else { () }

    });
}
;
```

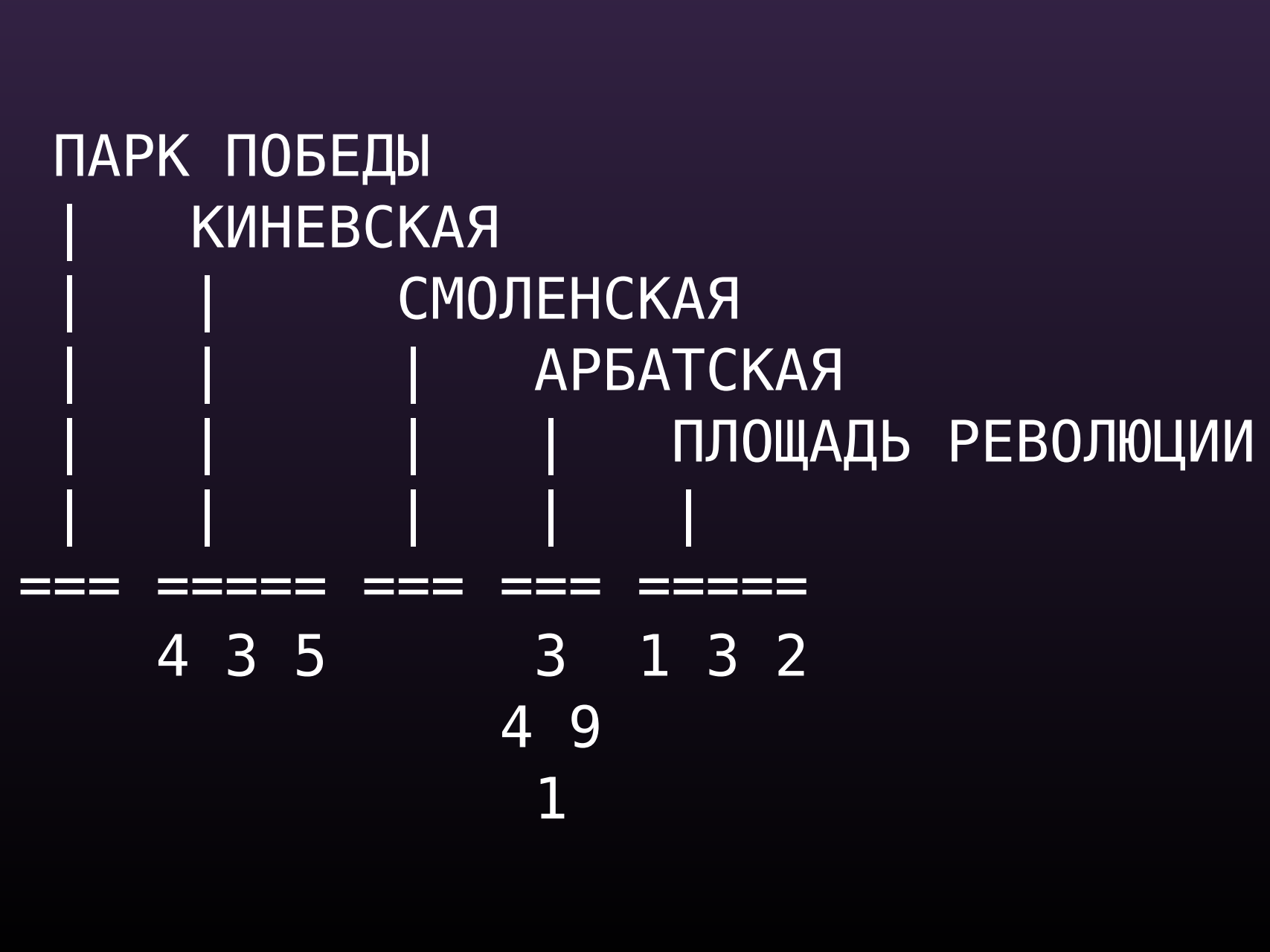
OMeta



tcp -- Transmission Control Protocol packet header [RFC 793]

TCP/IP stack: < 200 lines!


СТРОГИНО
КРЫЛАТСКОЕ
МОЛОДЕЖНАЯ
КУНЦЕВСКАЯ
СЛАВЯНСКИЙ БУЛЬВАР
ПАРК ПОБЕДЫ
КИЕВСКАЯ
СМОЛЕНСКАЯ
АРБАТСКАЯ
ПЛОЩАДЬ РЕВОЛЮЦИИ
КУРСКАЯ
БАУМАНСКАЯ
ЭЛЕКТРОЗАВОДСКАЯ
СЕМЕНОВСКАЯ
4 3
4 3 5
4 3 1 9
1 3 2
5 3 10
1- Сокольническая линия
2 - Замоскворецкая линия
3- Арбатско-Покровская линия
4 - Филев
5 - Кольце
9 - Серпух
10 - Любл


ПАРК ПОБЕДЫ
КИНЕВСКАЯ
СМОЛЕНСКАЯ
АРБАТСКАЯ
ПЛОЩАДЬ РЕВОЛЮЦИИ
4 3 5
3
1 3 2
4 9
1

conclusion

languages rock

shift of view ⇢ easier problem

little languages rock

write code on the right level

write code that is aware

stay aware

Attribution

Diwa N-601-1 slide rule (cc-by)

http://www.flickr.com/photos/bikeman04/4004086068/

Metro de Moscú (cc-by-nc)

http://www.flickr.com/photos/madelman/2949942186/

спасибо

есть вопросы?