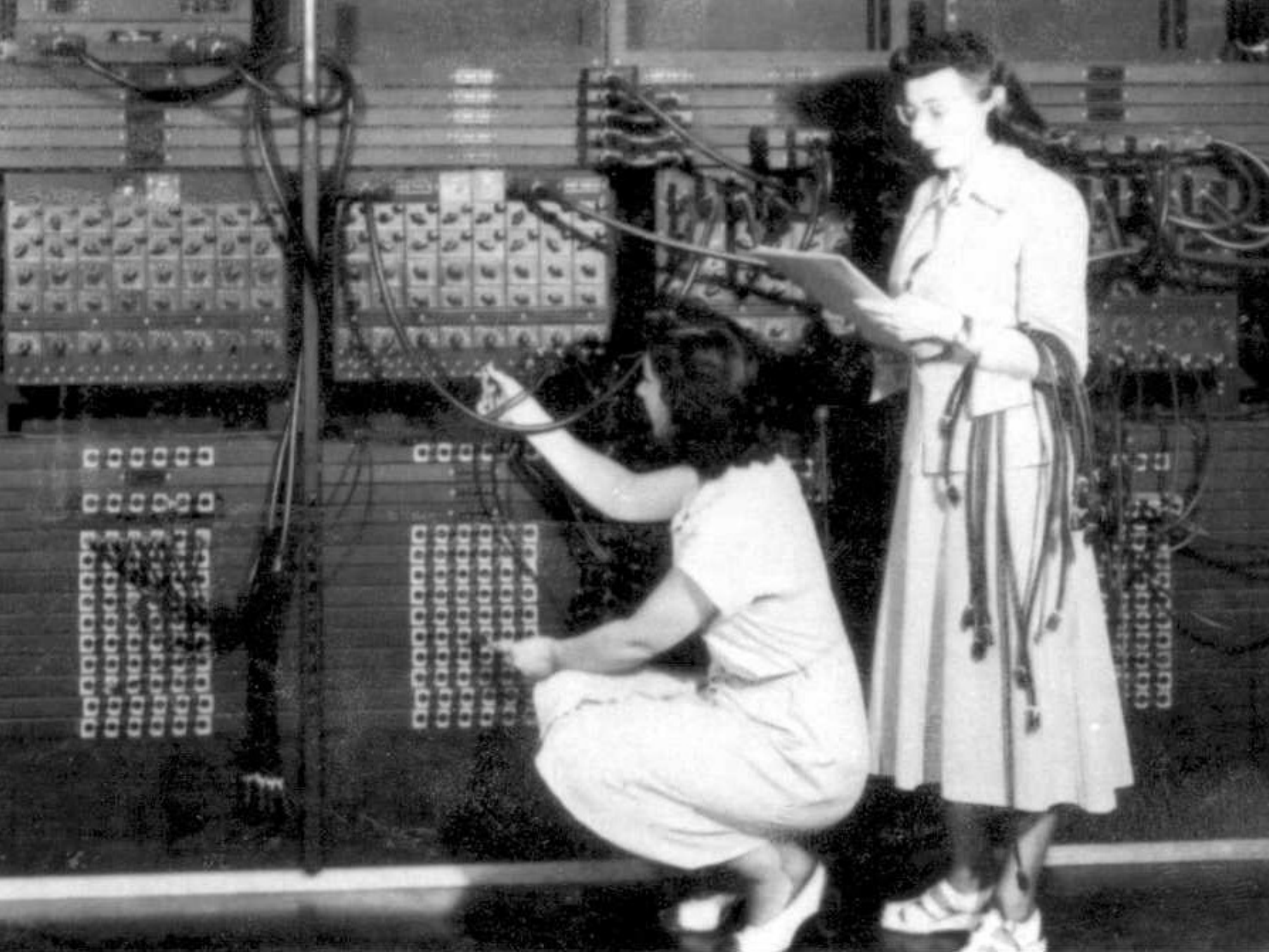
What I want from a coding tool

What I want from a ~~coding~~ tool

What I want from a ~~coding~~ tool programming

What is programming?

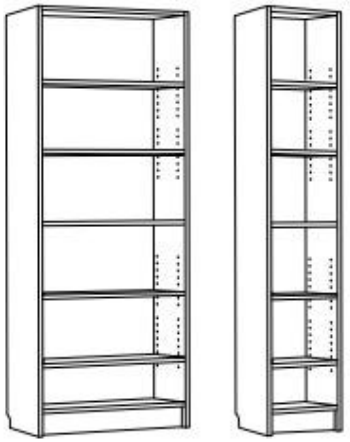


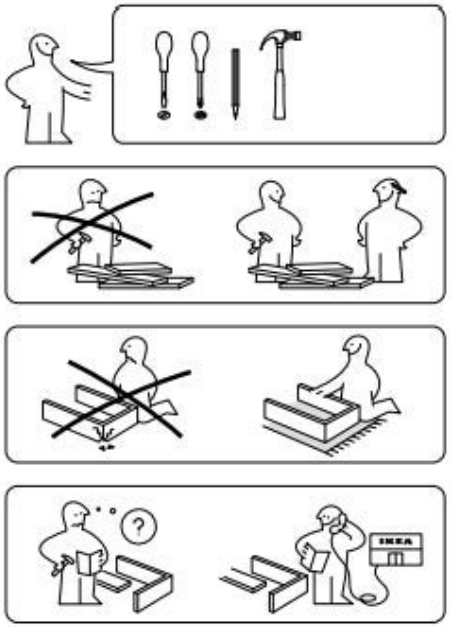

INTE
PROT
MEMR
INP
M1
OUT
HLTA
STACK
WO
INT
STATUS
WAIT
HLDA
A15
A14
A13
A12
A11
A10
A9
A8
A7
A6
A5
A4
A3
A2
A1
A0
D7
D6
D5
D4
D3
D2
D1
D0
SENSE SW
OFF
ON
STOP
RUN
SINGLE STEP
EXAMINE
EXAMINE NEXT
DEPOSIT
DEPOSIT NEXT
RESET
CLR
PROTECT
UNPROTECT
AUX
AUX
MITS
ALTAIR 8800 COMPUTER
© WWW.ALTAIR32.COM

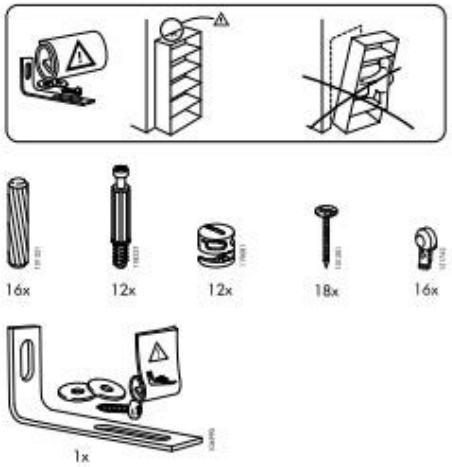

TELETYPE
SHIFT
SHIFT
STOP

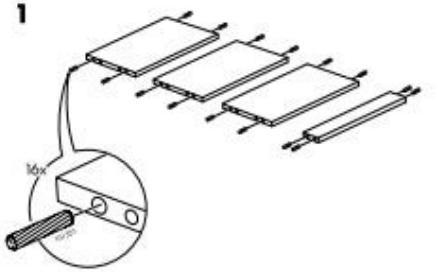
Is programming *making computers do things?*

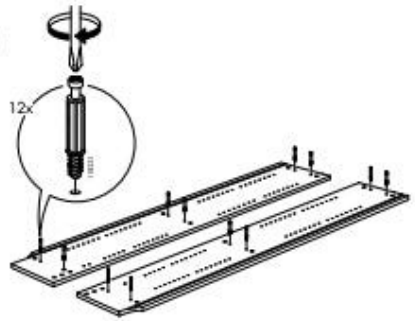
BILLY

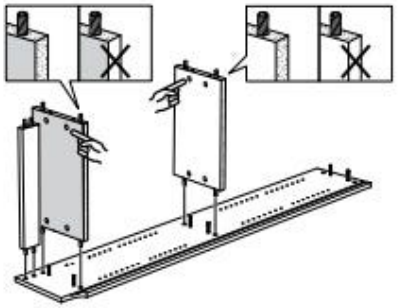


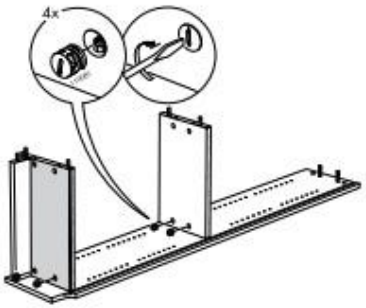


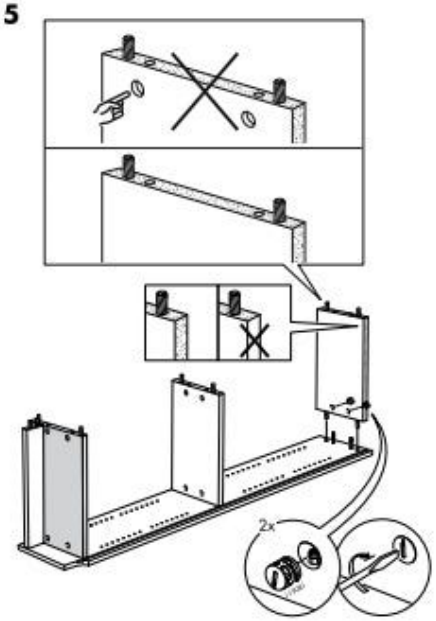


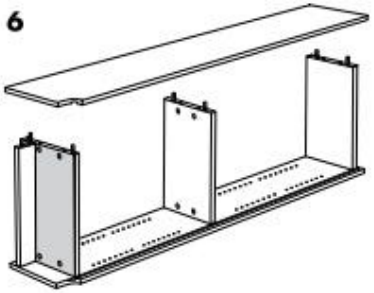


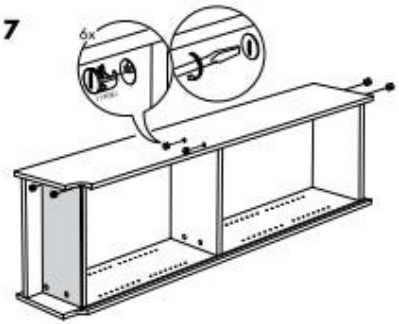


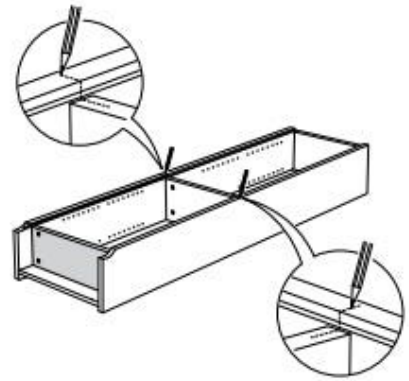


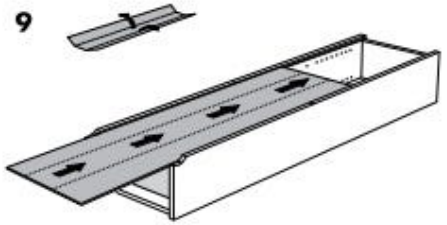




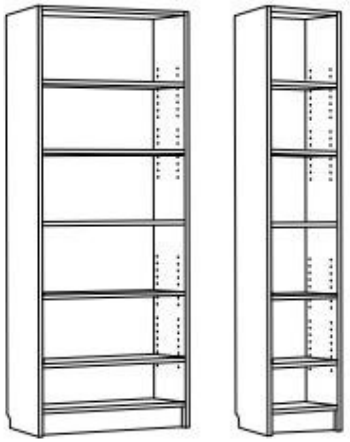




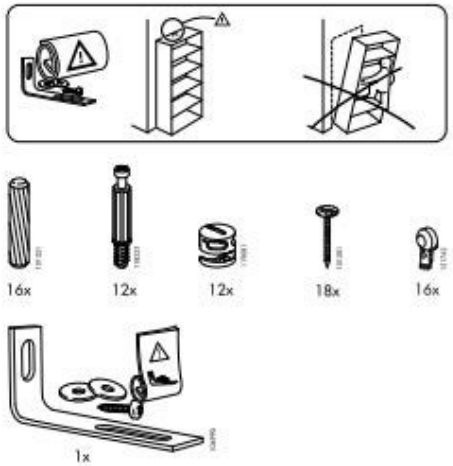


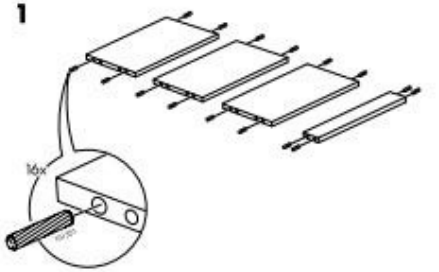


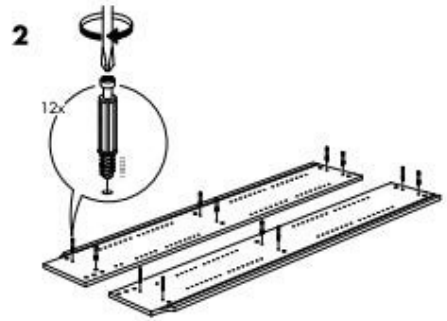
BILLY

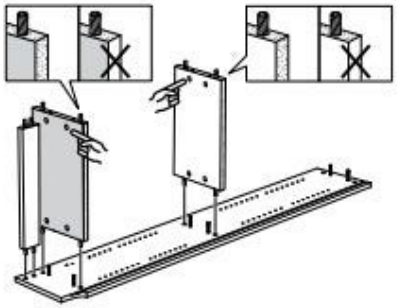


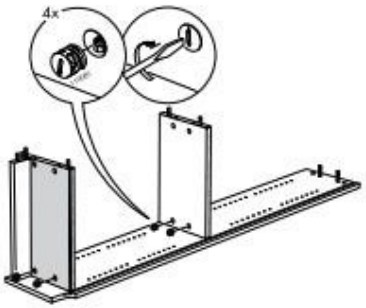
IKEA
Design and Quality
IKEA of Sweden

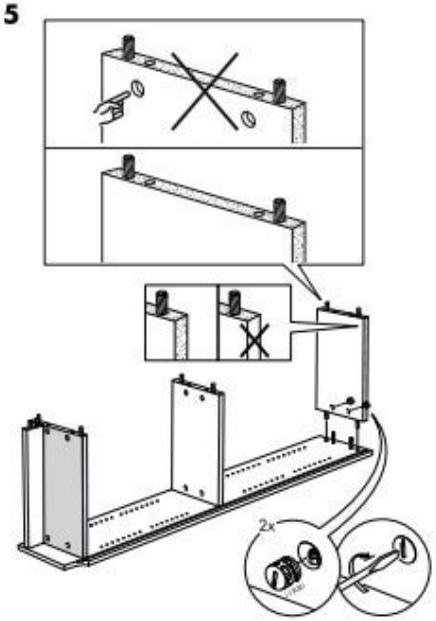


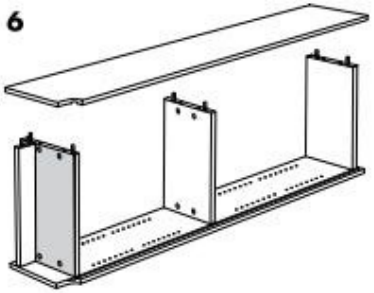


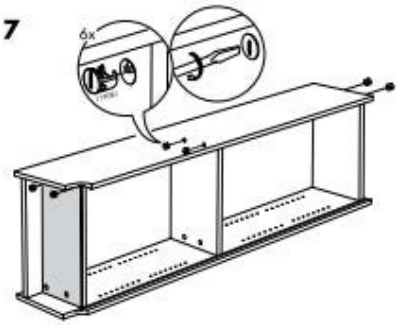


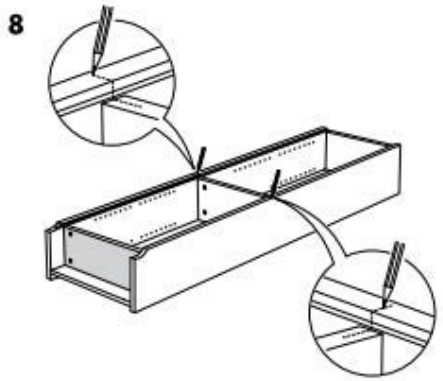


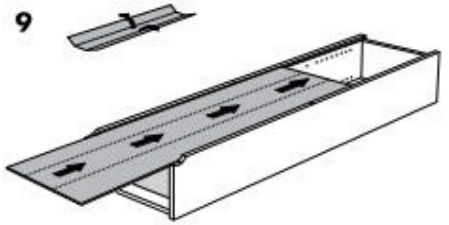






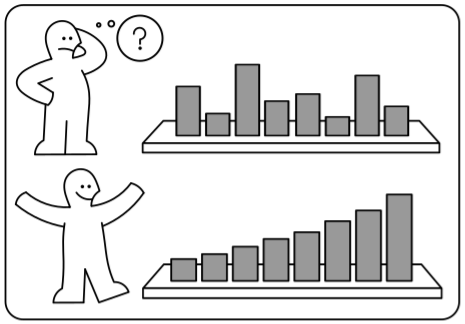


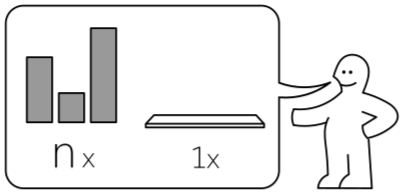




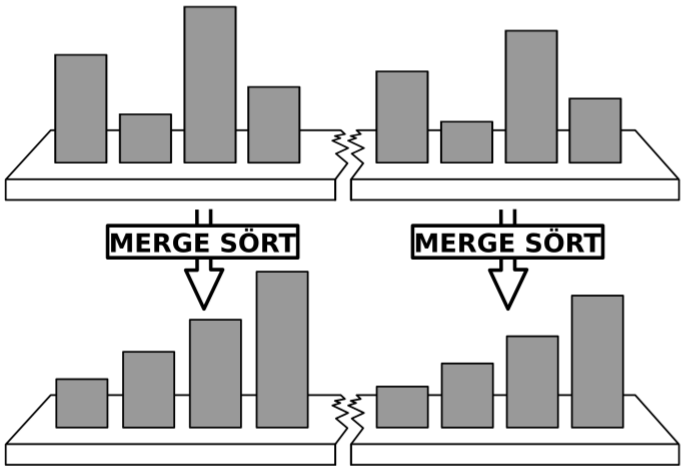
MERGE SÖRT

idea-instructions.com/merge-sort/
v1.2, CC by-nc-sa 4.0

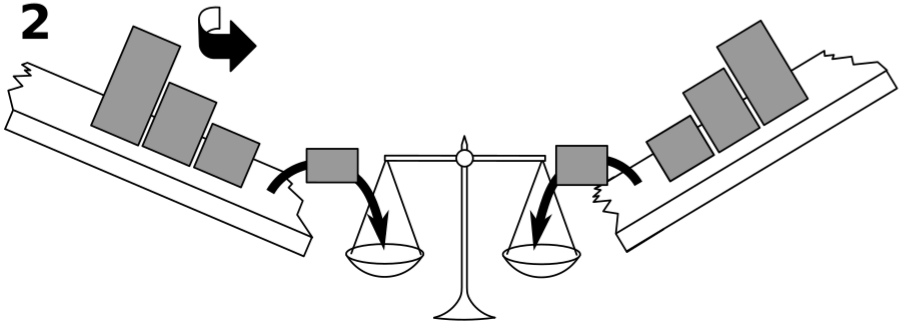




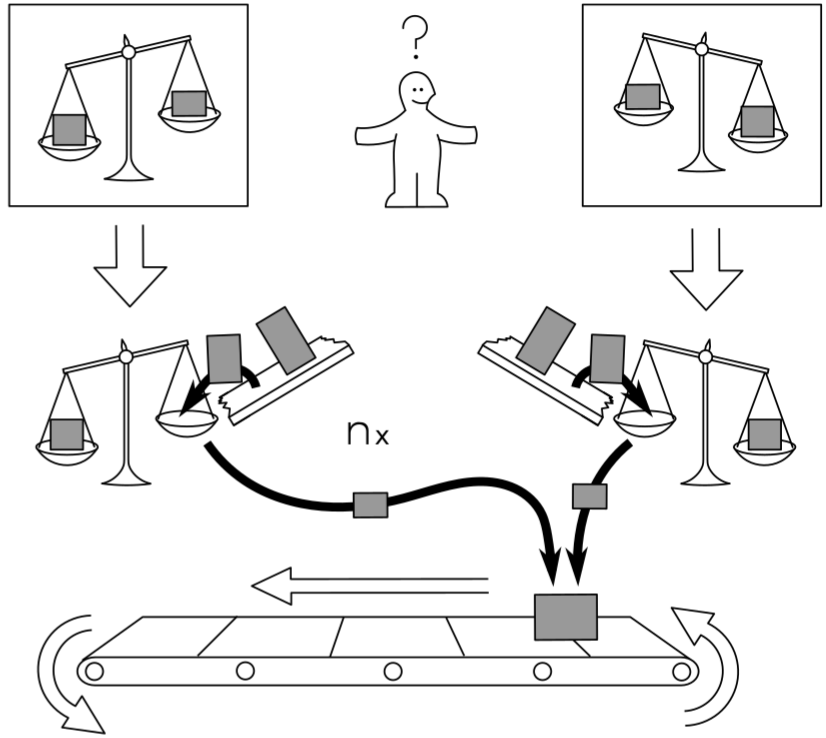
1



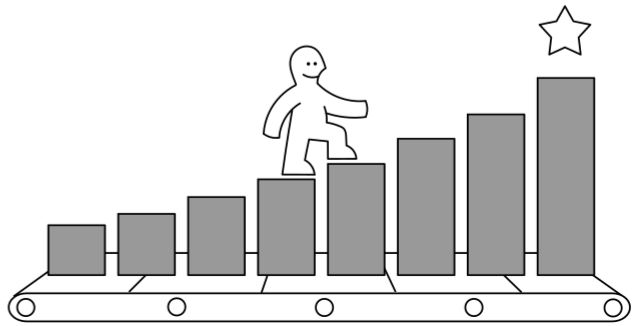
2



3



4



Programming is a complex intellectual activity that involves modeling

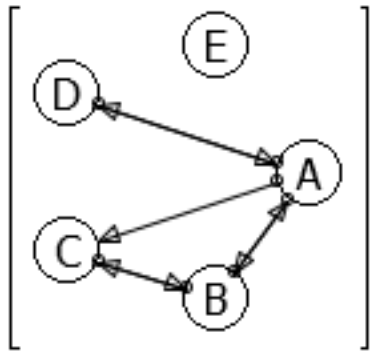
[Scheme'22] Programming is (should be) fun!

GRASP LIMB

```
x
[ f x ]
[ f [ f x ] ]
[ [ ] ]
```

```
[ define [ ! n ]
  [ [ if [ = n 0 ]
      1
      [ * n [ ! [ - n 1 ] ] ] ] ] ]
```

```
[ e.g. [ ! 5 ] ===> 120 ]
```



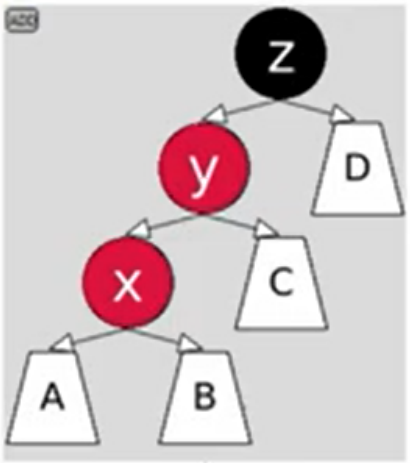
I want a single piece of glass I can use to read email on the toilet

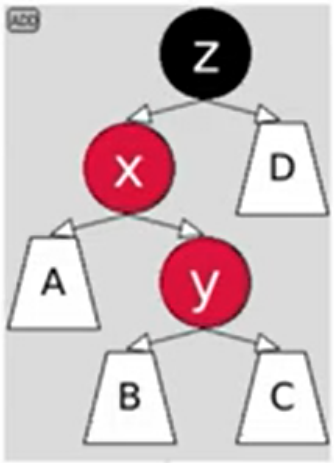
– Steve Jobs

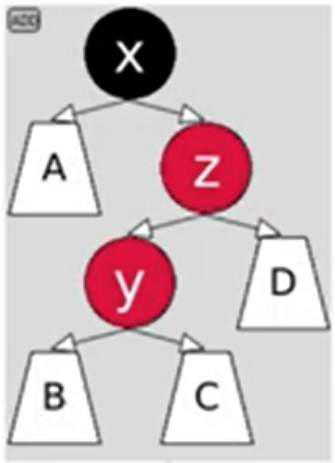

Fn
Control
Alt
Option
Command
Shift
Enter
Home
PgUp
PgDn
End

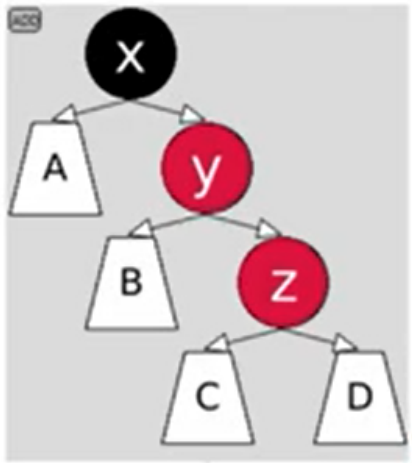
GRASP/Java demo

```
(define (balance t)
  (match t
    [(or
```

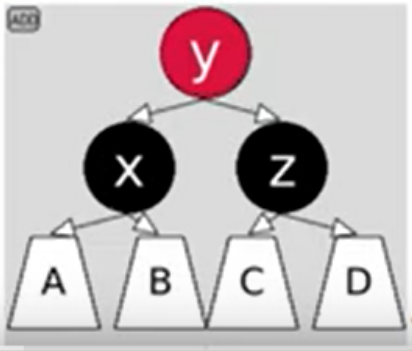









```
)
; =>
```




```
]
[else t]))
```

1:02 / 12:24

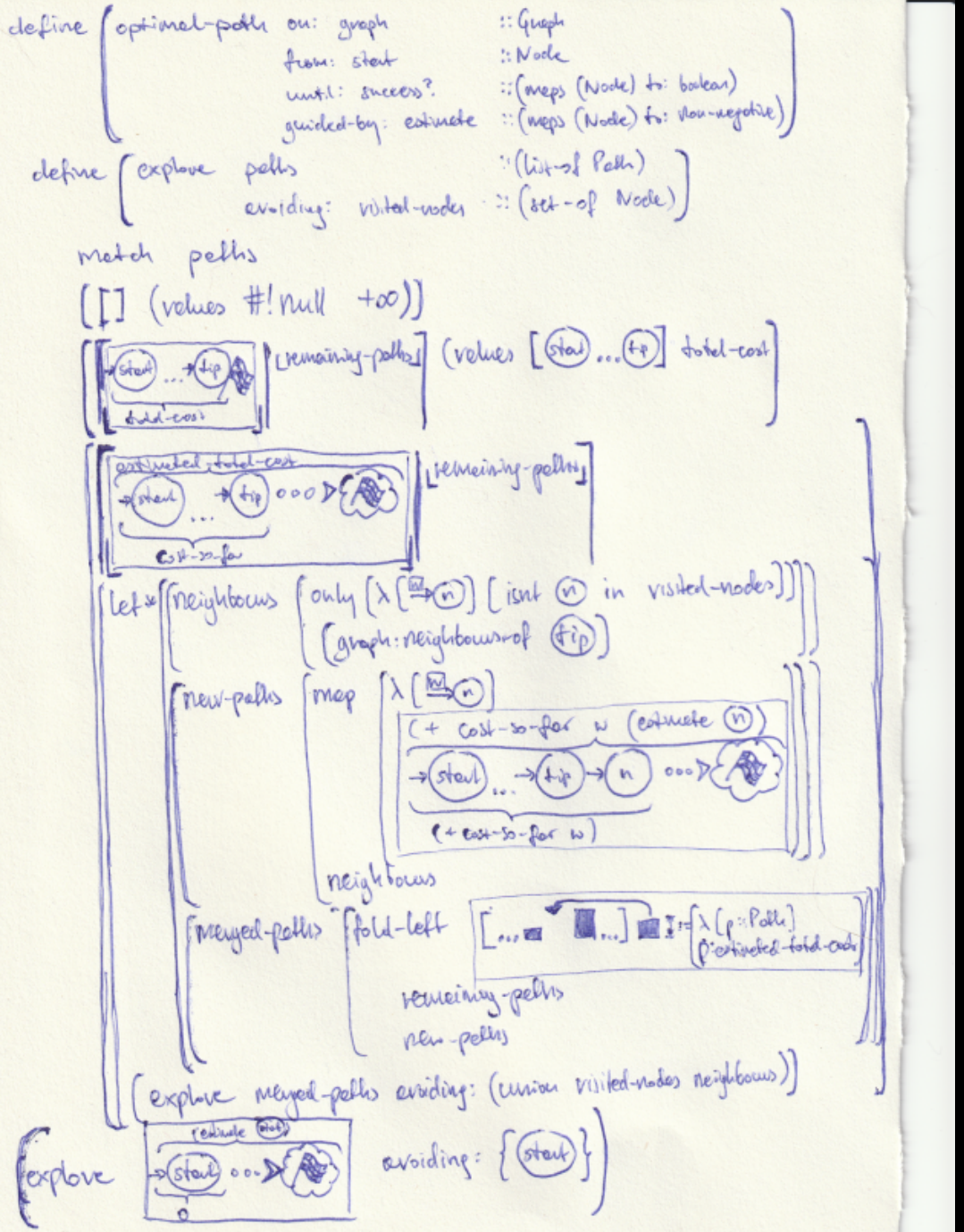
Adding Interactive Visual Syntax to Textual Code

Leif Andersen
47 subskrybentów

Subskrybuj

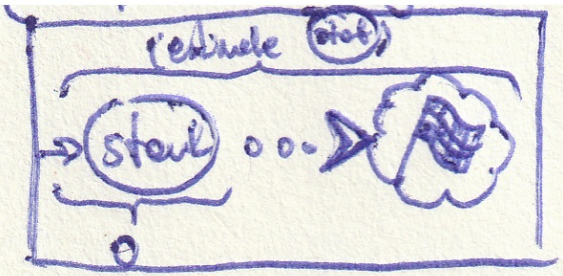
83 | Udostępnij | Klip | Zapisz

GRASP/Kawa demo (stepper)


define
optimal-path on: graph ::Graph
from: start ::Node
until: success? ::(maps (Node) to: boolean)
guided-by: estimate ::(maps (Node) to: non-negative)
define
explore paths ::(list-of Path)
avoiding: visited-nodes ::(set-of Node)
match paths
[] (values #!null +∞)
start
tip
total-cost
remaining-paths
(values [start ... tip] total-cost
estimated-total-cost
cost-so-far
let*
neighbours
only [λ [n] [isnt n in visited-nodes]]
(graph:neighbours-of tip)
new-paths
map
λ [n]
(+ cost-so-far w (estimate n))
(+ cost-so-far w)
merged-paths
fold-left
ρ::Path
ρ.estimated-total-cost
explore merged-paths avoiding: (union visited-nodes neighbours)
explore
(estimate start)
avoiding: {start}

```
define (optimal-path on: graph          :: Graph
                    from: start         :: Node
                    until: success?     :: (maps (Node) to: boolean)
                    guided-by: estimate :: (maps (Node) to: non-negative))

define (explore  paths                  :: (list-of Path)
                 avoiding: visited-nodes :: (set-of Node))
```

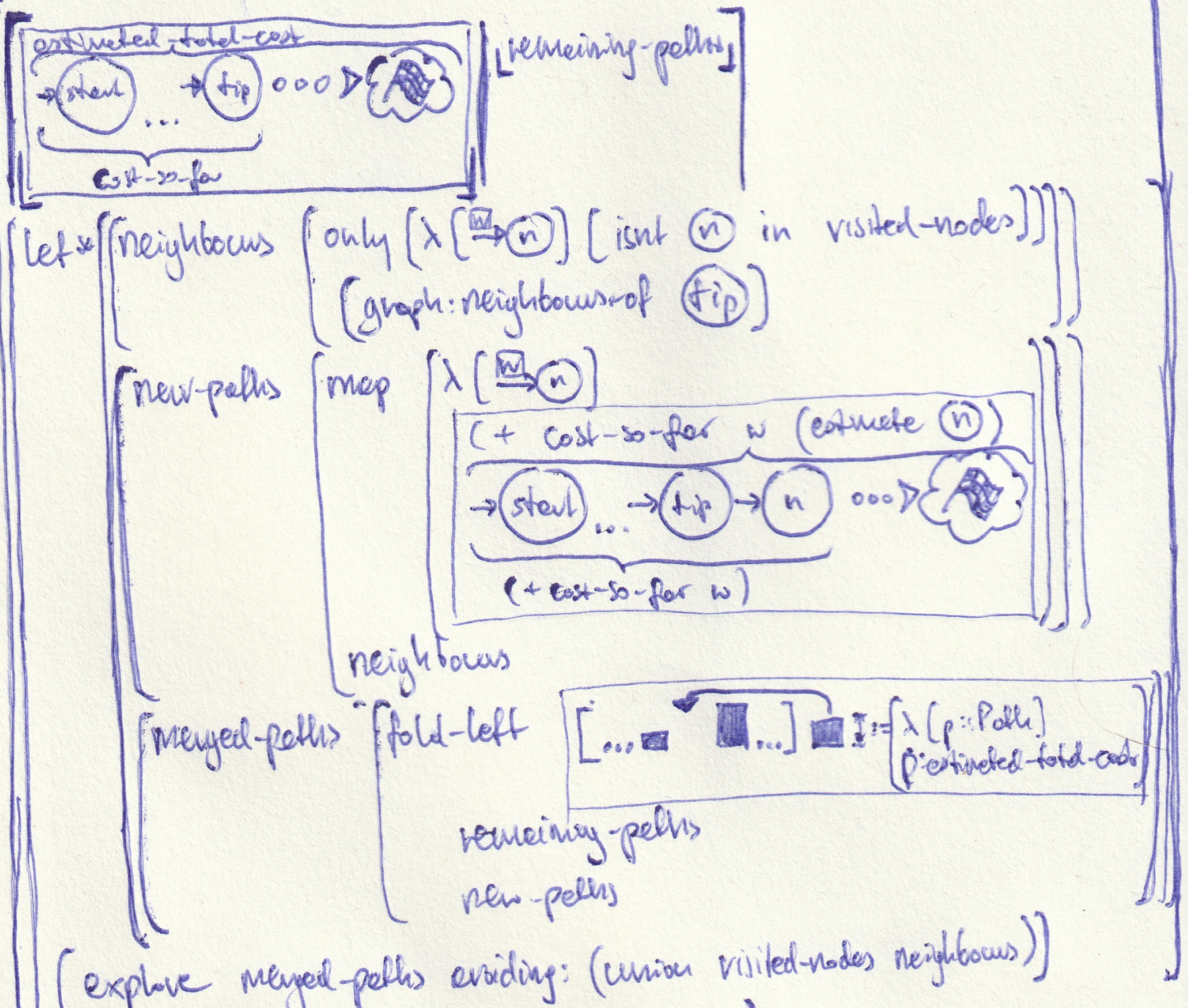
(explore 

avoiding: {start})

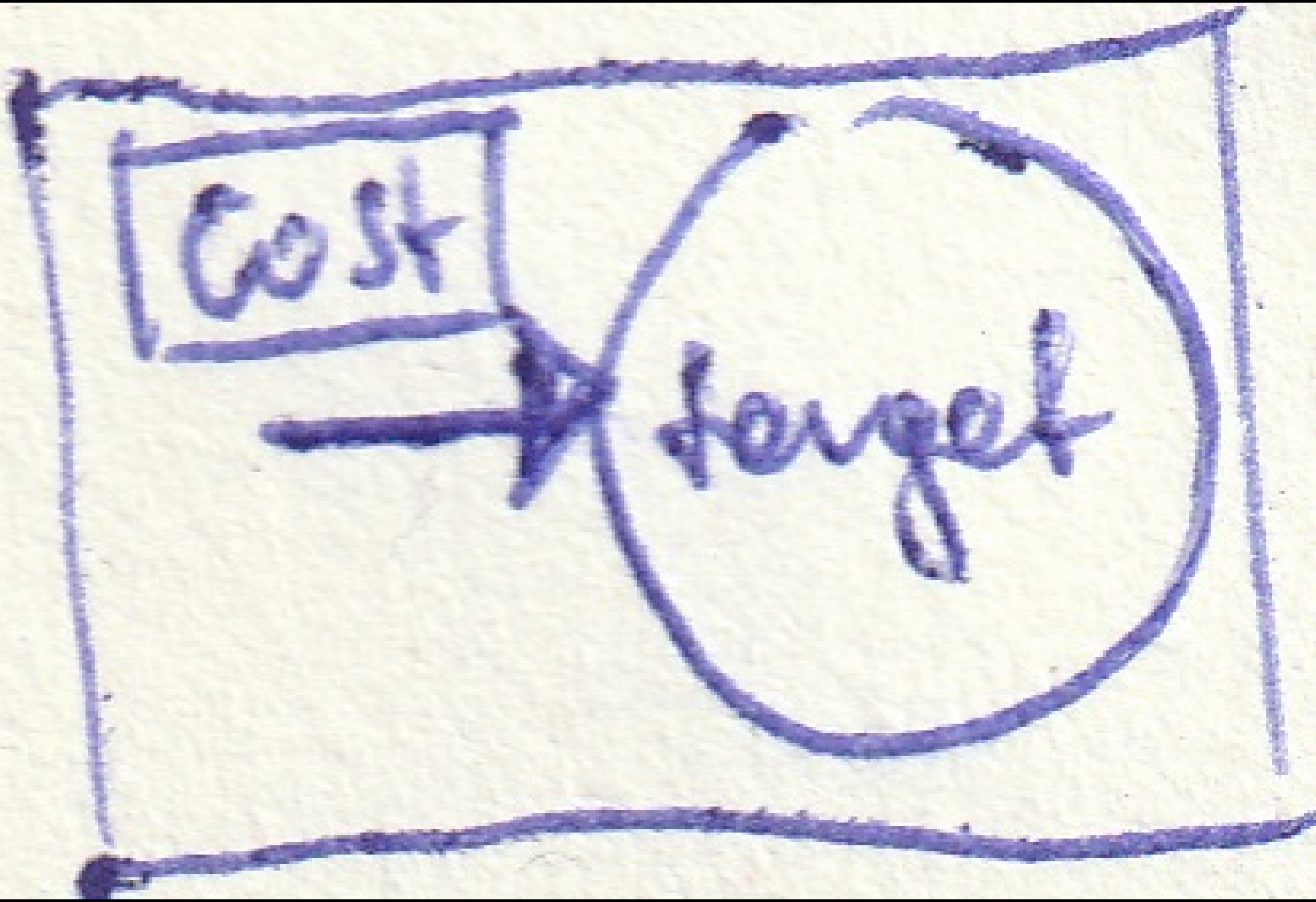
match paths

[[] (values #!null +∞)]

[[[start ... tip] total-cost] remaining-paths] (values [start ... tip] total-cost)

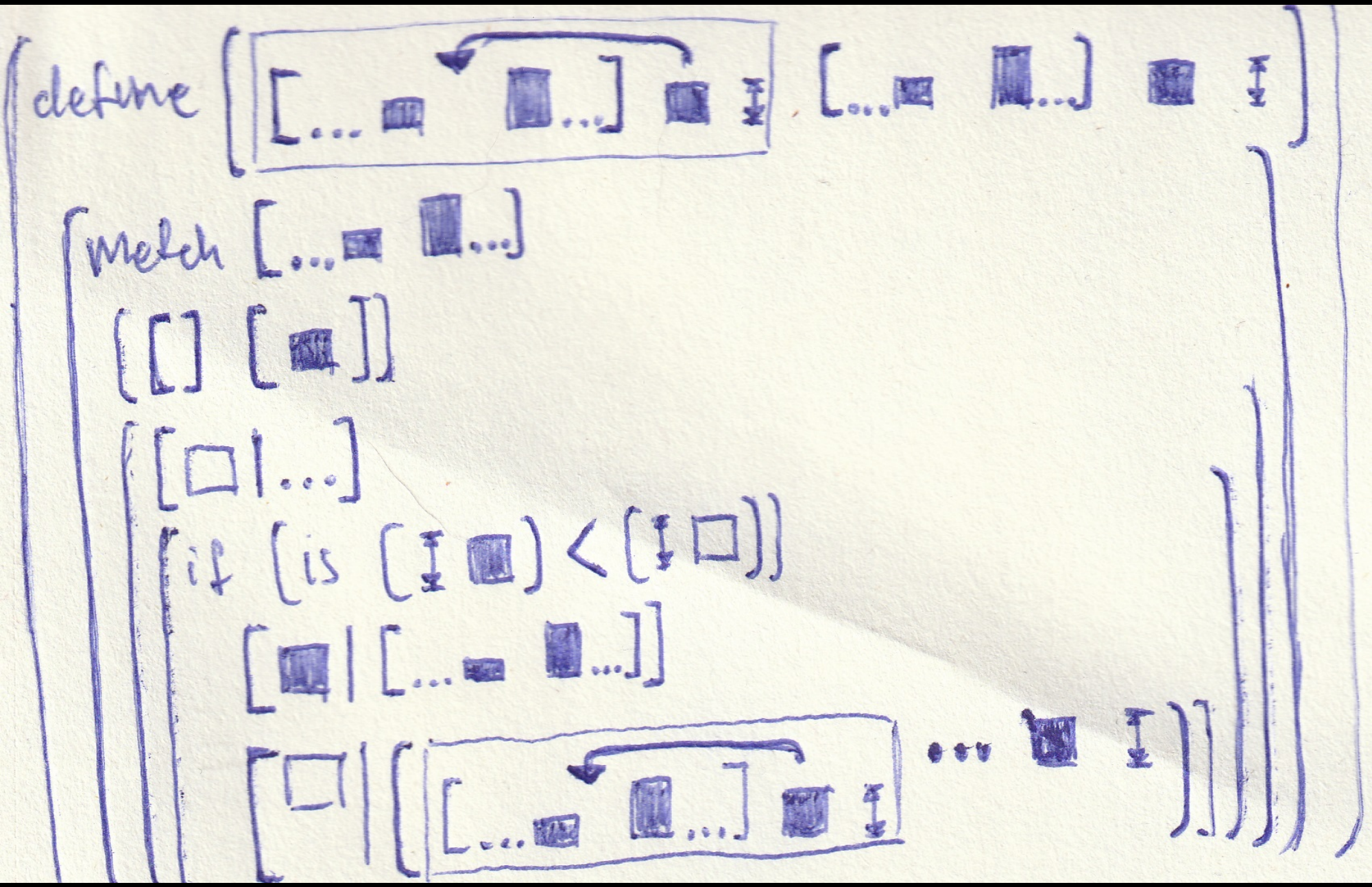

estimated-total-cost
start
tip
cost-so-far
remaining-paths
let*
neighbours
only
(λ [w → n] [isnt n in visited-nodes])
(graph:neighbours-of tip)
new-paths
map
(λ [w → n]
(+ cost-so-far w (estimate n)
start
tip
n
(+ cost-so-far w)
neighbours
merged-paths
fold-left
(λ [p :: Path]
p:estimated-total-cost
remaining-paths
new-paths
(explore merged-paths avoiding: (union visited-nodes neighbours))

```
(define-type (Step cost: non-negative
                   target : Node))
```


cost
forget

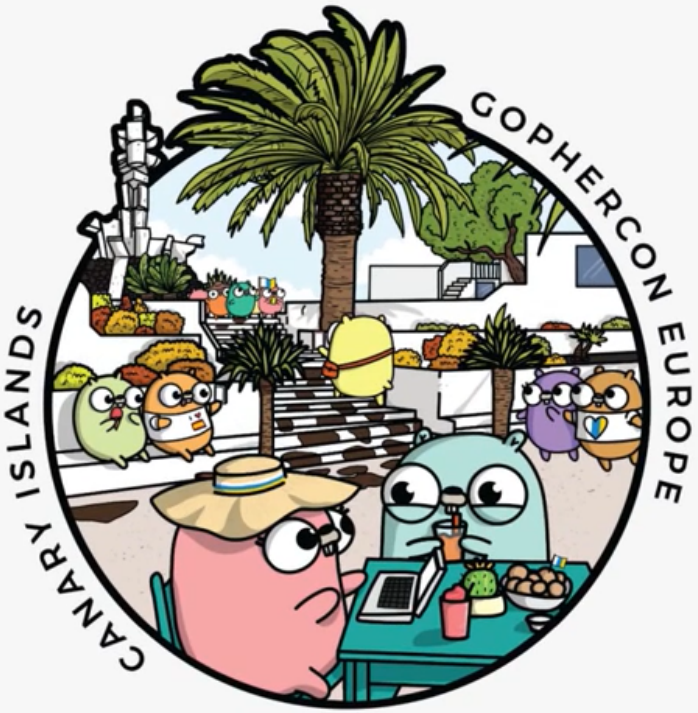
```
(define-type (Path
              steps : (List Step)
              cost-so-far : Non-negative
              estimated-total-cost : Non-negative))
```


step
cost-so-far



GRASP/Kawa demo (zoom out)


IS THIS
settings.gradle
assets
build
fonts
ios
lib
main.dart
test
ARCHITECTURE DIAGRAM?



RETHINKING VISUAL PROGRAMMING

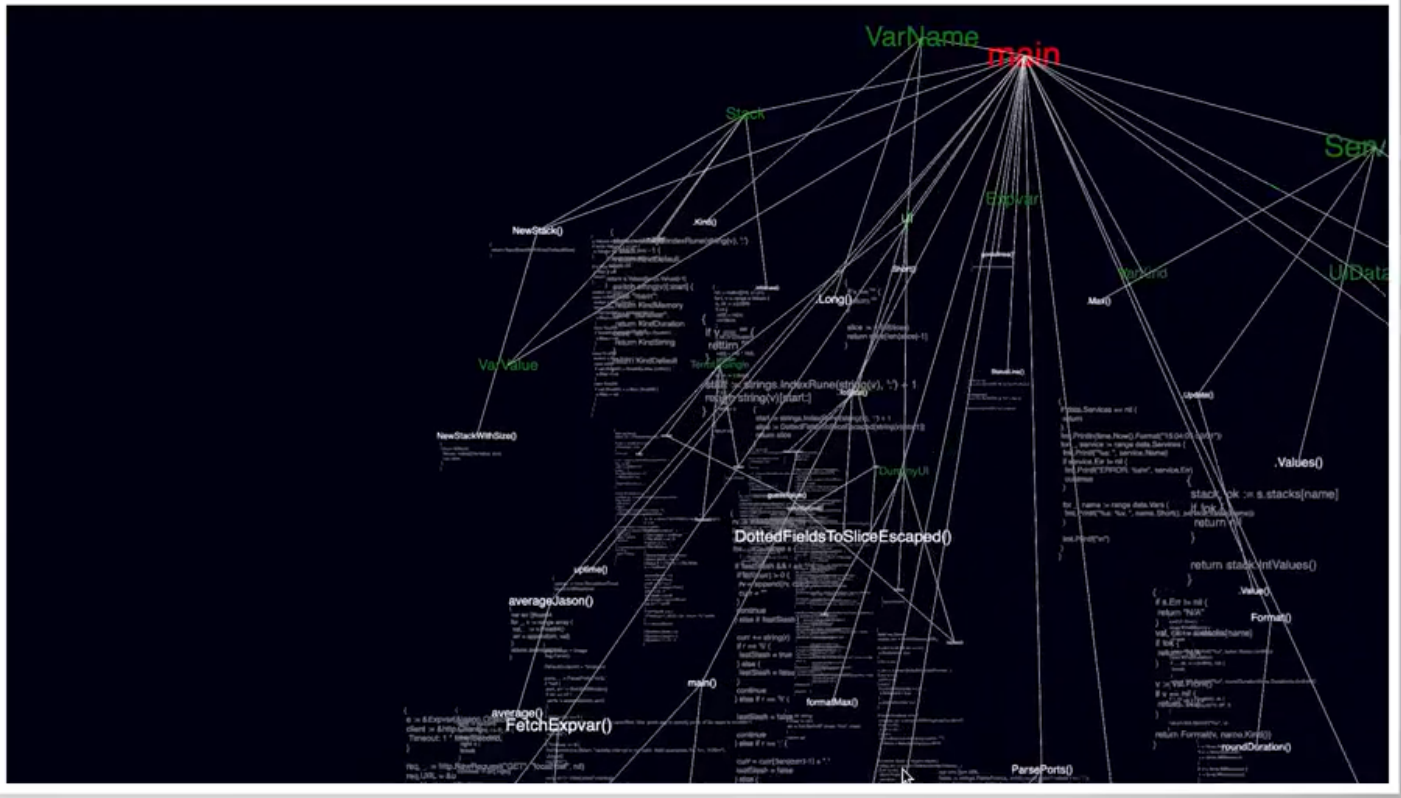
Ivan Daniluk
@idanyliuk

0:05 / 30:53 • Introduction >

GopherCon Europe 2019: Ivan Daniluk - Rethinking Visual Programming

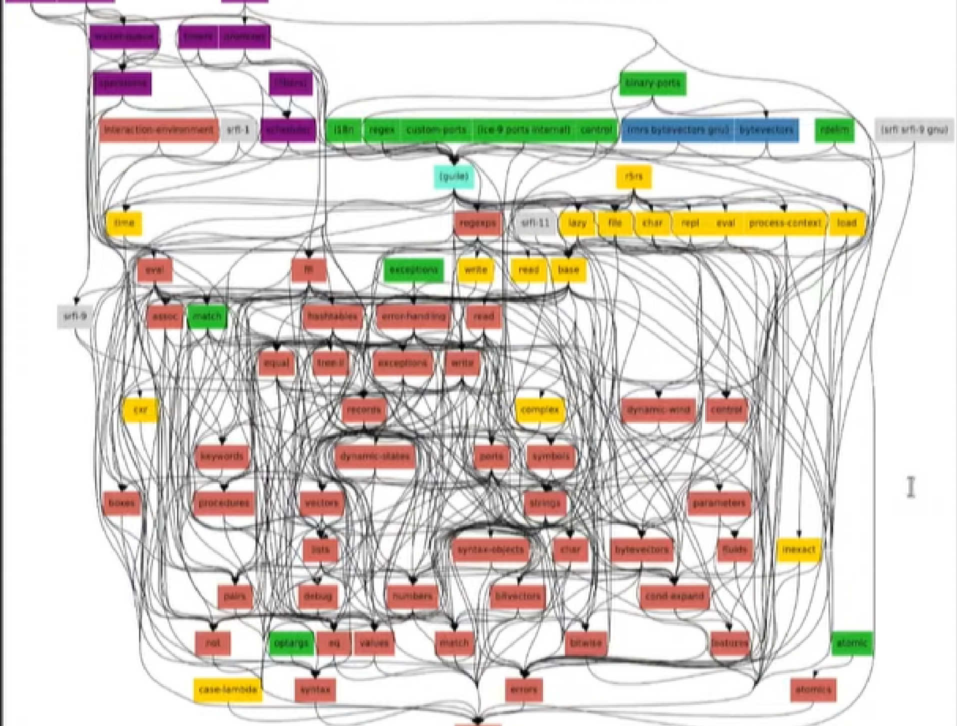
0:29 / 30:53 • Textual Programming >

GopherCon Europe 2019: Ivan Daniluk - Rethinking Visual Programming



GopherCon Europe 2019: Ivan Daniluk - Rethinking Visual Programming


Middle Earth
Forochel
Forodwaith
Angmar
Arnor
Eriador
Lindon
Forlindon
Gulf of Lune
Shire
Harlindon
Minhiriath
Eregion
Hollin
Lorien
Mirkwood
Rhovanion
Anduin
Dunland
Fangorn
Enedwaith
Rohan
the Brown Lands
Dead Marshes
Udun
Mordor
Gondor
Anfalas
Belfalas
Lebennin
Bay of Belfalas
Khand



binary-ports
i18n
regex
custom-ports
(ice-9 ports internal)
control
(rnrs bytevectors gnu)
bytevectors
rdelim
(srfi srfi-9 gnu)
interaction-environment
srfi-1
scheduler
operations
(guile)
r5rs
time
regexps
srfi-11
lazy
file
char
repl
eval
process-context
load
eval
ffi
exceptions
write
read
base
srfi-9
assoc
match
hashtables
read
equal
tree-il
exceptions
write
cxr
records
complex
dynamic-wind
control
keywords
dynamic-states
ports
symbols
boxes
procedures
vectors
strings
parameters
lists
syntax-objects
char
bytevectors
fluids
inexact
pairs
debug
numbers
bitvectors
cond-expand
not
optargs
eq
values
match
bitwise
atomic
case-lambda
syntax
errors
atomics

So, I want my code editor to

- be pleasant to use on the phone
- support interactive visual syntax
- provide abstract code view and a module browser