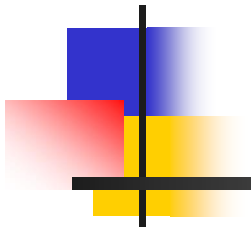
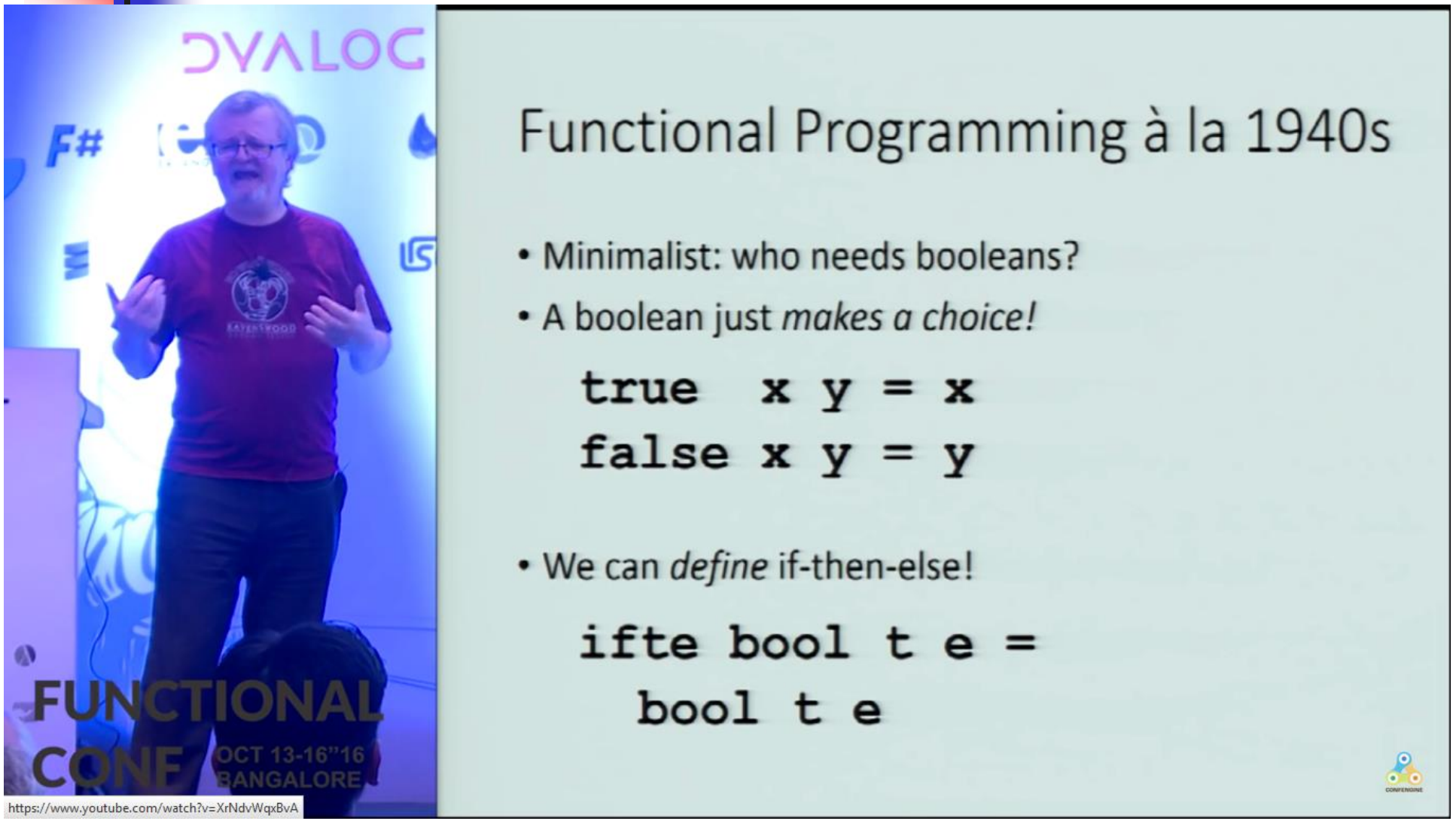


Prečo na FP záleží

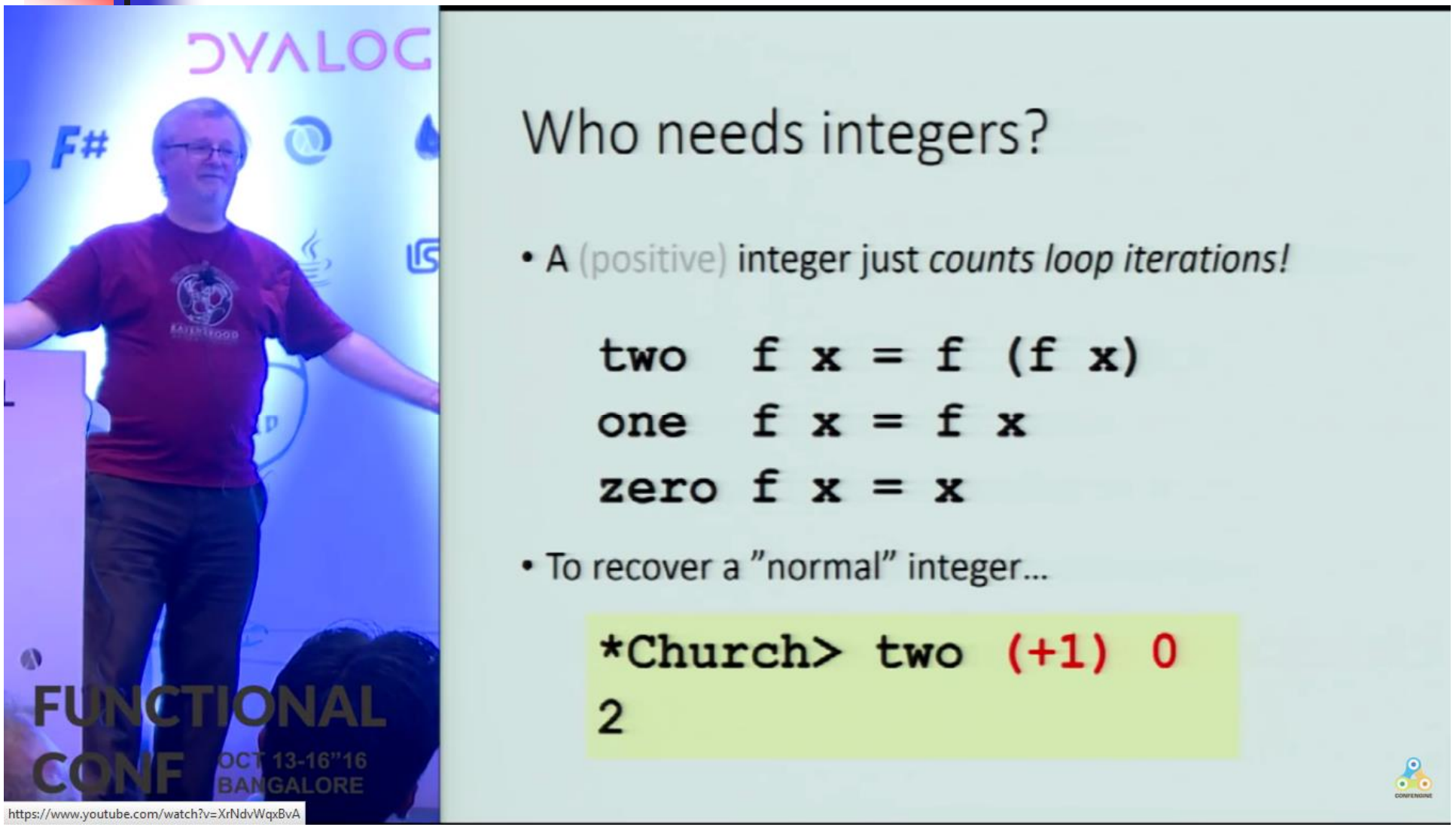
https://www.youtube.com/watch?v=XrNdvWqxBvA

www.cse.chalmers.se/~rjmh/Papers/whyfp.pdf

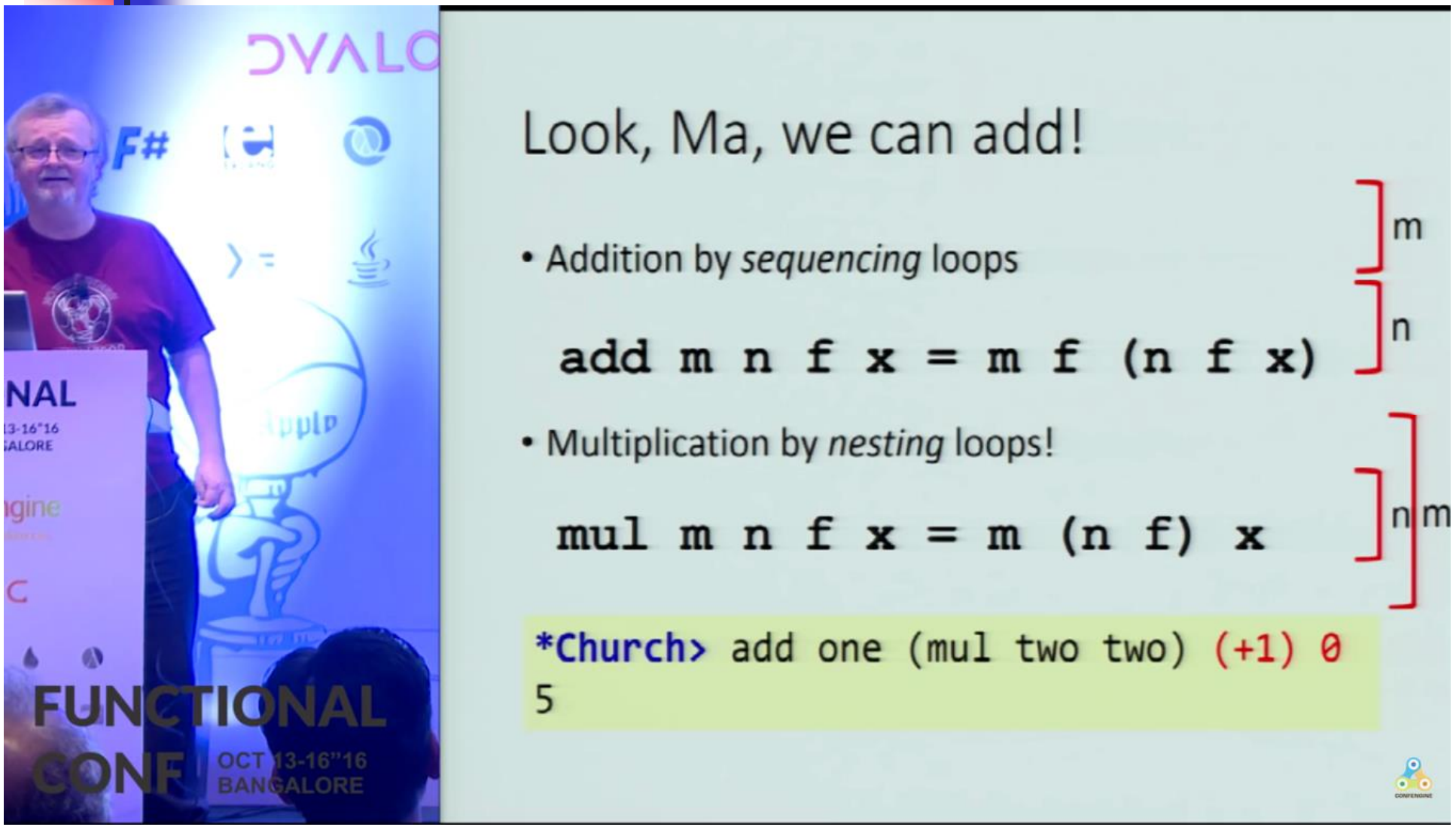
1


DVALOC
F#
FUNCTIONAL
CONF
OCT 13-16"16
BANGALORE
https://www.youtube.com/watch?v=XrNdvWqxBvA
Functional Programming à la 1940s
• Minimalist: who needs booleans?
• A boolean just *makes a choice!*
true x y = x
false x y = y
• We can *define* if-then-else!
ifte bool t e =
bool t e

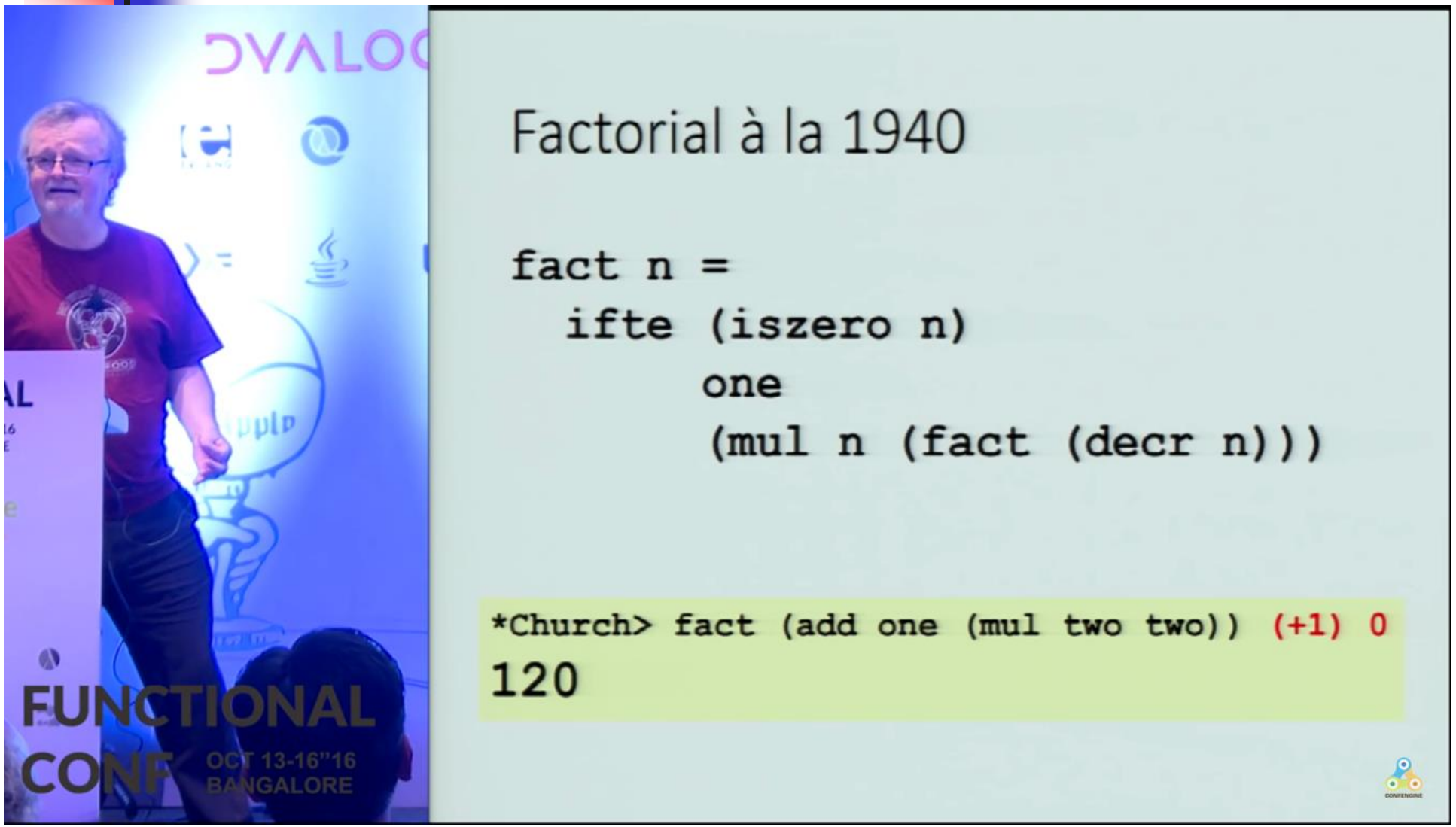
2


DYALOG
F#
FUNCTIONAL
CONF
OCT 13-16"16
BANGALORE
https://www.youtube.com/watch?v=XrNdvWqxBvA
Who needs integers?
• A (positive) integer just *counts loop iterations!*
two f x = f (f x)
one f x = f x
zero f x = x
• To recover a "normal" integer...
*Church> two (+1) 0
2

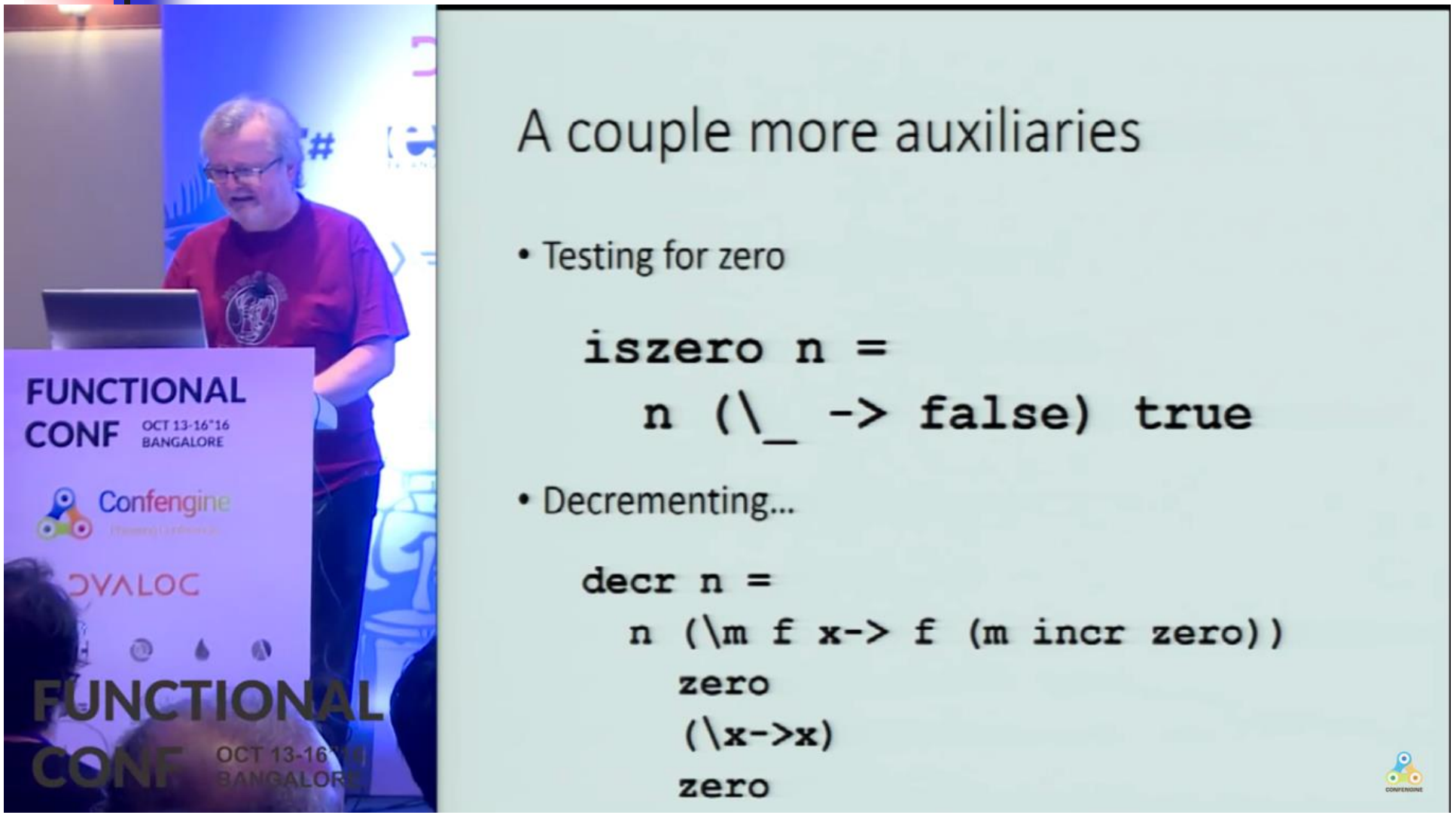
3


Look, Ma, we can add!
• Addition by *sequencing* loops
add m n f x = m f (n f x)
m
n
• Multiplication by *nesting* loops!
mul m n f x = m (n f) x
n m
*Church> add one (mul two two) (+1) 0
5
FUNCTIONAL
CONF
OCT 13-16"16
BANGALORE

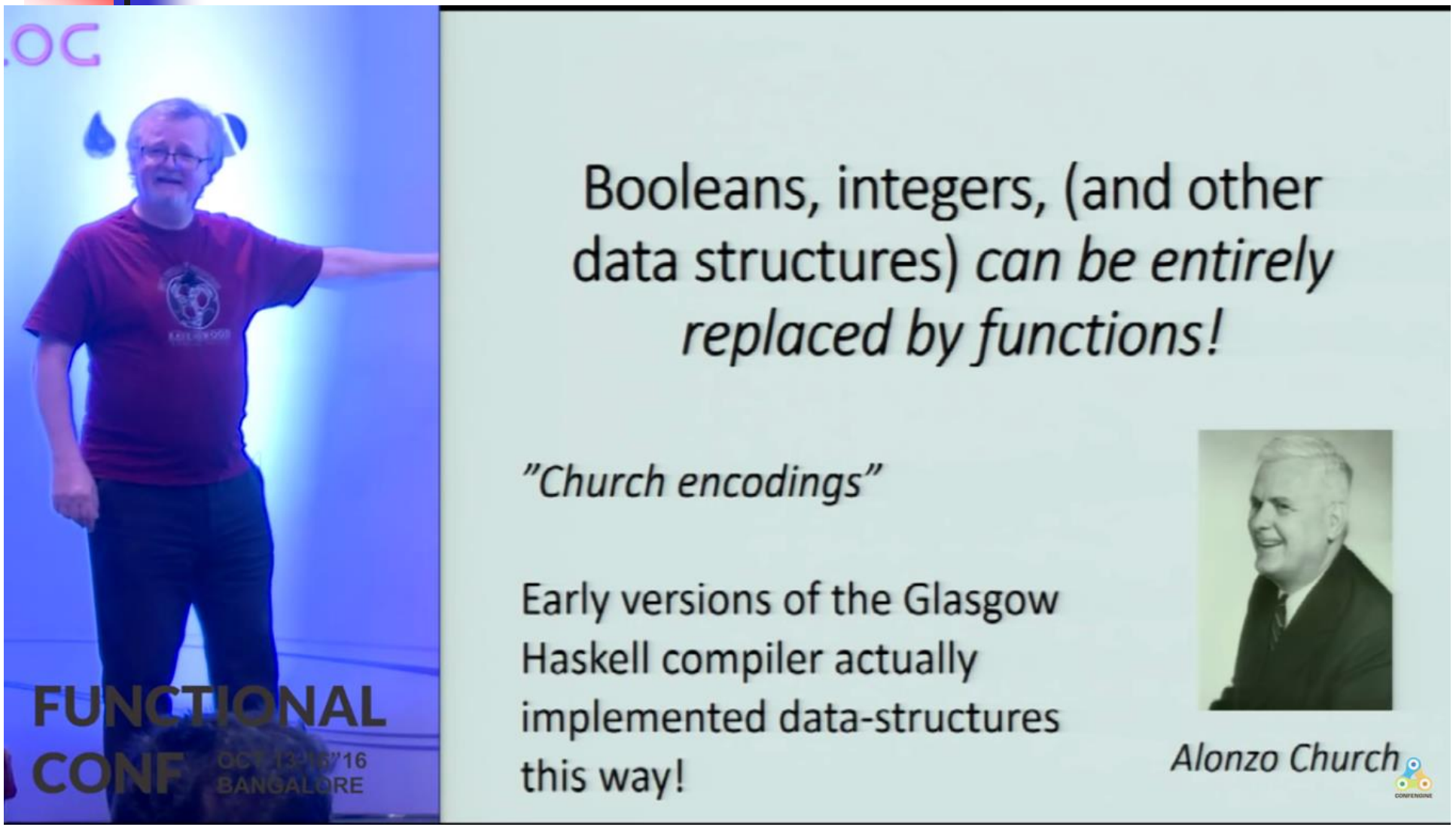
4

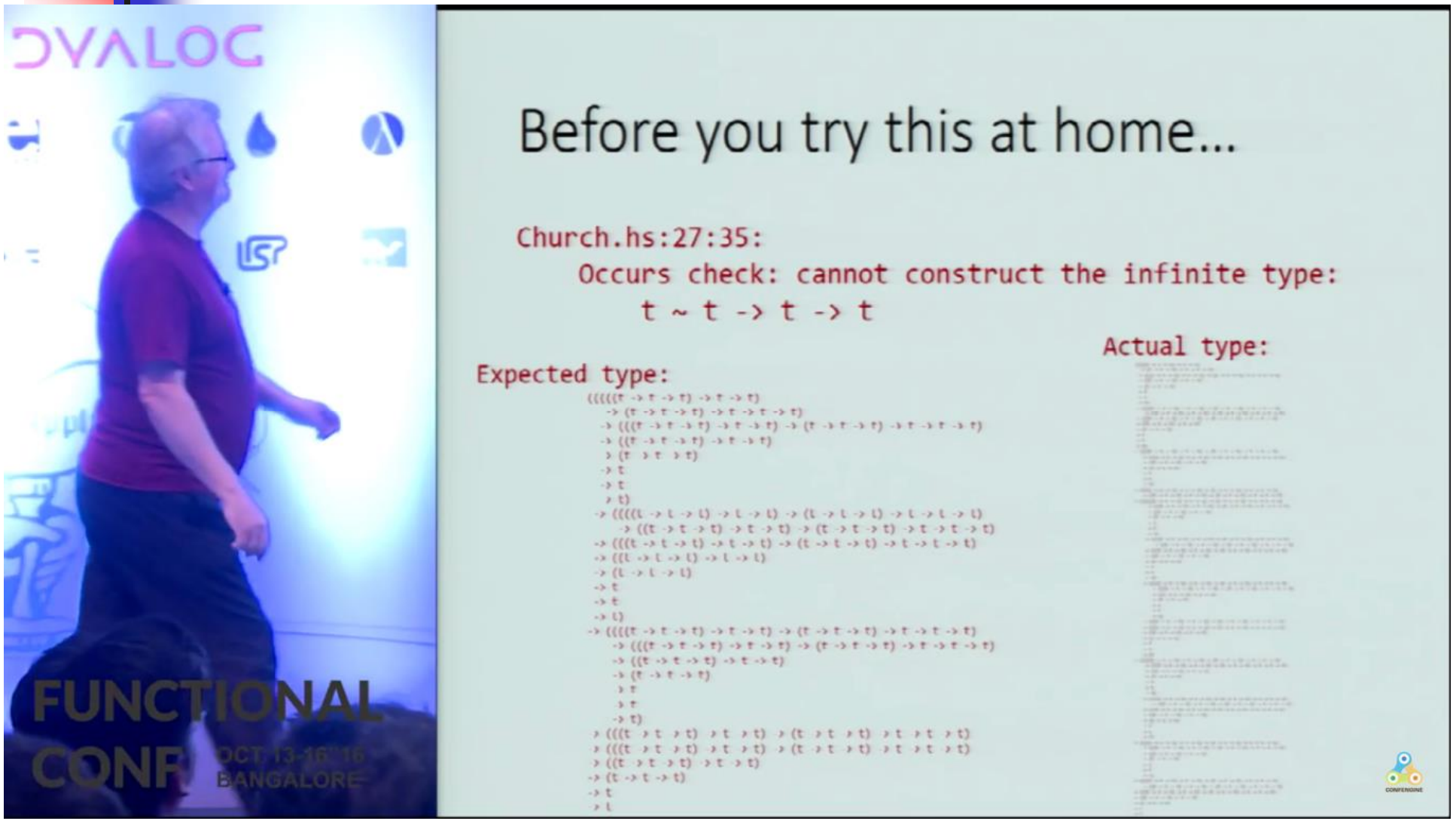

DYALOG
FUNCTIONAL
CONF
OCT 13-16"16
BANGALORE
Factorial à la 1940
fact n =
ifte (iszero n)
one
(mul n (fact (decr n)))
*Church> fact (add one (mul two two)) (+1) 0
120

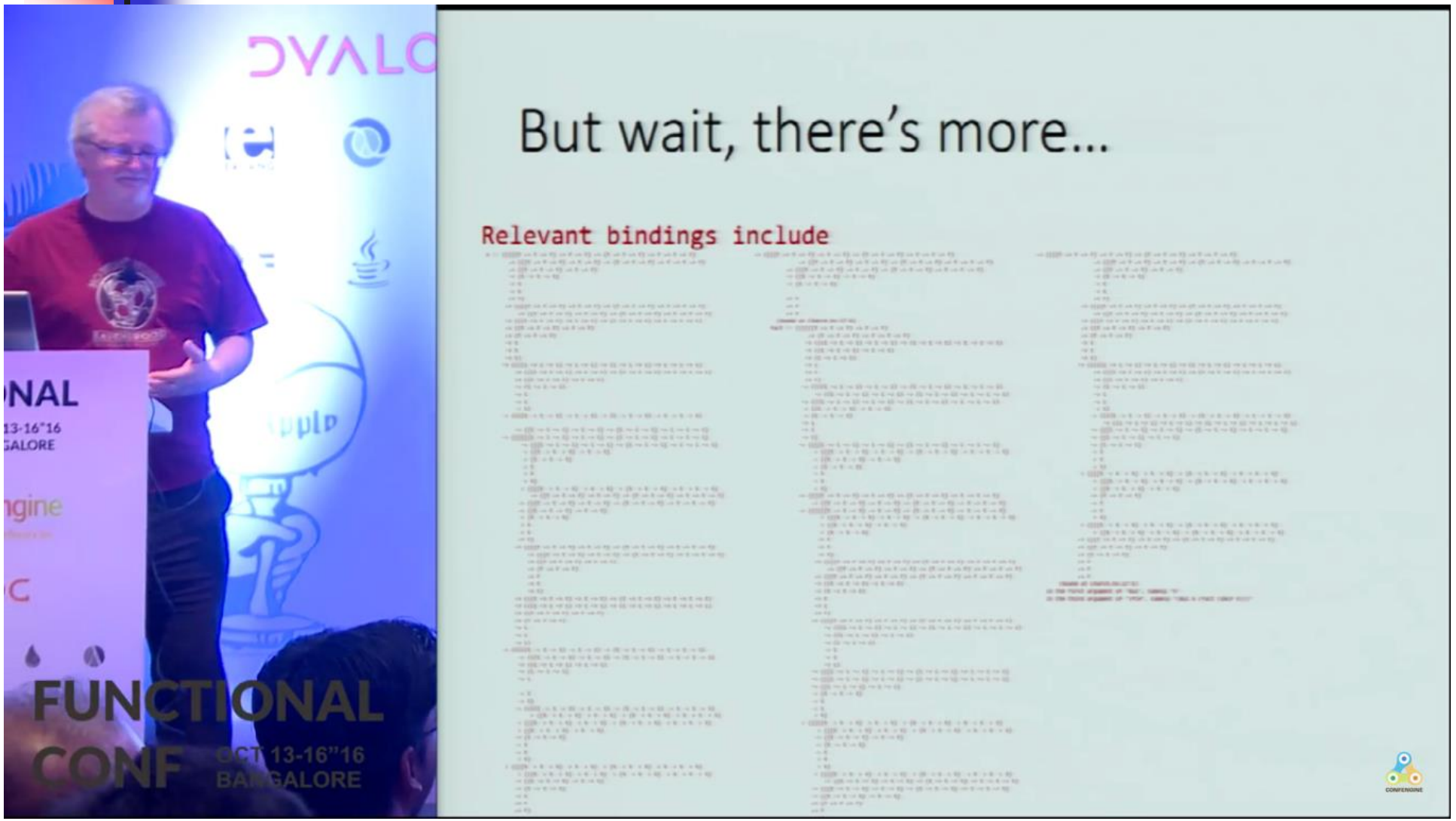
5


FUNCTIONAL
CONF
OCT 13-16 16
BANGALORE
Confengine
OVALOC
FUNCTIONAL
CONF
OCT 13-16
BANGALORE
A couple more auxiliaries
• Testing for zero
iszero n =
n (_ -> false) true
• Decrementing...
decr n =
n (\m f x-> f (m incr zero))
zero
(\x->x)
zero

6


FUNCTIONAL
CONF
BANGALORE
Booleans, integers, (and other data structures) *can be entirely replaced by functions!*
"Church encodings"
Early versions of the Glasgow Haskell compiler actually implemented data-structures this way!
Alonzo Church


DVALOC
FUNCTIONAL
CONF
OCT 13-16 '16
BANGALORE
Before you try this at home...
Church.hs:27:35:
Occurs check: cannot construct the infinite type:
t ~ t -> t -> t
Actual type:
Expected type:


But wait, there's more...
Relevant bindings include
FUNCTIONAL
CONF
OCT 13-16"16
BANGALORE

The type-checker needs a *little bit* of help

```
fact ::
   (forall a. (a->a)->a->a) ->
   (a->a) -> a -> a
```

10

Factorial à la 1960

```
(LABEL FACT (LAMBDA (N)
  (COND ((ZEROP N) 1)
        (T (TIMES N (FACT (SUB1 N)))))))
```

Higher-order functions!

```
(MAPLIST FACT (QUOTE (1 2 3 4 5)))

(1 2 6 24 120)
```

The Next 700 Programming Languages

P. J. Landin

Univac Division of Sperry Rand Corp., New York, New York

"... today ... 1,700 special programming languages used to 'communicate' in over 700 application areas."—*Computer Software* Issues, an American Mathematical Association Prospectus, July 1965.

Factorial in ISWIM

```
fac(5)
  where rec fac(n) =
    (n=1) → 1;
            n*fac(n-1)
```

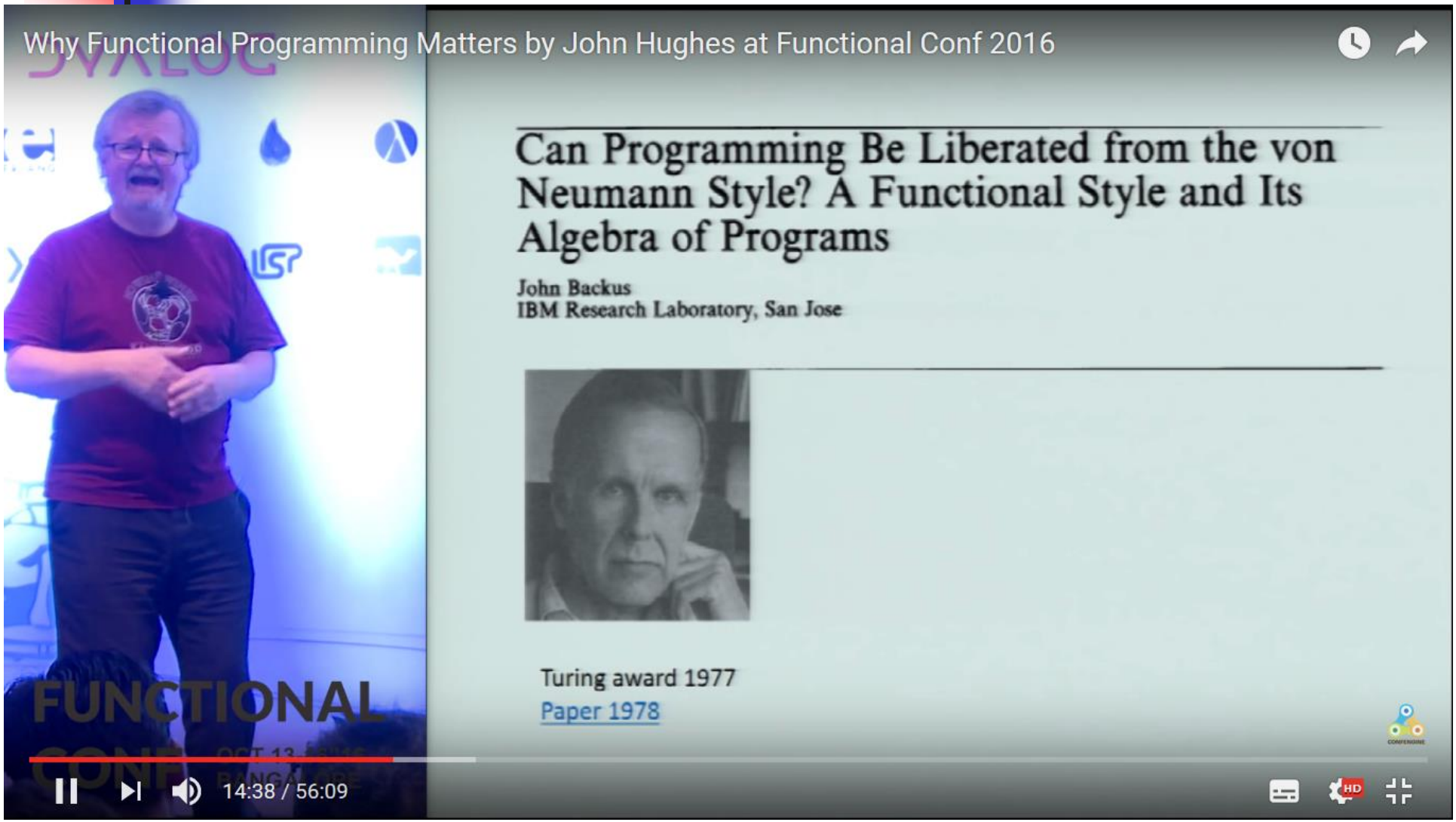
Laws

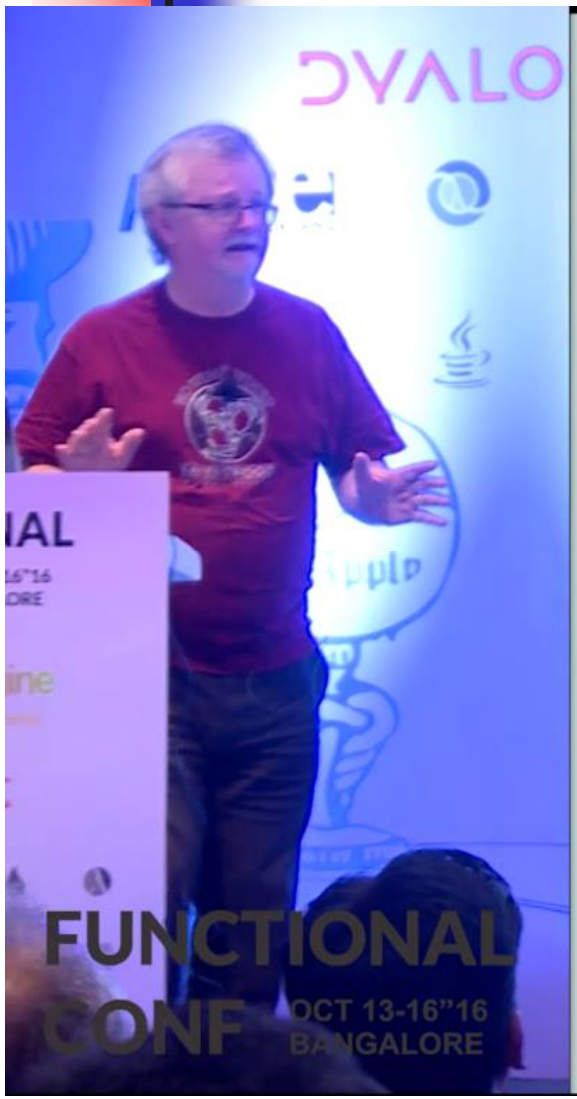
```
(MAPLIST F (REVERSE L)) ≡ (REVERSE (MAPLIST F L))
```

What's the point of two different ways to do the same thing?

Wouldn't *two* facilities be better than one?

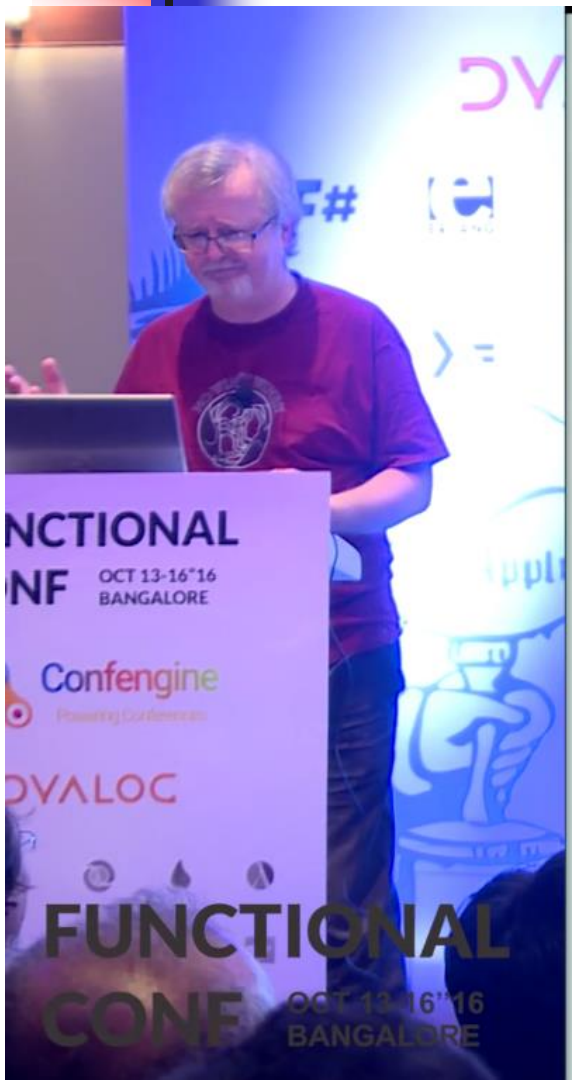
Expressive power should be by design, rather than by accident!


Why Functional Programming Matters by John Hughes at Functional Conf 2016
Can Programming Be Liberated from the von Neumann Style? A Functional Style and Its Algebra of Programs
John Backus
IBM Research Laboratory, San Jose
Turing award 1977
Paper 1978
14:38 / 56:09



Conventional programming languages are growing ever more enormous, but not stronger.

15

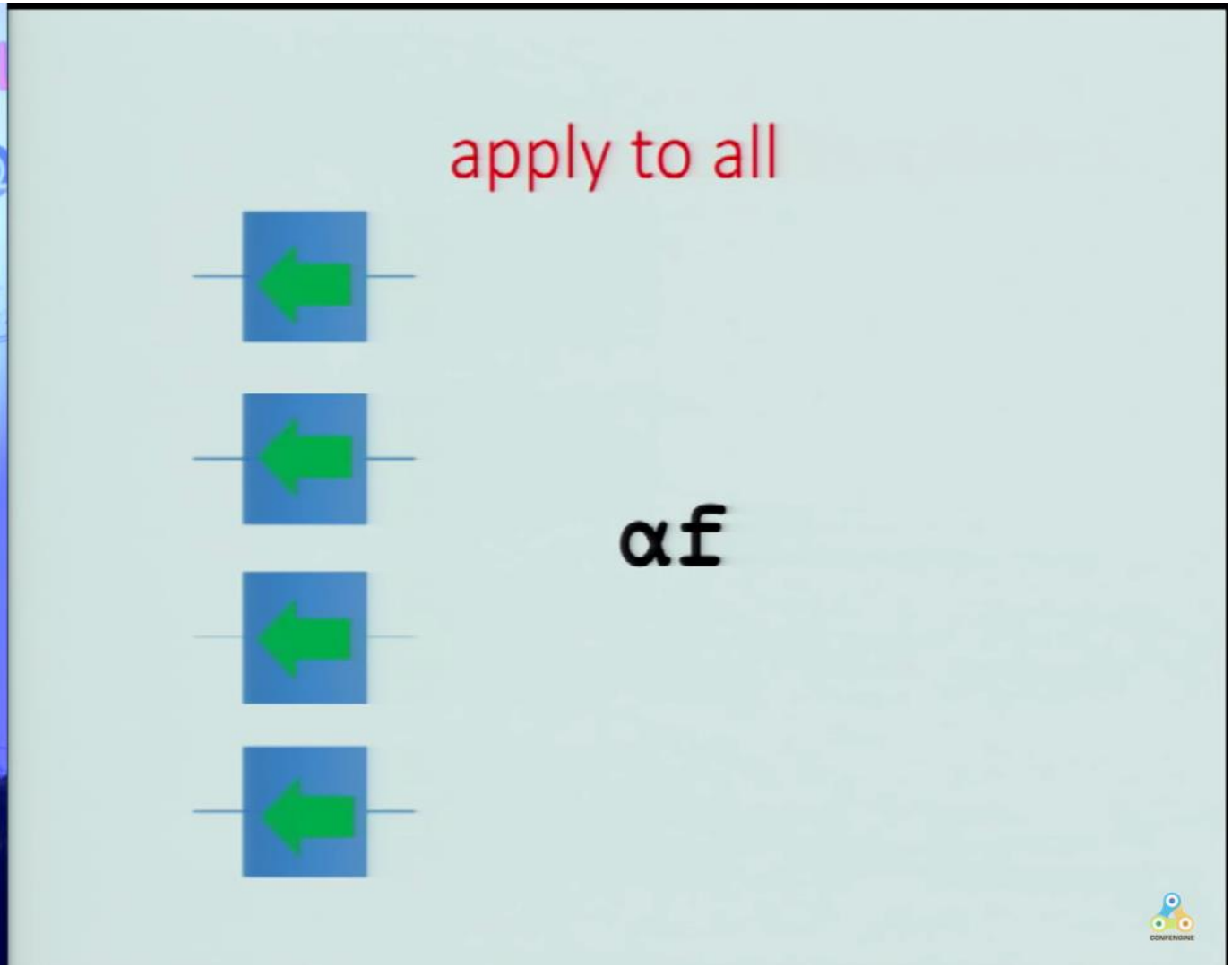


Inherent defects at the most basic level cause them to be both **fat** and **weak:**

16

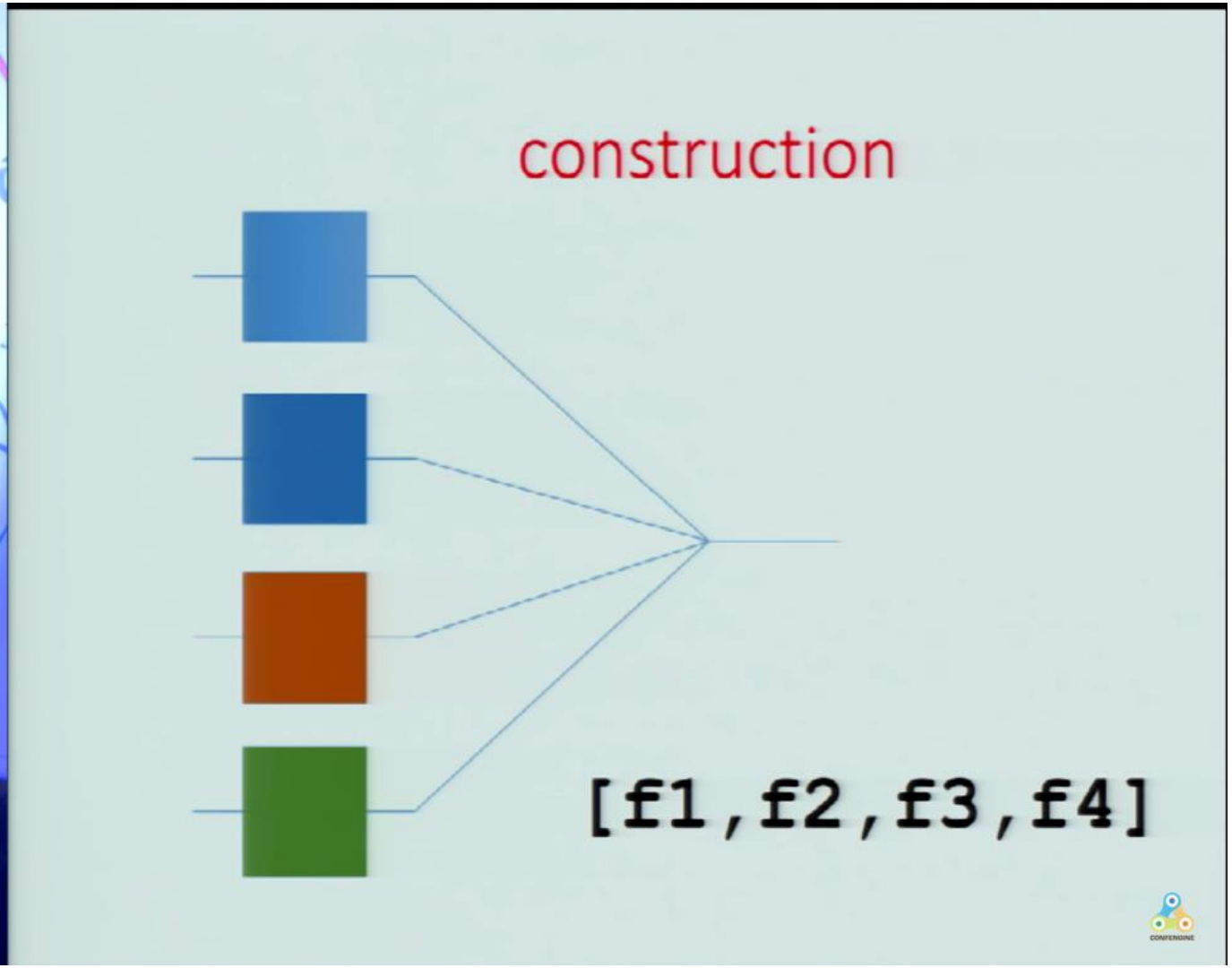
their inability to effectively use
powerful combining forms
for building new programs from
existing ones


TIONAL
OCT 13-16'16
BANGALORE
nfengine
LOC
FUNCTIONAL
CONF
OCT 13-16'16
BANGALORE

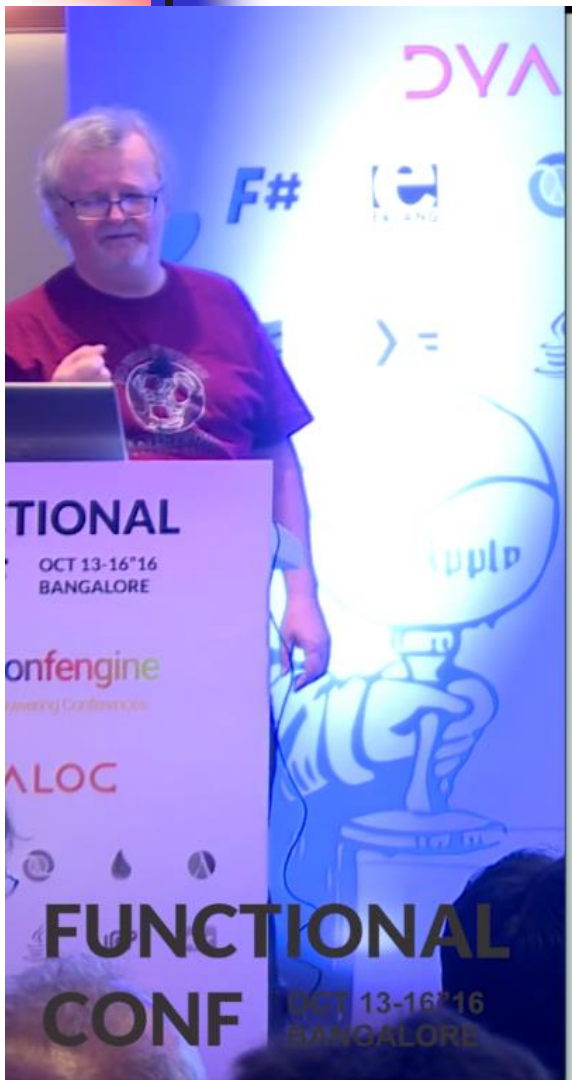

apply to all
αf

19


CTIONAL
OCT 13-16"16
BANGALORE
Confengine
VALOC
FUNCTIONAL
CONF
OCT 13-16"16
BANGALORE


construction
[f1,f2,f3,f4]

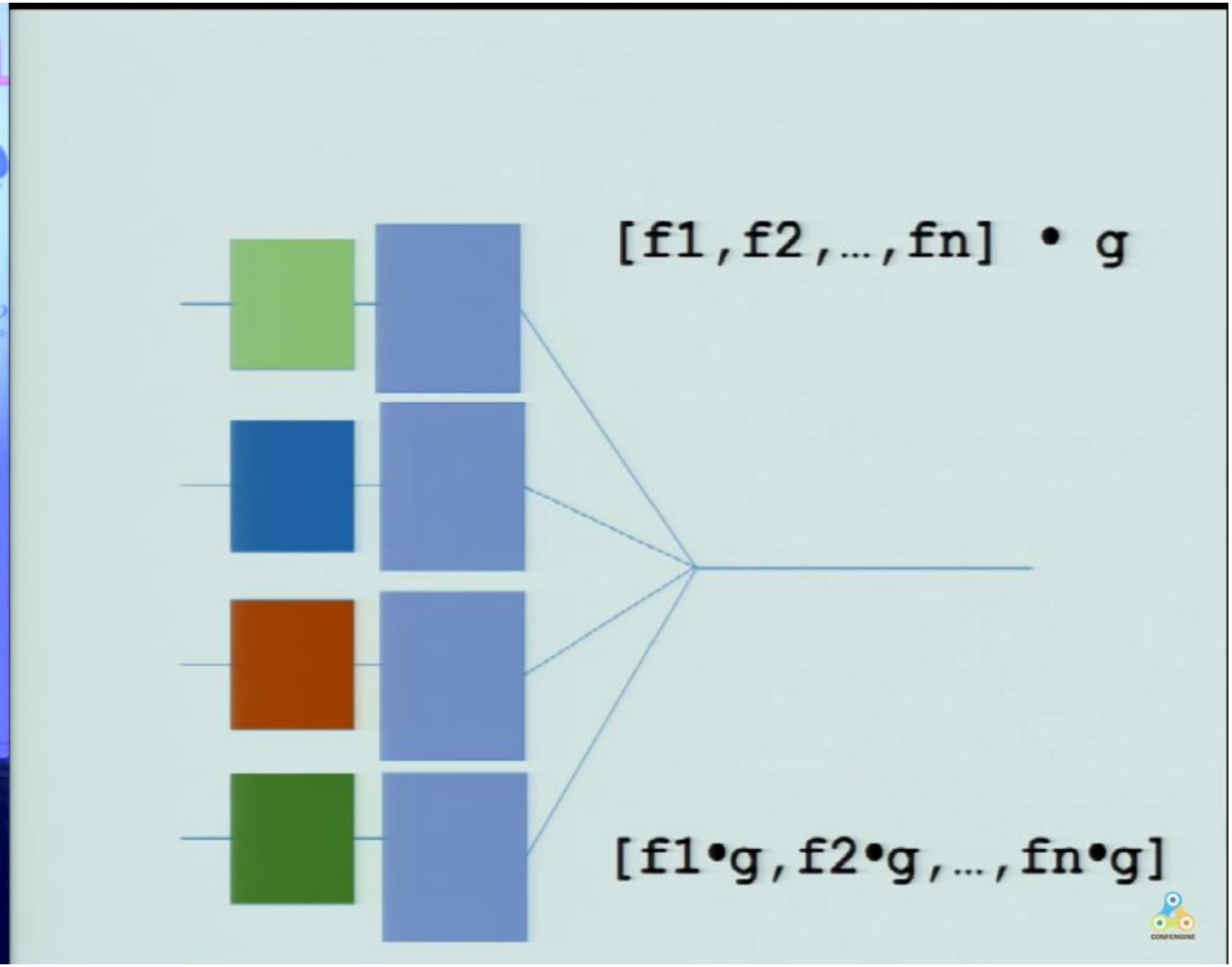
20



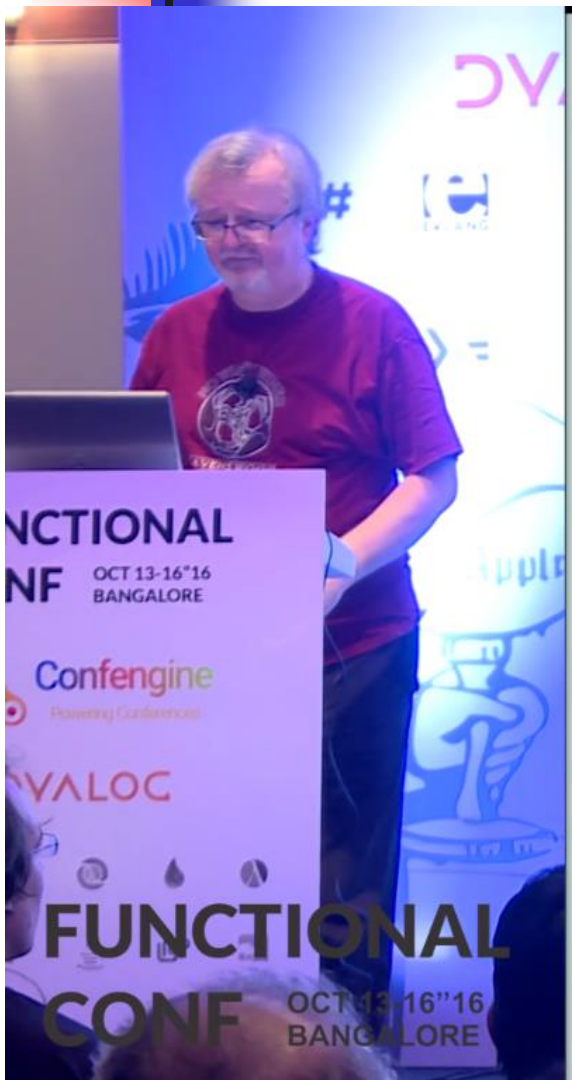
their lack of useful mathematical properties for reasoning about programs

21


IONAL
OCT 13-16"16
BANGALORE
nfengine
LOC
FUNCTIONAL
CONF
OCT 13-16"16
BANGALORE


[f1,f2,…,fn] • g
[f1•g,f2•g,…,fn•g]

22




```
Def ScalarProduct  =
   (Insert +) • (ApplyToAll ×) • Transpose
```

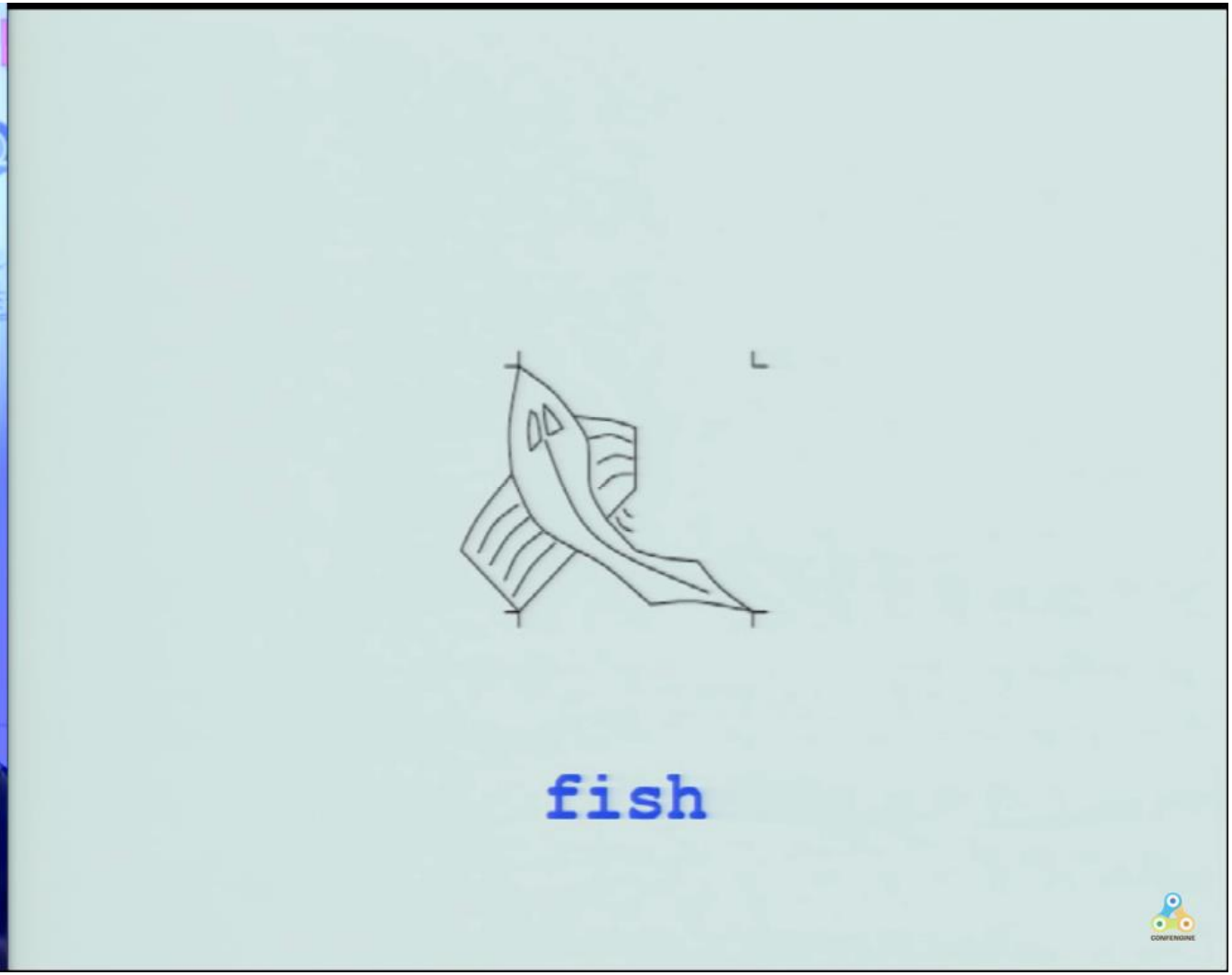
23

Def SP = (/ +) • (α ×) • Trans

Peter Henderson, Functional Geometry, 1982

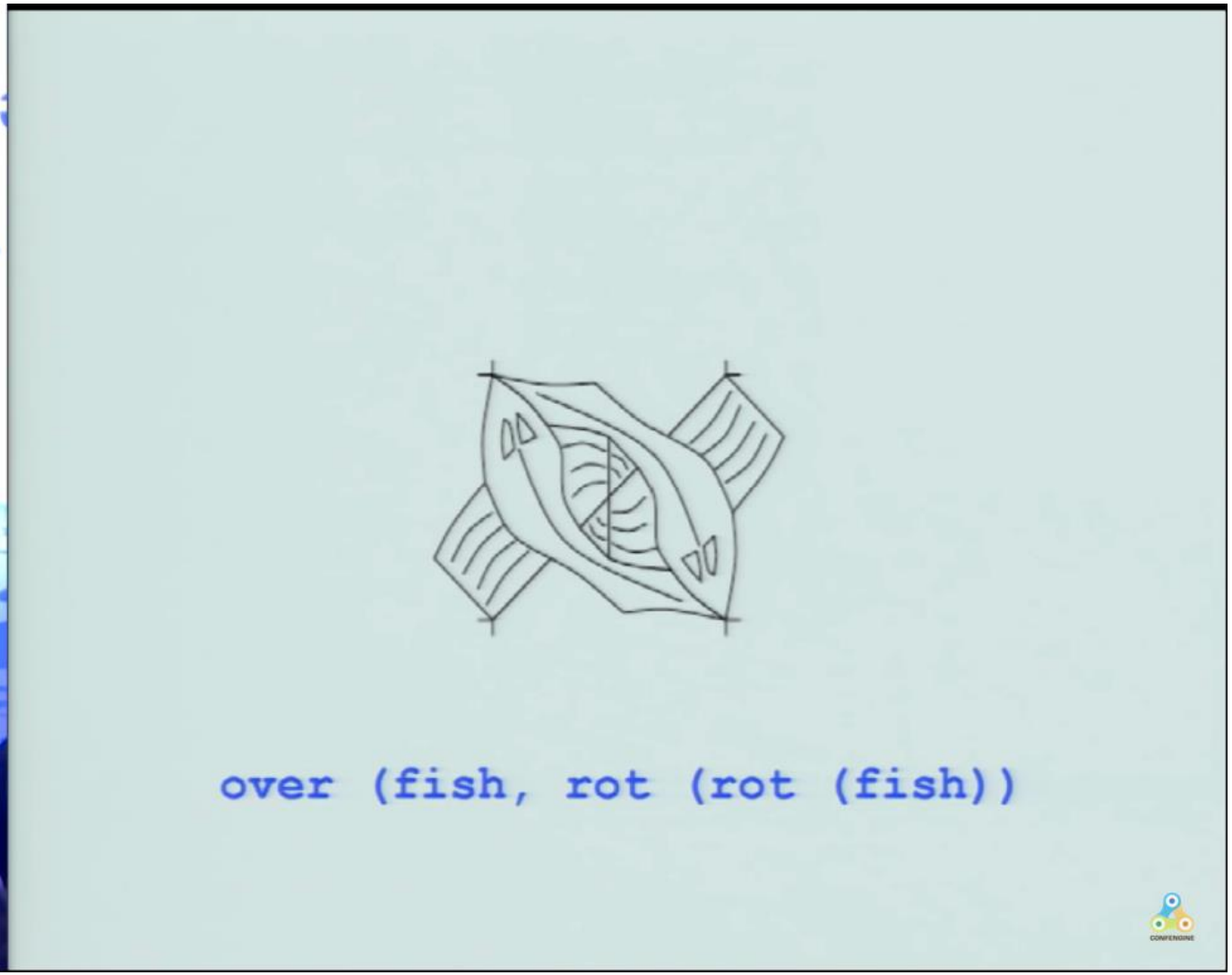
25

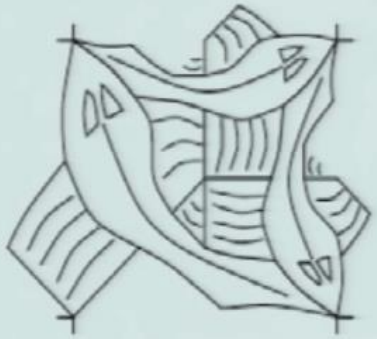

TIONAL
OCT 13-16"16
BANGALORE
nfengine
LOC
FUNCTIONAL
CONF
OCT 13-16"16
BANGALORE


fish

26


F#
FUNCTIONAL
CONF
OCT 13-16"16
BANGALORE
Confengine
FUNCTIONAL
CONF
OCT 13-16"16
BANGALORE

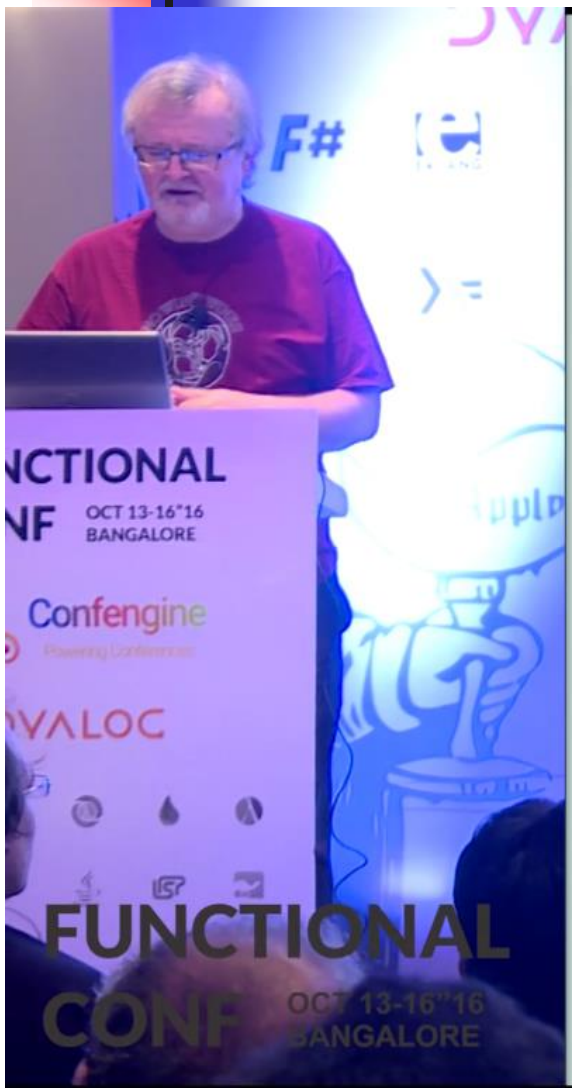

over (fish, rot (rot (fish))

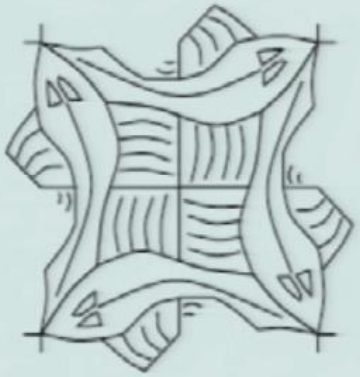


```
t = over (fish, over (fish2, fish3))

fish2 = flip (rot45 fish)
fish3 = rot (rot (rot (fish2)))
```

28





```
u = over (over (fish2, rot (fish2)),
          over (rot (rot (fish2)),
                rot (rot (rot (fish2)))))
```

P	Q
R	S

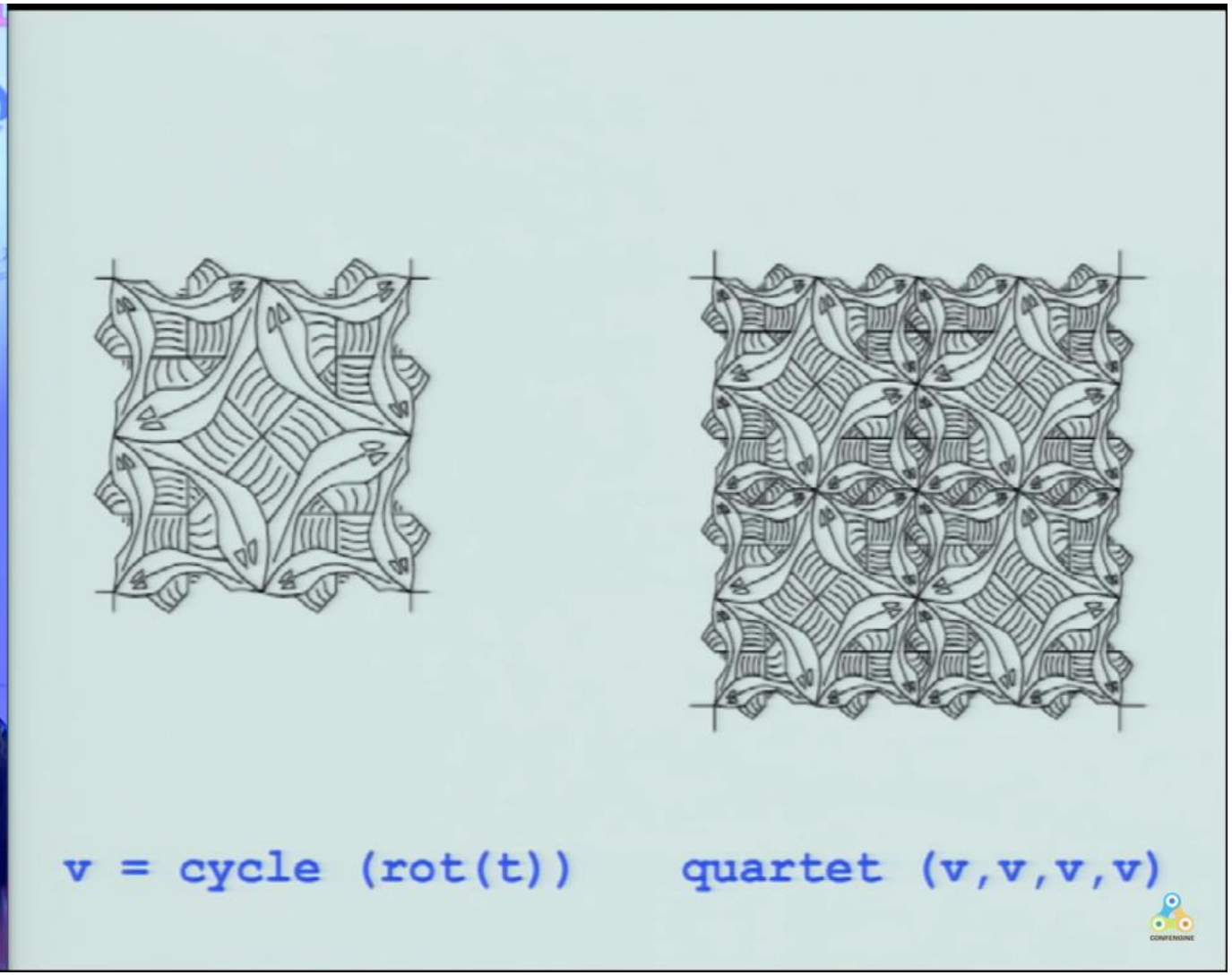
quartet

R	ꓤ
ꓤ	ꓤ

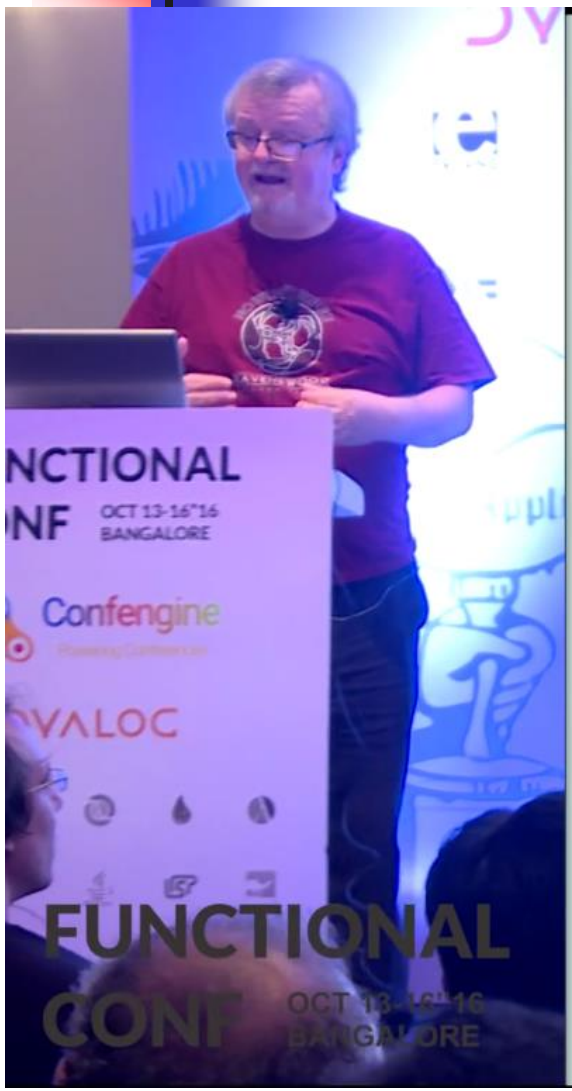
cycle

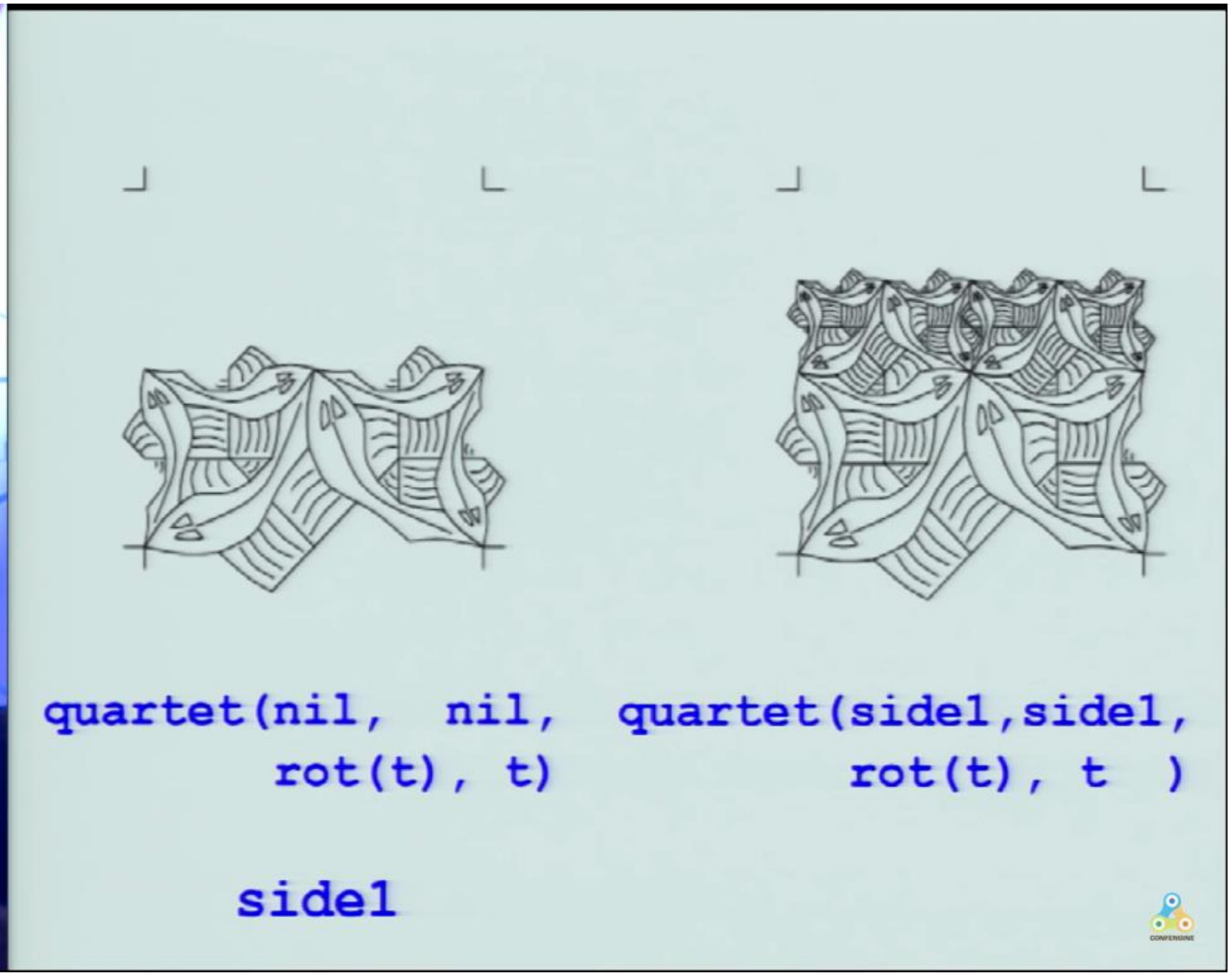
30


F#
TIONAL
OCT 13-16 '16
BANGALORE
nfengine
LOC
FUNCTIONAL
CONF
OCT 13-16 '16
BANGALORE

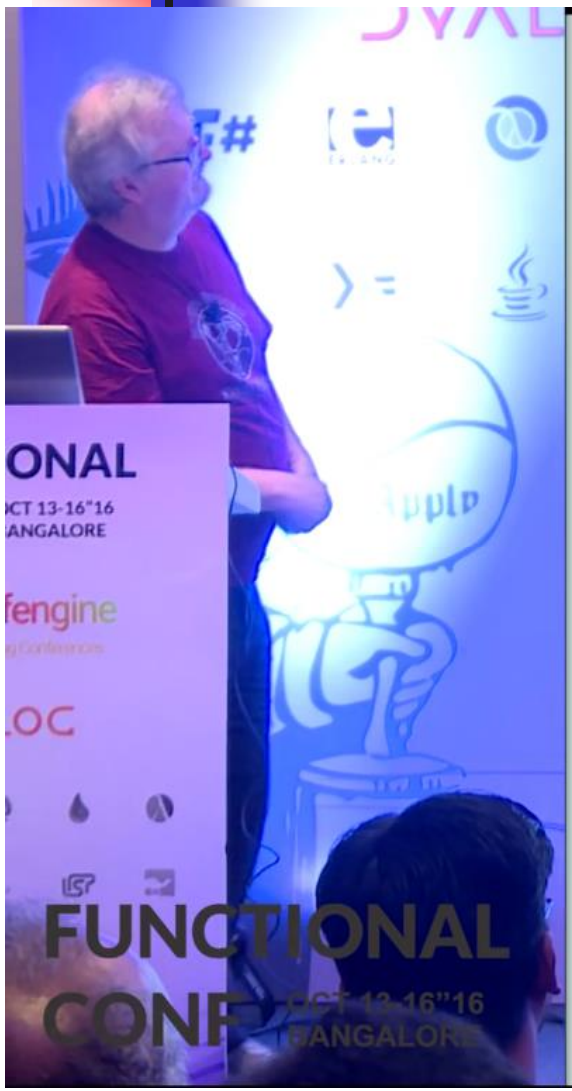

v = cycle (rot(t))
quartet (v,v,v,v)

31


NCTIONAL
NF
OCT 13-16'16
BANGALORE
Confengine
VALOC
FUNCTIONAL
CONF
OCT 13-16'16
BANGALORE


quartet(nil, nil,
rot(t), t)
quartet(side1,side1,
rot(t), t)
side1

32

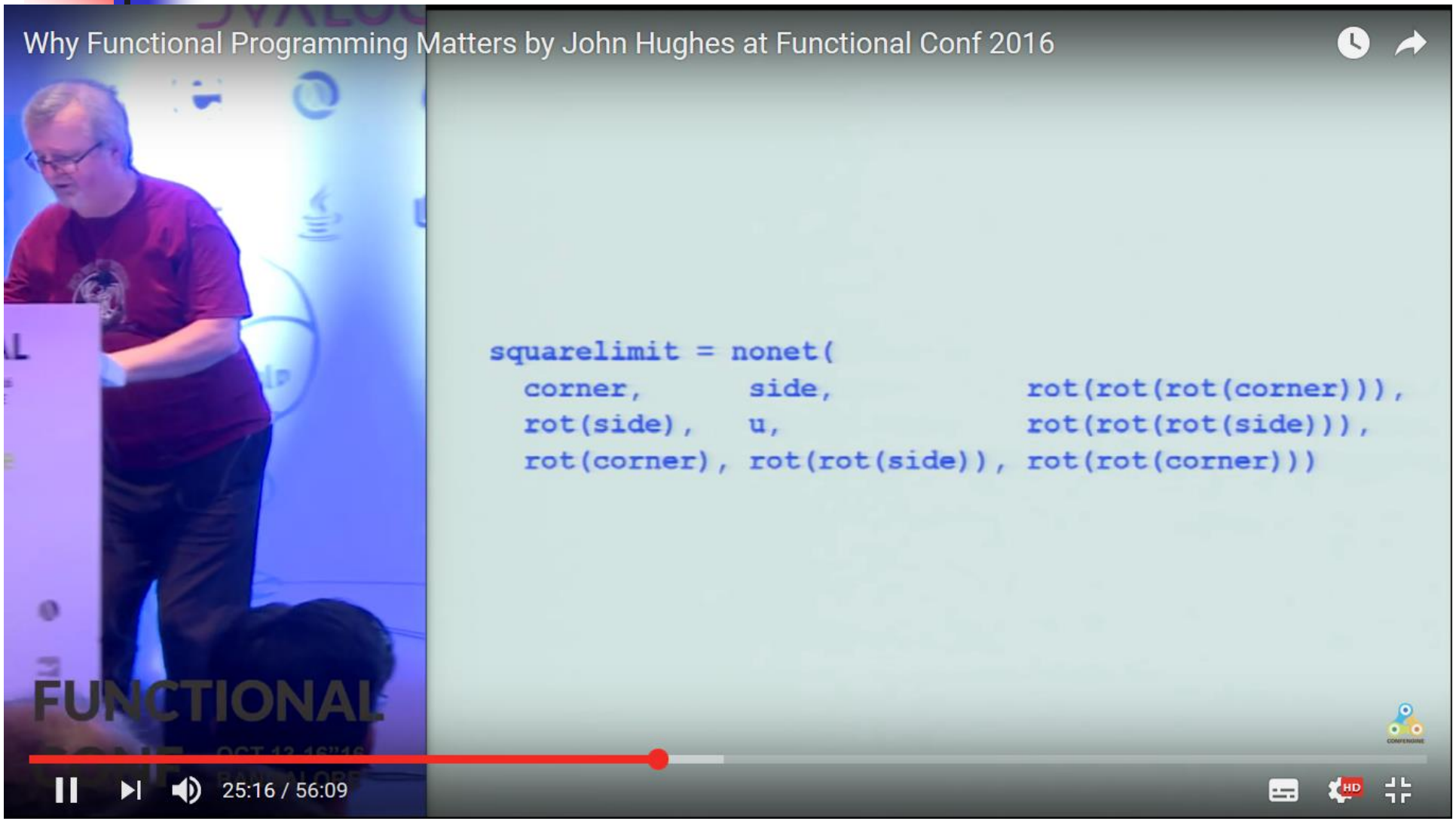


quartet (nil,nil,nil,u)

corner1

quartet(corner1,
side1,
rot(side1),
u)

33


Why Functional Programming Matters by John Hughes at Functional Conf 2016
squarelimit = nonet(
corner, side, rot(rot(rot(corner))),
rot(side), u, rot(rot(rot(side))),
rot(corner), rot(rot(side)), rot(rot(corner)))
FUNCTIONAL
25:16 / 56:09


ONAL
engine
FUNCTIONAL
CONF

Fighting spam with Haskell

Simon Marlow

One of our weapons in the fight against spam, malware, and other abuse on Facebook is a system called Sigma. Its job is to proactively identify malicious actions on Facebook, such as spam, phishing attacks, posting links to malware, etc. Bad content detected by Sigma is removed automatically so that it doesn't show up in your News Feed.

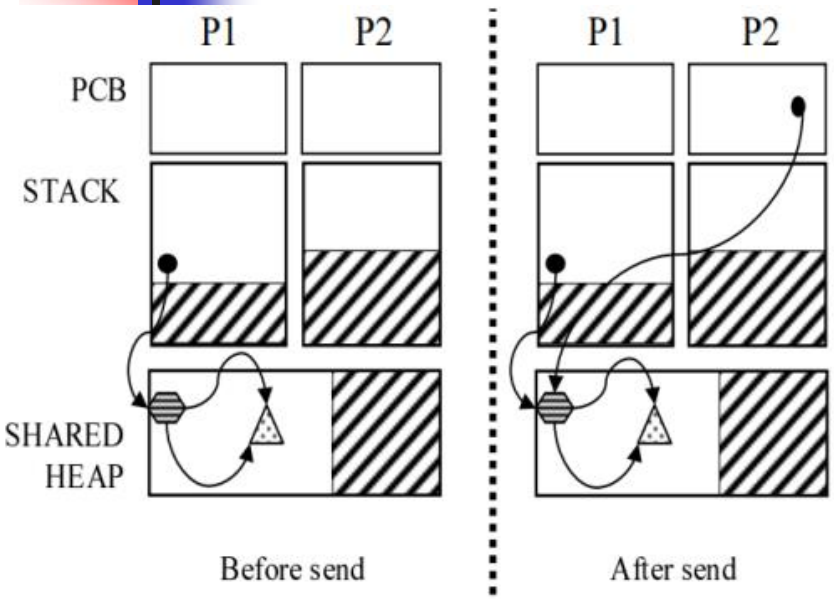
We recently completed a two-year-long major redesign of Sigma, which involved replacing the **in-house FXL language** previously used to program Sigma with **Haskell**. The Haskell-powered Sigma now runs in production, serving more than one million requests per second.

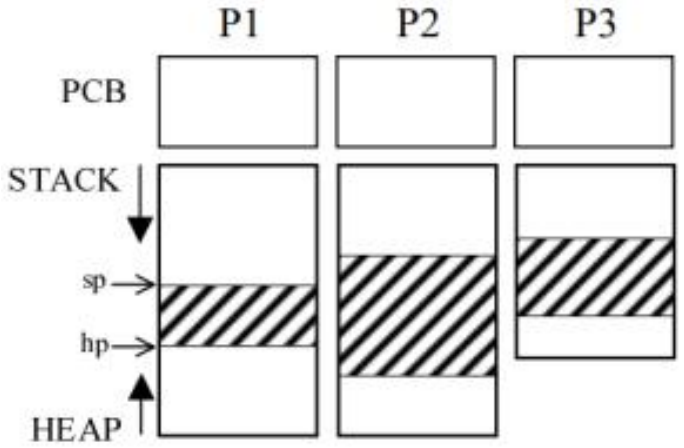
https://code.facebook.com/posts/745068642270222/fighting-spam-with-haskell/

Inside Erlang, The Rare Programming Language Behind WhatsApp's Success

Facebook's $19 billion acquisition is winning the messaging wars thanks to an unusual programming language.

https://www.fastcompany.com/3026758/inside-erlang-the-rare-programming-language-behind-whatsapps-success





https://www.quora.com/Why-is-WhatsApp-written-in-Erlang-and-not-in-Java

- asdas