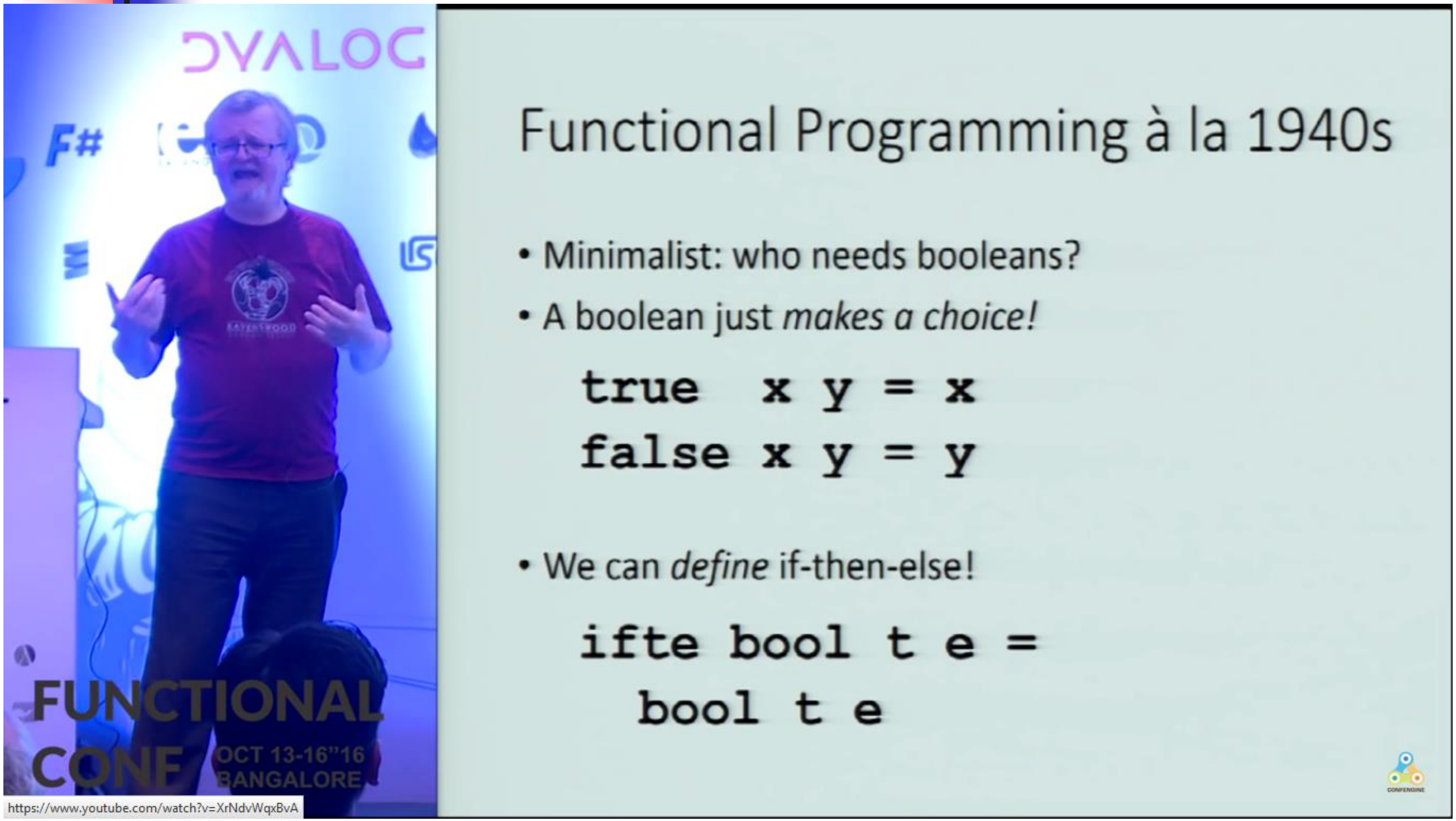
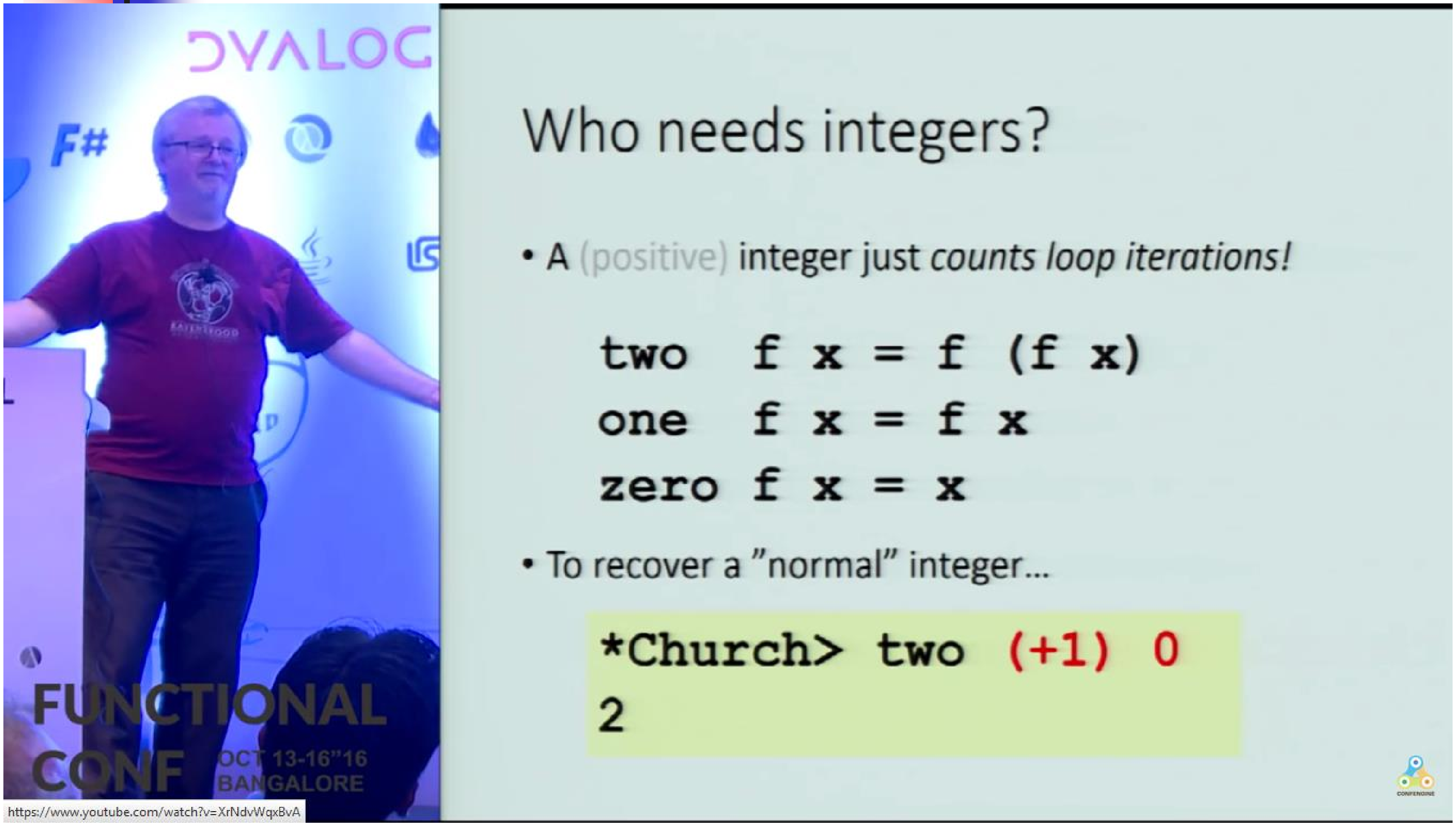
Prečo na FP záleží

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

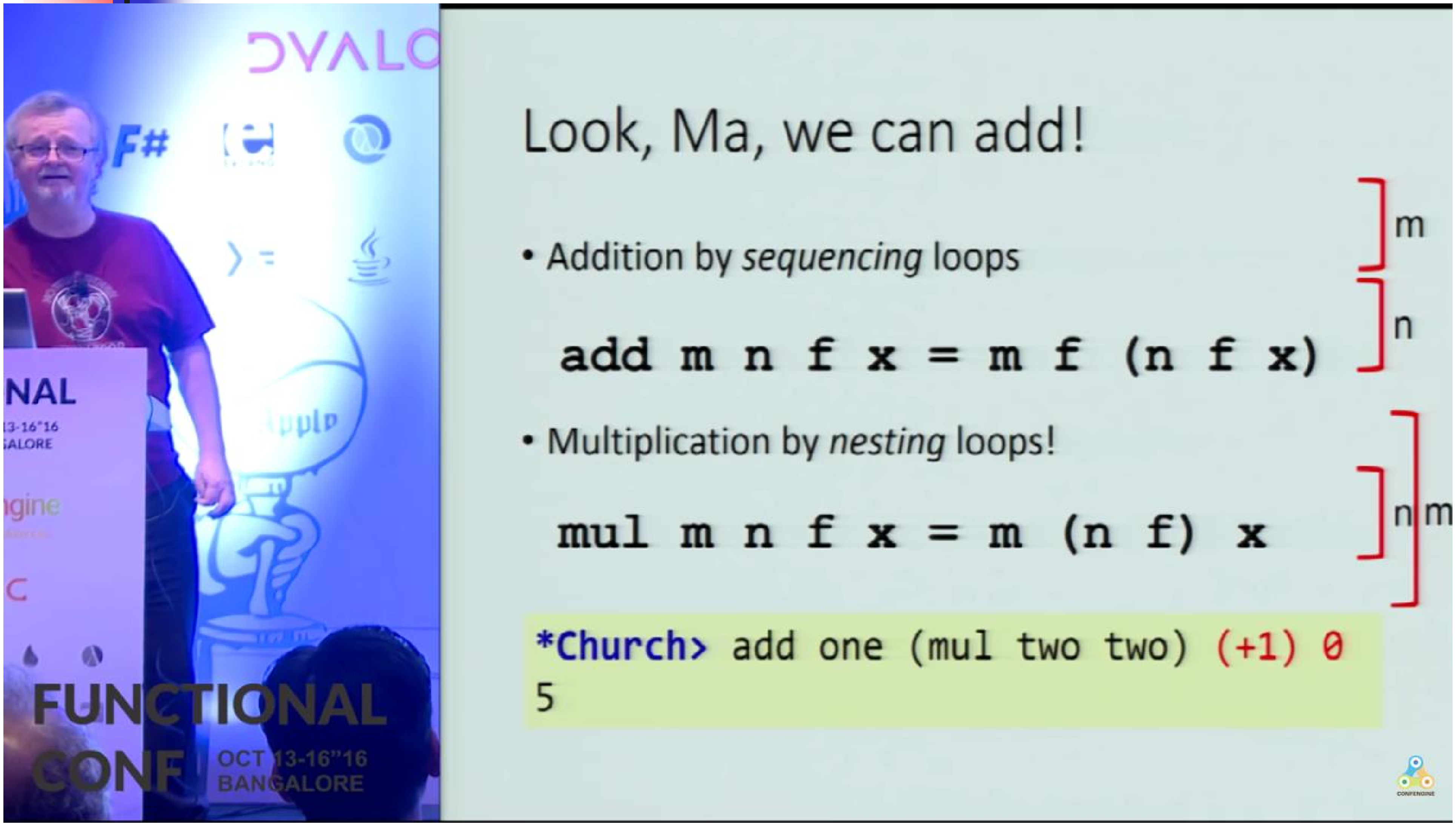
www.cse.chalmers.se/~rjmh/Papers/whyfp.pdf


DYALOG
F#
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
FUNCTIONAL
CONF
OCT 13-16''16
BANGALORE
https://www.youtube.com/watch?v=XrNdvWqxBvA

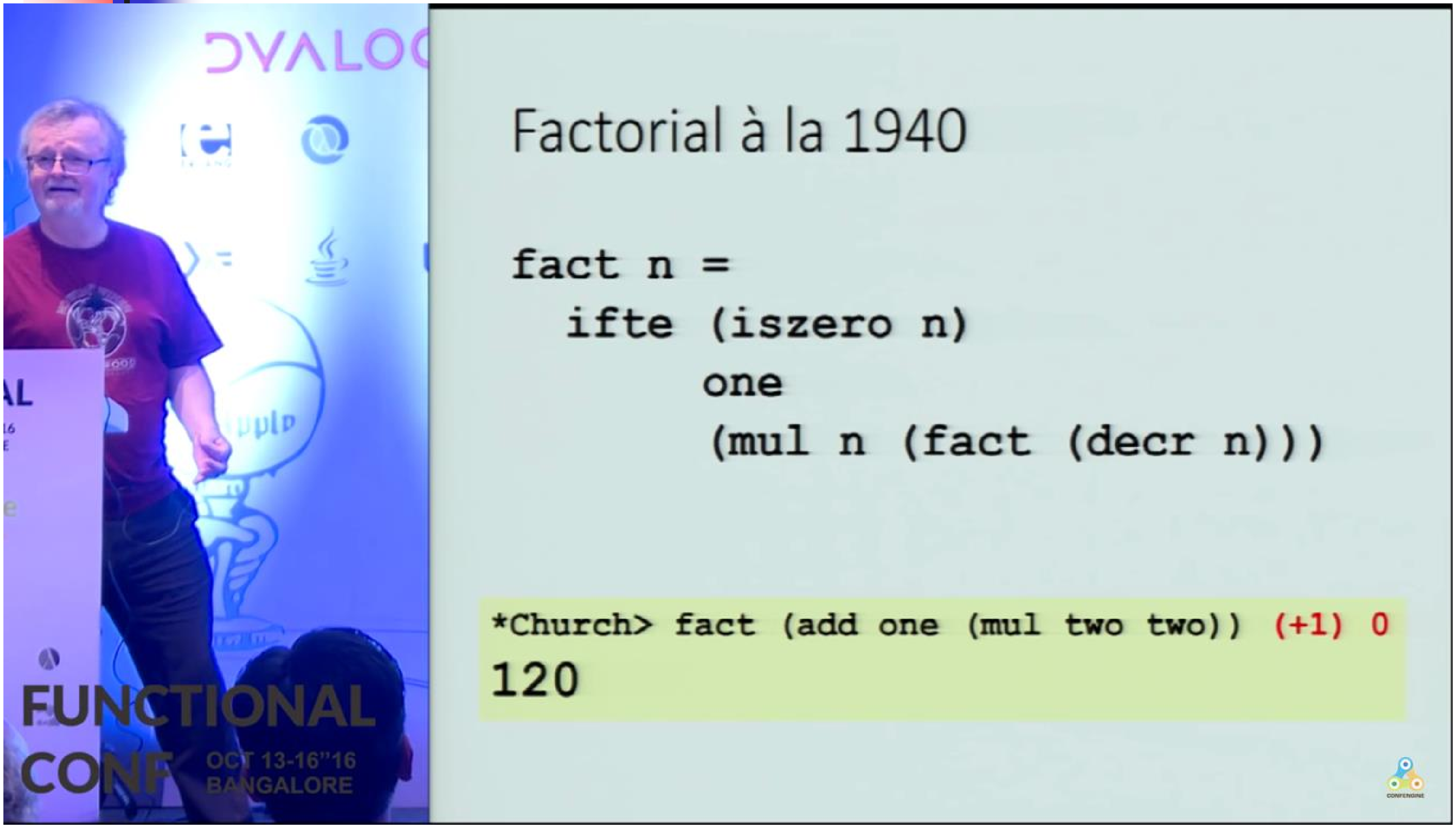
Churchove čísla



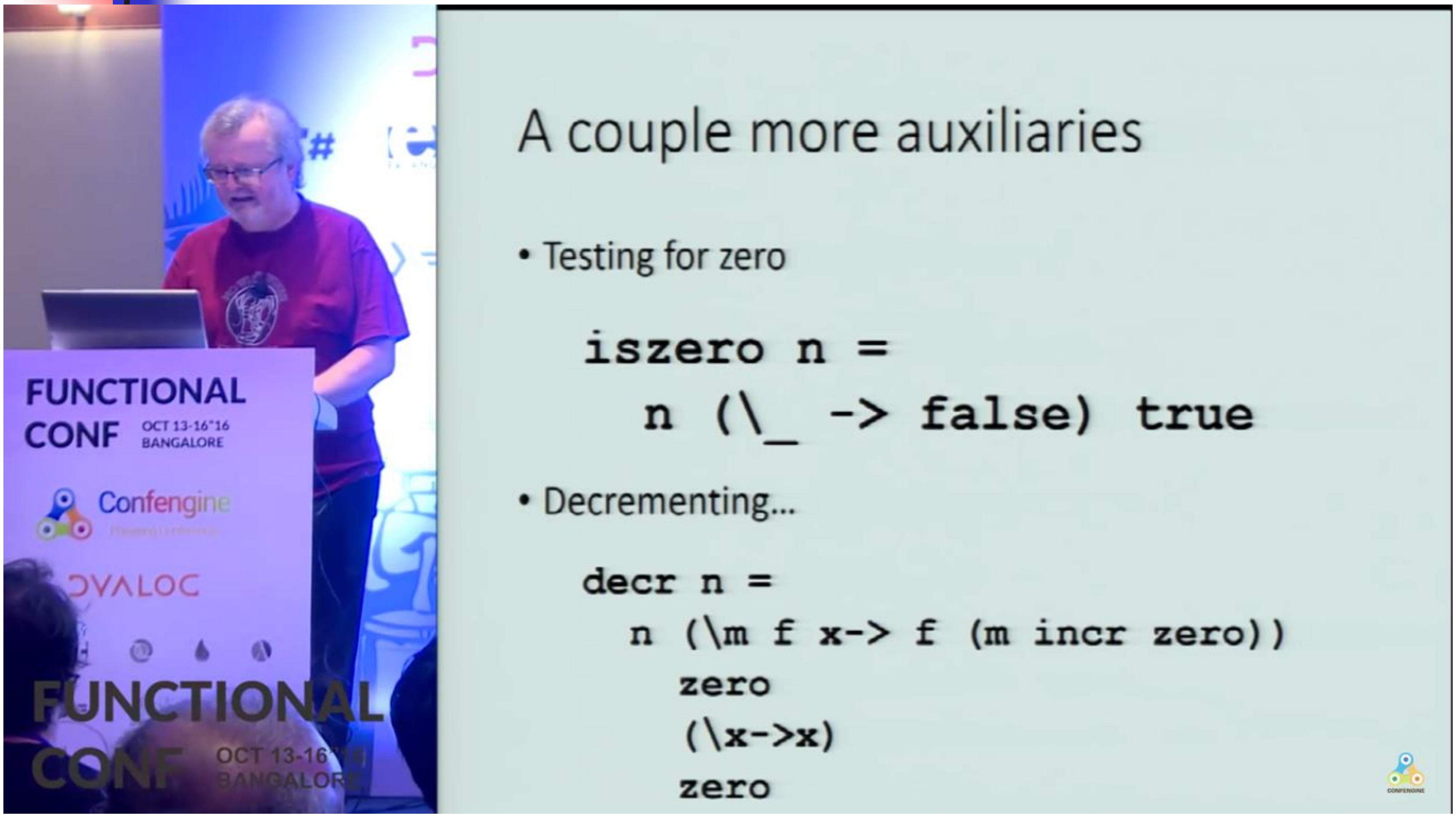
Churchove čísla



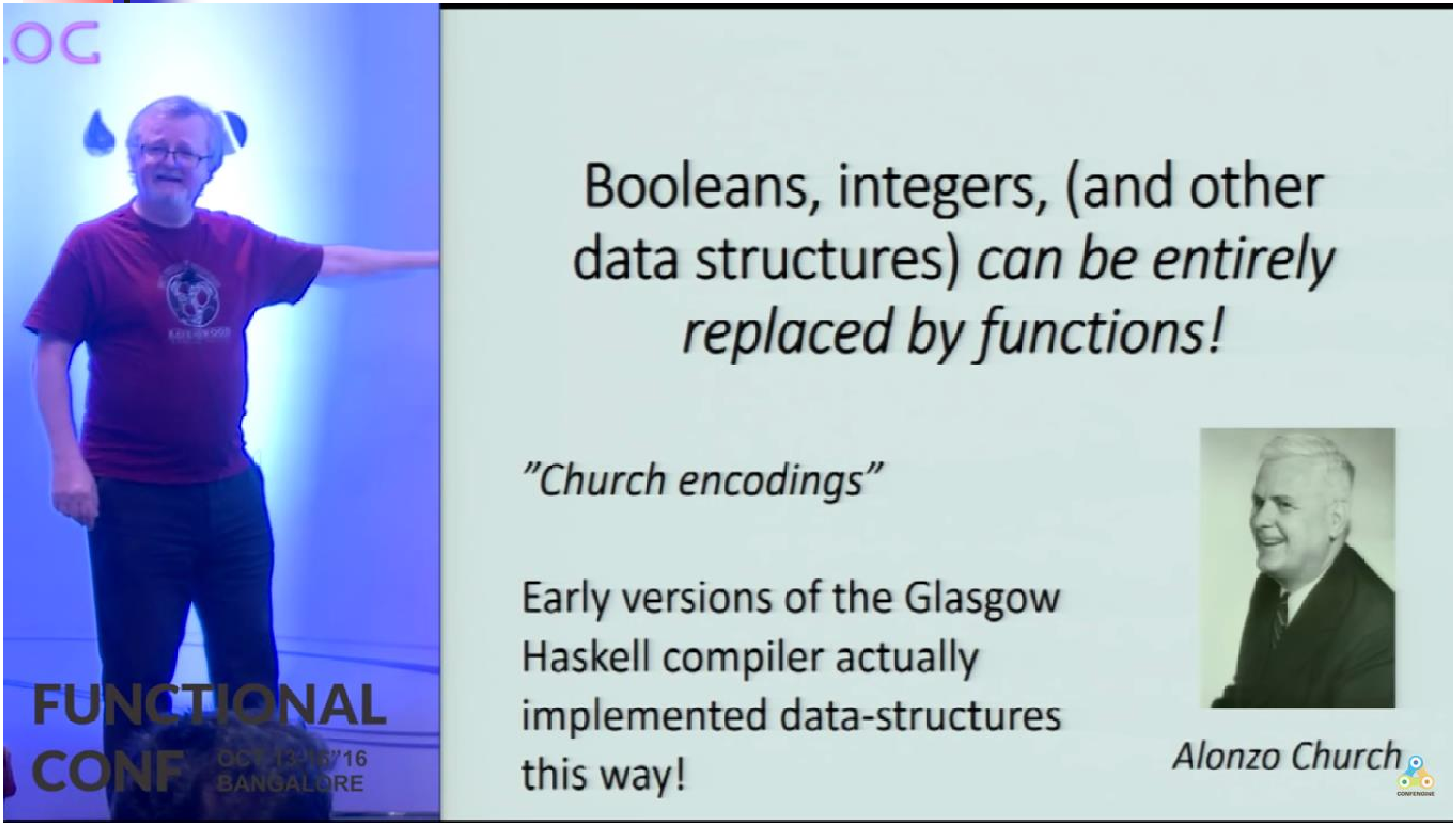
Churchove čísla

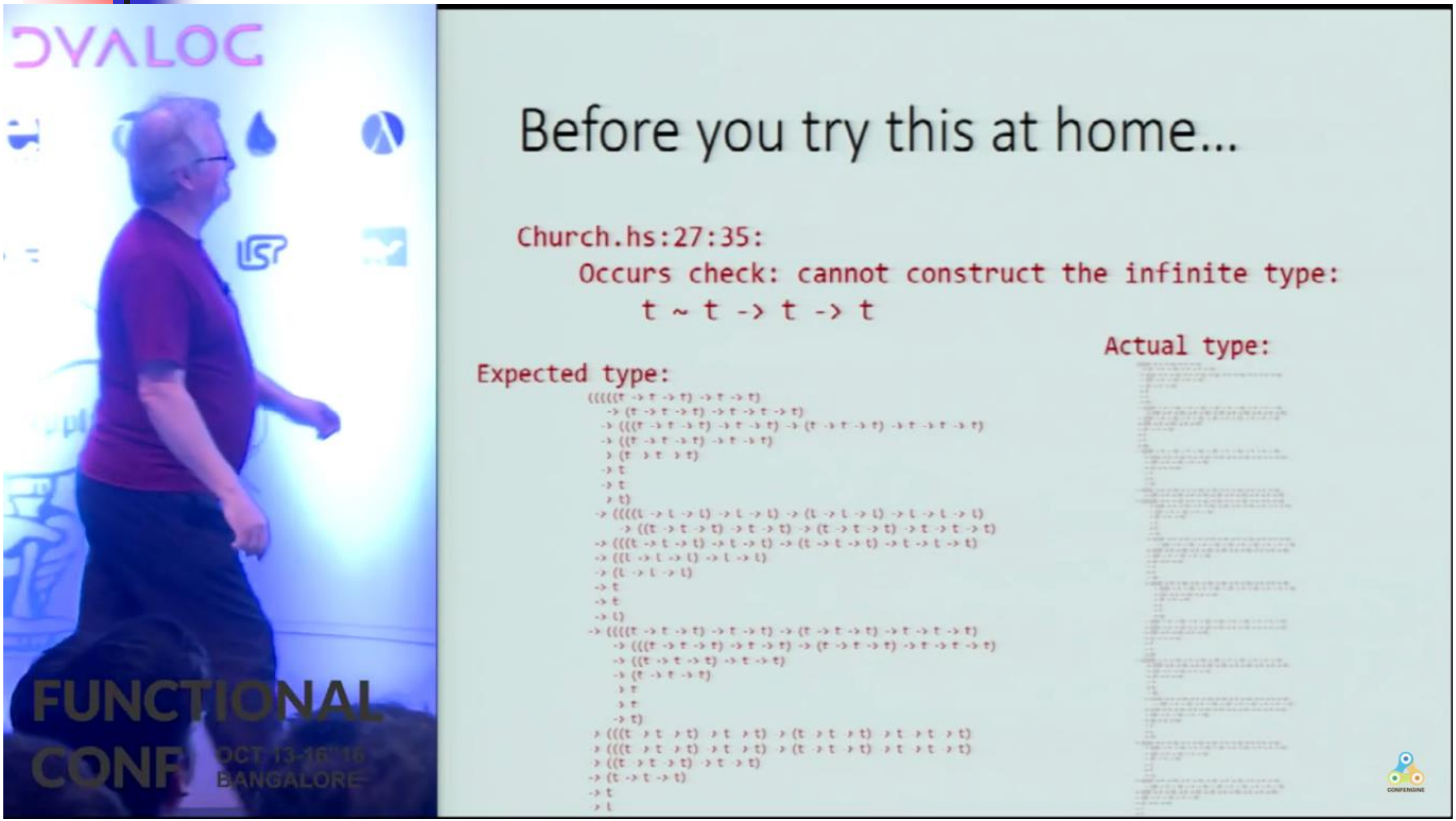


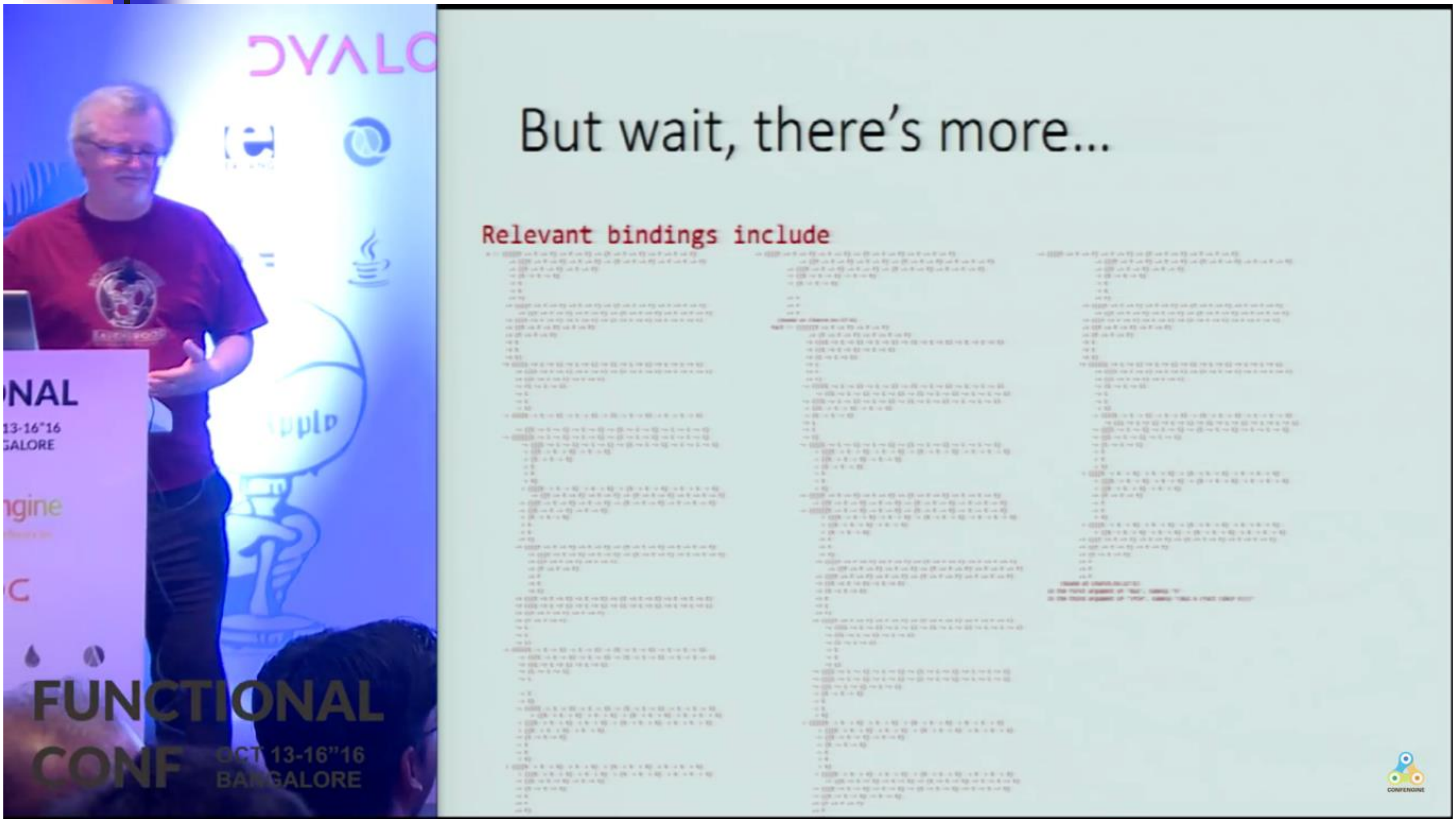
Churchove čísla

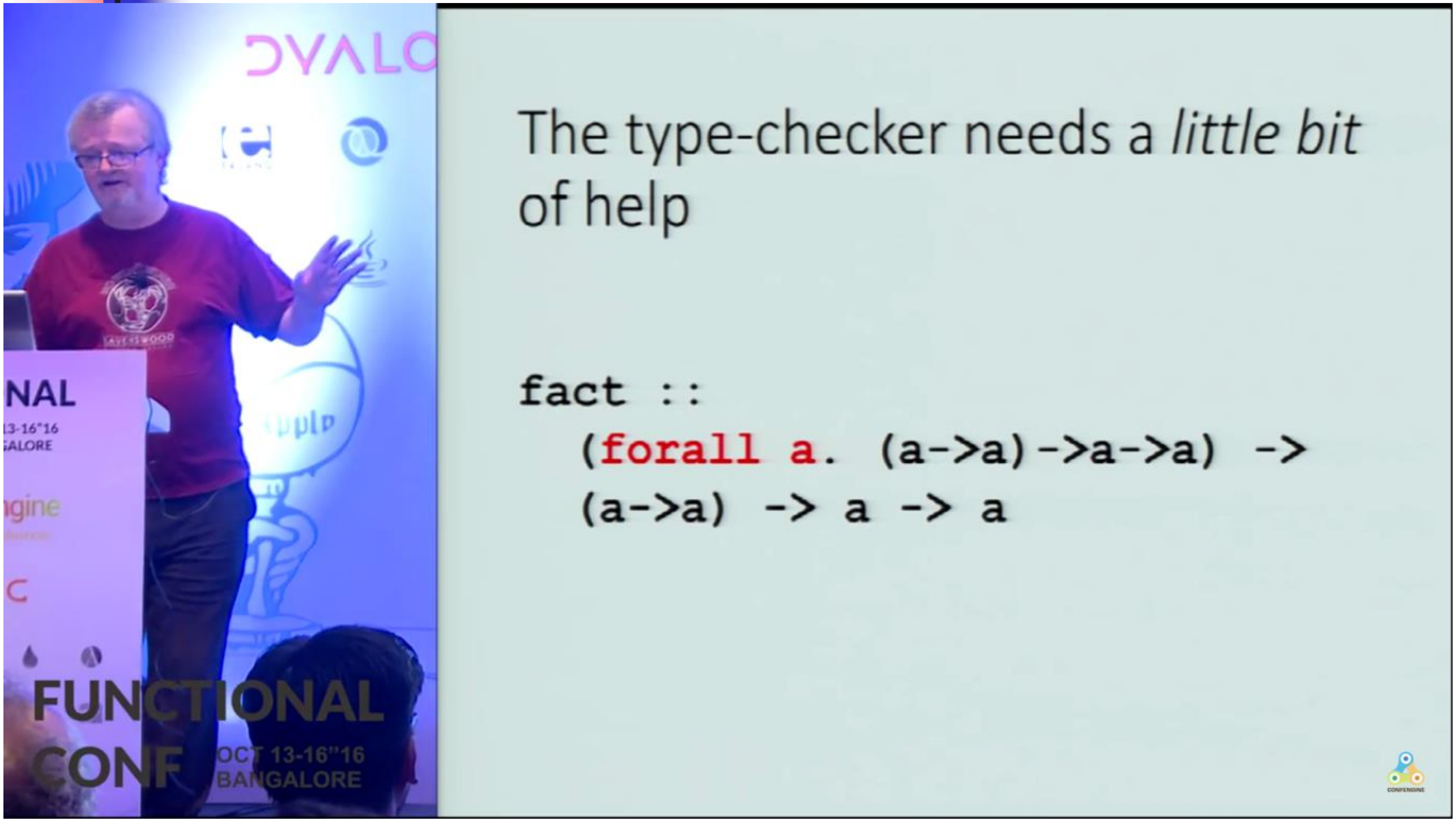


Churchove čísla




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:
FUNCTIONAL
CONF
BANGALORE


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

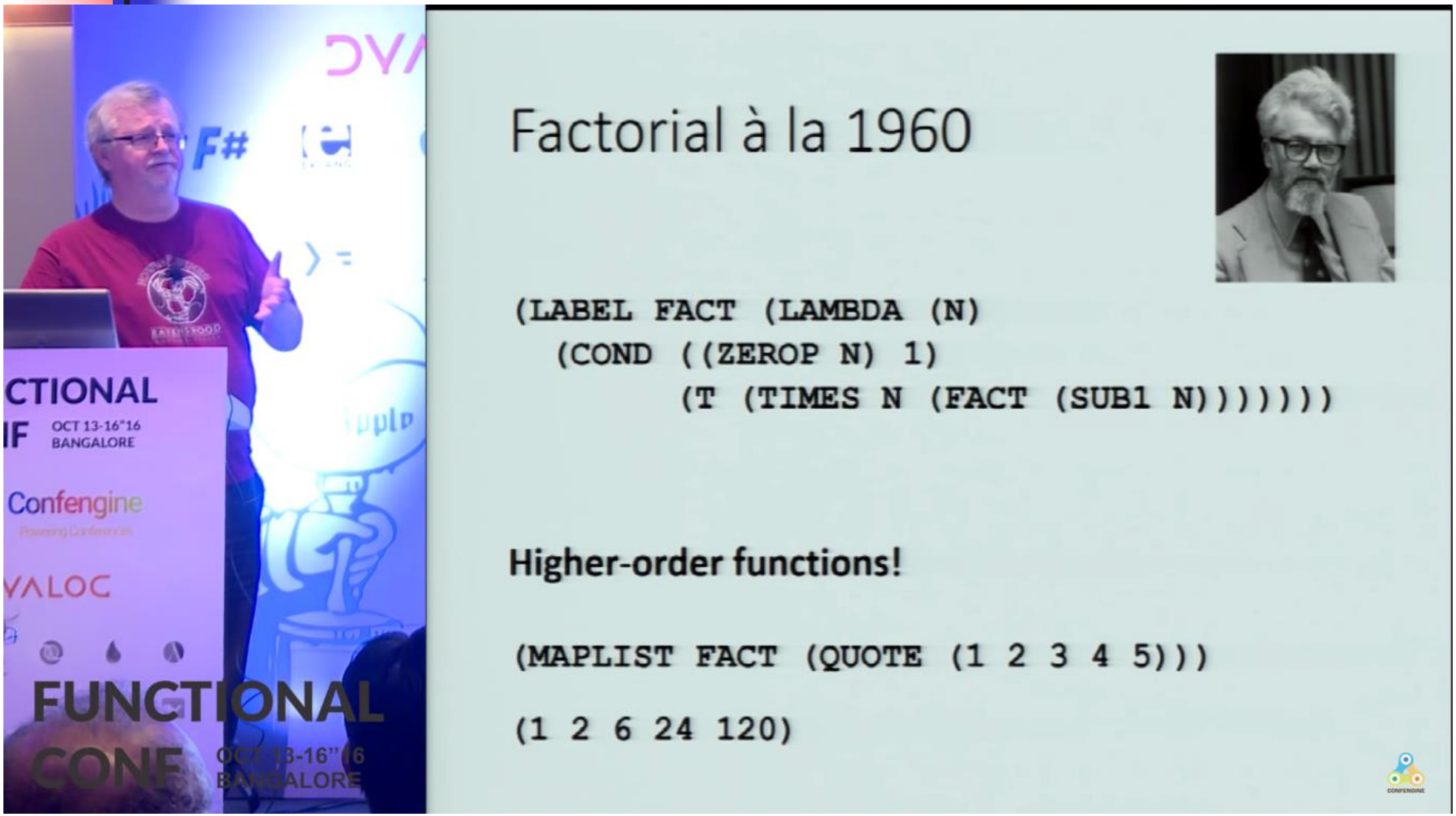

The type-checker needs a *little bit* of help
fact ::
(forall a. (a->a)->a->a) ->
(a->a) -> a -> a
FUNCTIONAL
CONF
OCT 13-16"16
BANGALORE

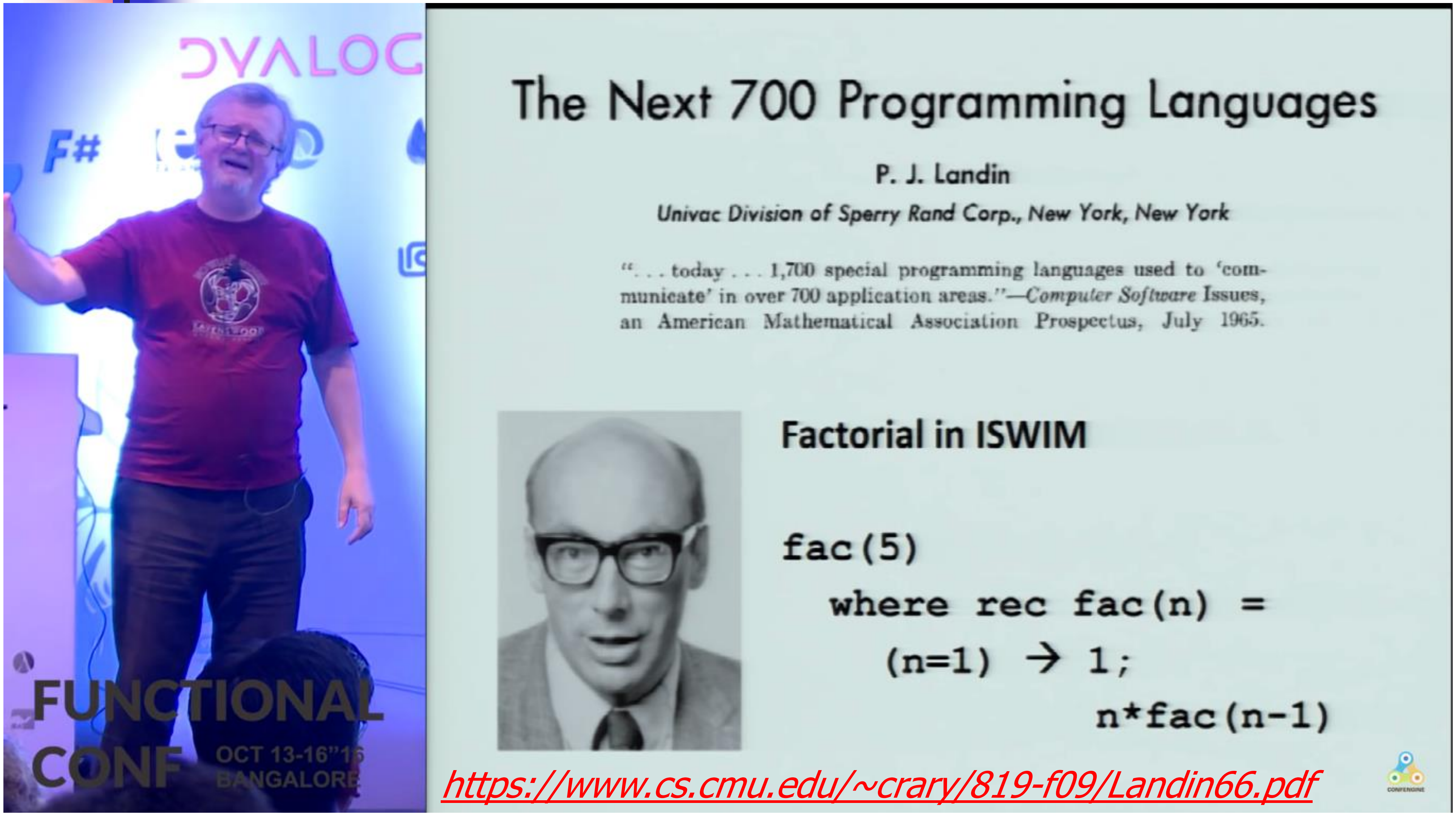
```haskell
true  x y = x
false x y = y

ifte  c t e = c t e

two   f x = f (f x)
one   f x = f x
zero  f x = x

incr n f x = f (n f x)

add   m n f x = m f (n f x)
mul   m n f x = m (n f) x

isZero n =  n (\_ -> false) true

decr n = n (\m f x -> f (m incr zero))
           zero
           (\x -> x)
           zero
```

```haskell
fact :: (forall a. (a->a)->a->a) -> (a->a) -> a -> a
fact n  =
     ifte (isZero n)
          one
          (mul n (fact (decr n)))

main =
  -- print $ (decr (add (mul two two) one)) (+1) 0
  -- print $ (fact (add (mul two two) one)) (+1) 0
  print $ (fact (add two
                  (add (mul two two) (mul two two))))
           (+1) 0

-- 3628800
-- (4.75 secs, 2,598,673,208 bytes)
```

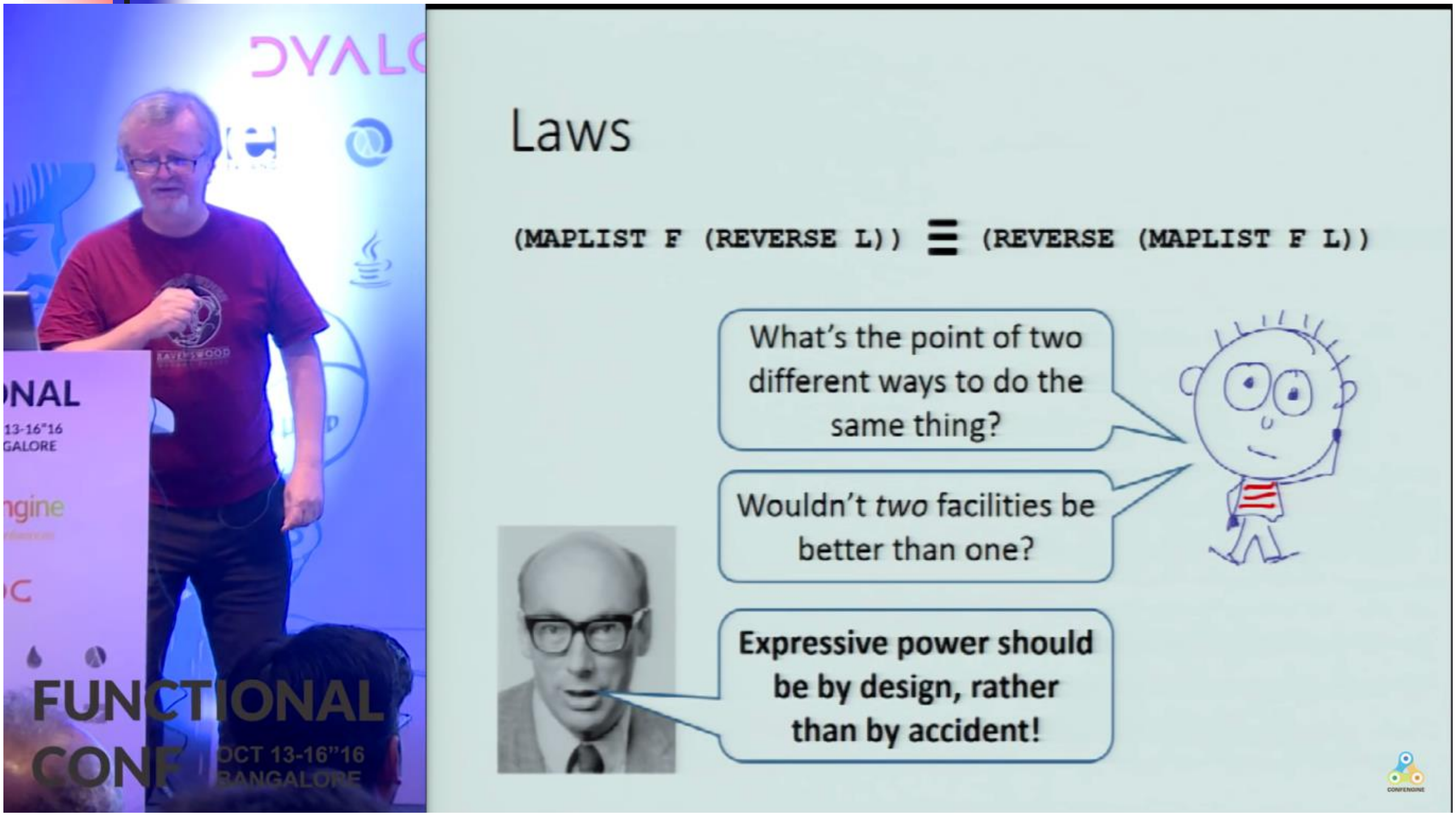
https://github.com/Funkcionalne/Prednasky/blob/master/01/src/Church.hs

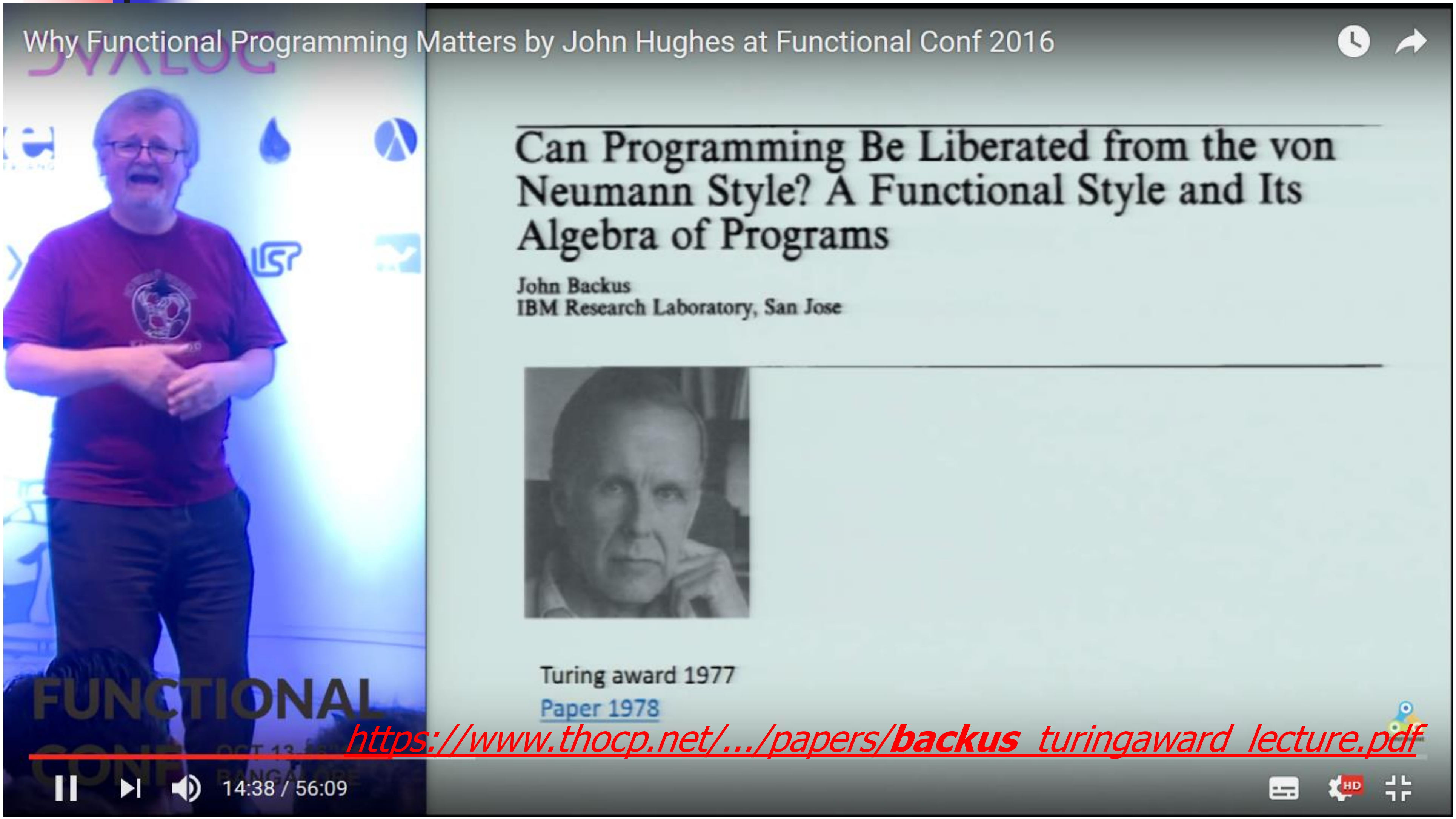
Lisp


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)
Confengine
FUNCTIONAL

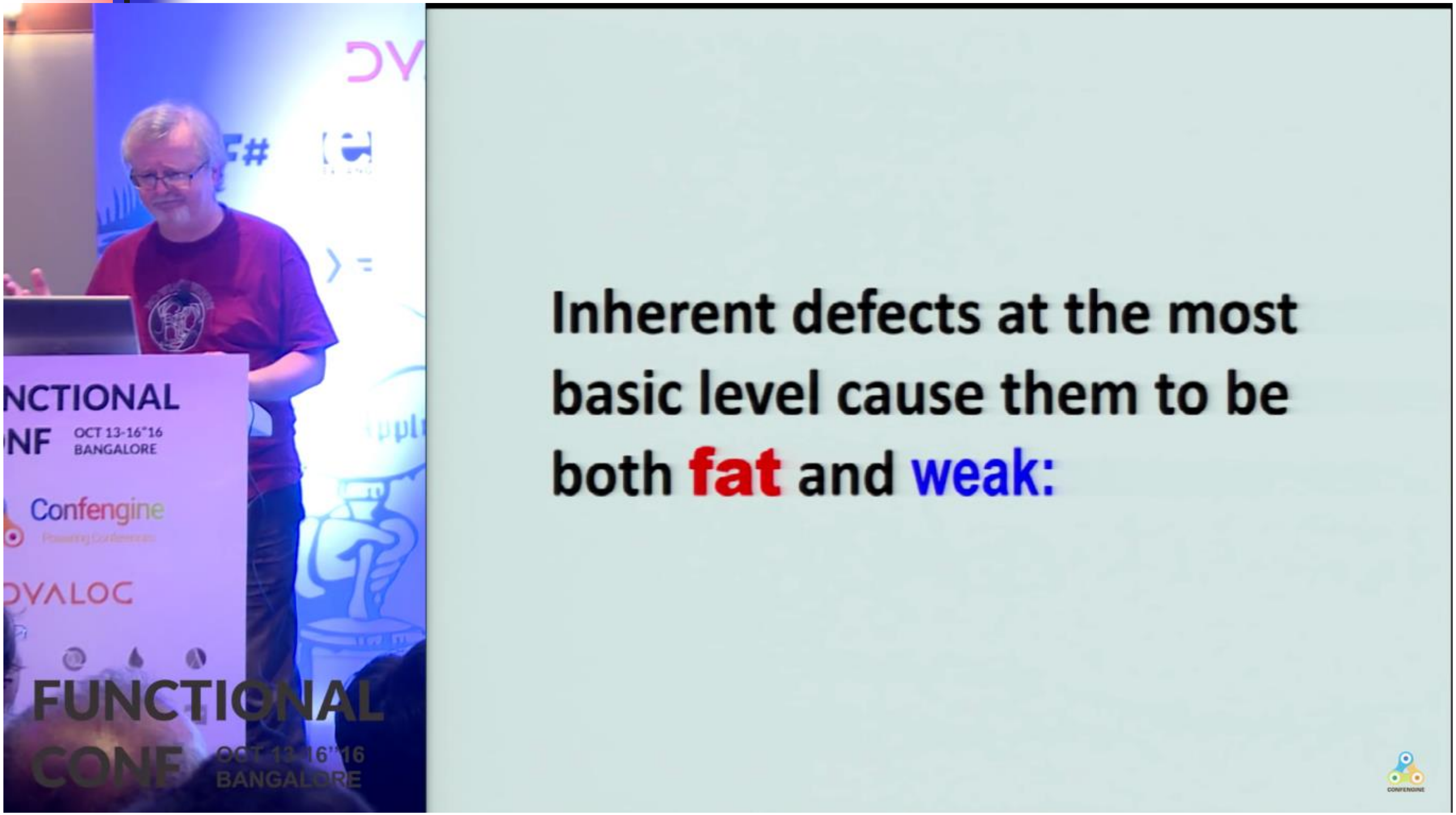


https://www.cs.cmu.edu/~crary/819-f09/Landin66.pdf

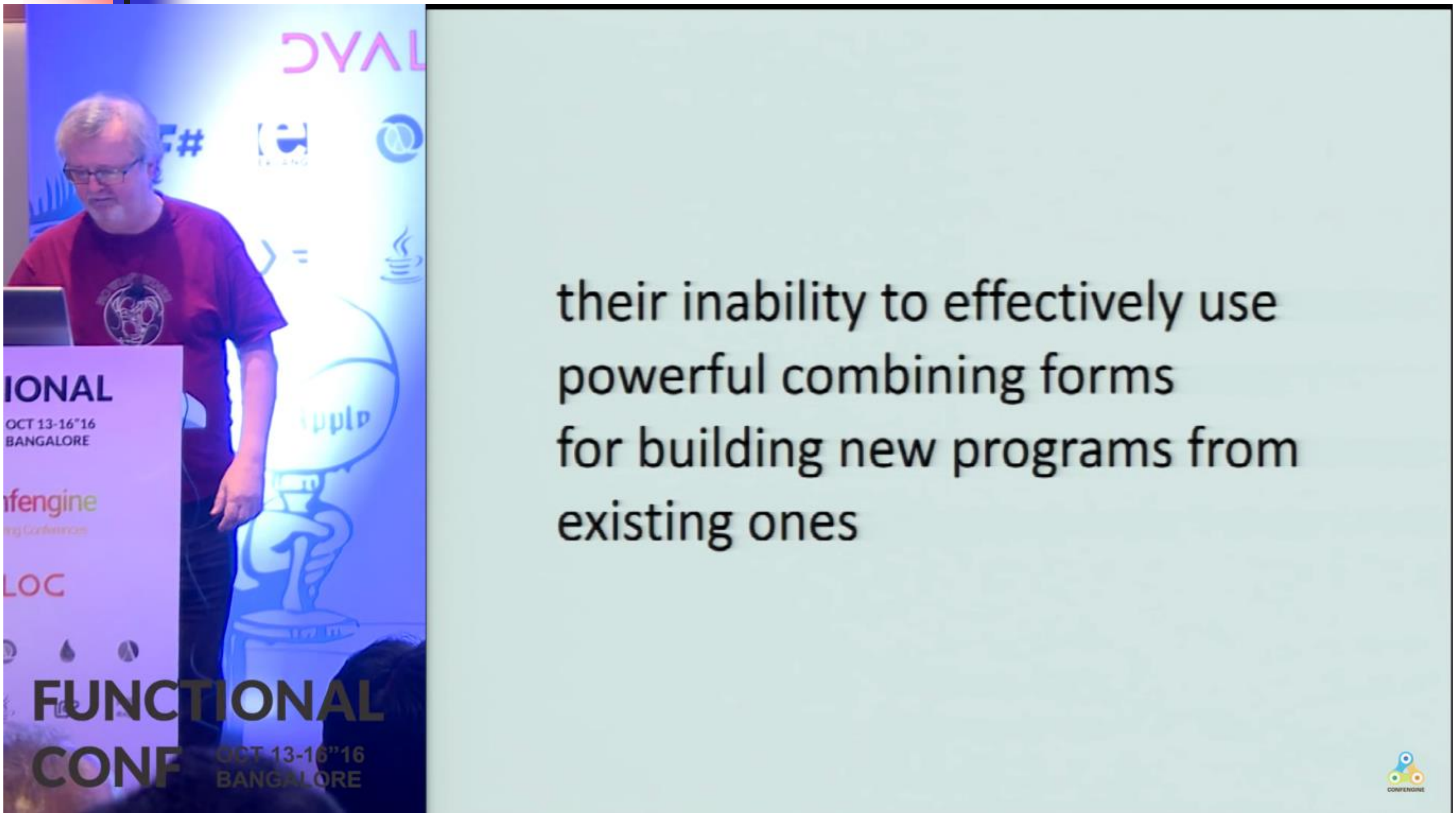

Laws
(MAPLIST F (REVERSE L)) ≡ (REVERSE (MAPLIST F L))
What's the point of two different ways to do the same thing?
Wouldn't *two* facilities be better than one?
Expressive power should be by design, rather than by accident!

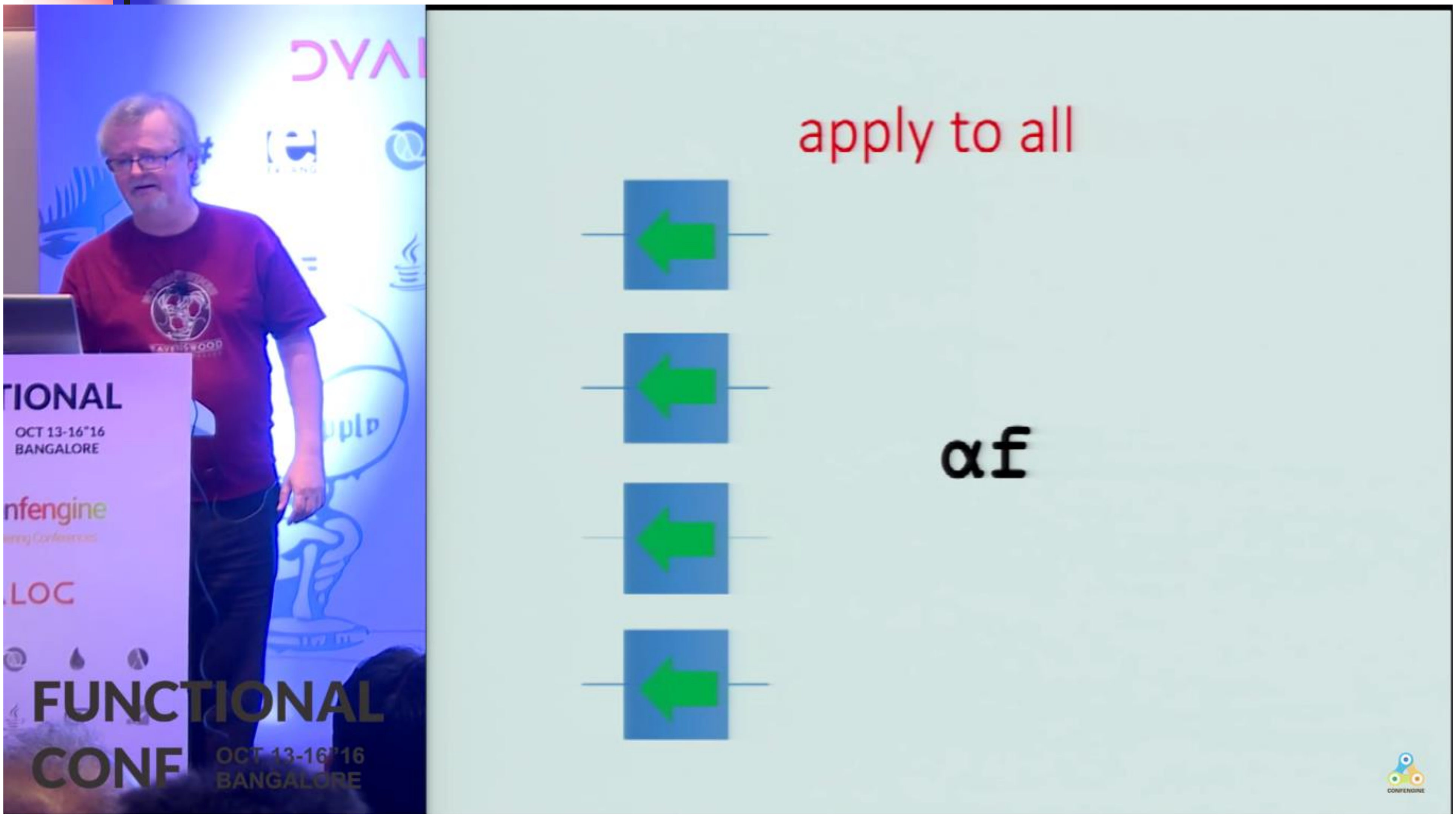

Why Functional Programming Matters by John Hughes at Functional Conf 2016
Can Programming Be Liberated from the von Neumann Style? A Functional Style and Its Algebra of Programs
John Backus
IBM Research Laboratory, San Jose
Turing award 1977
Paper 1978
FUNCTIONAL
14:38 / 56:09
https://www.thocp.net/.../papers/**backus**_turingaward_lecture.pdf

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

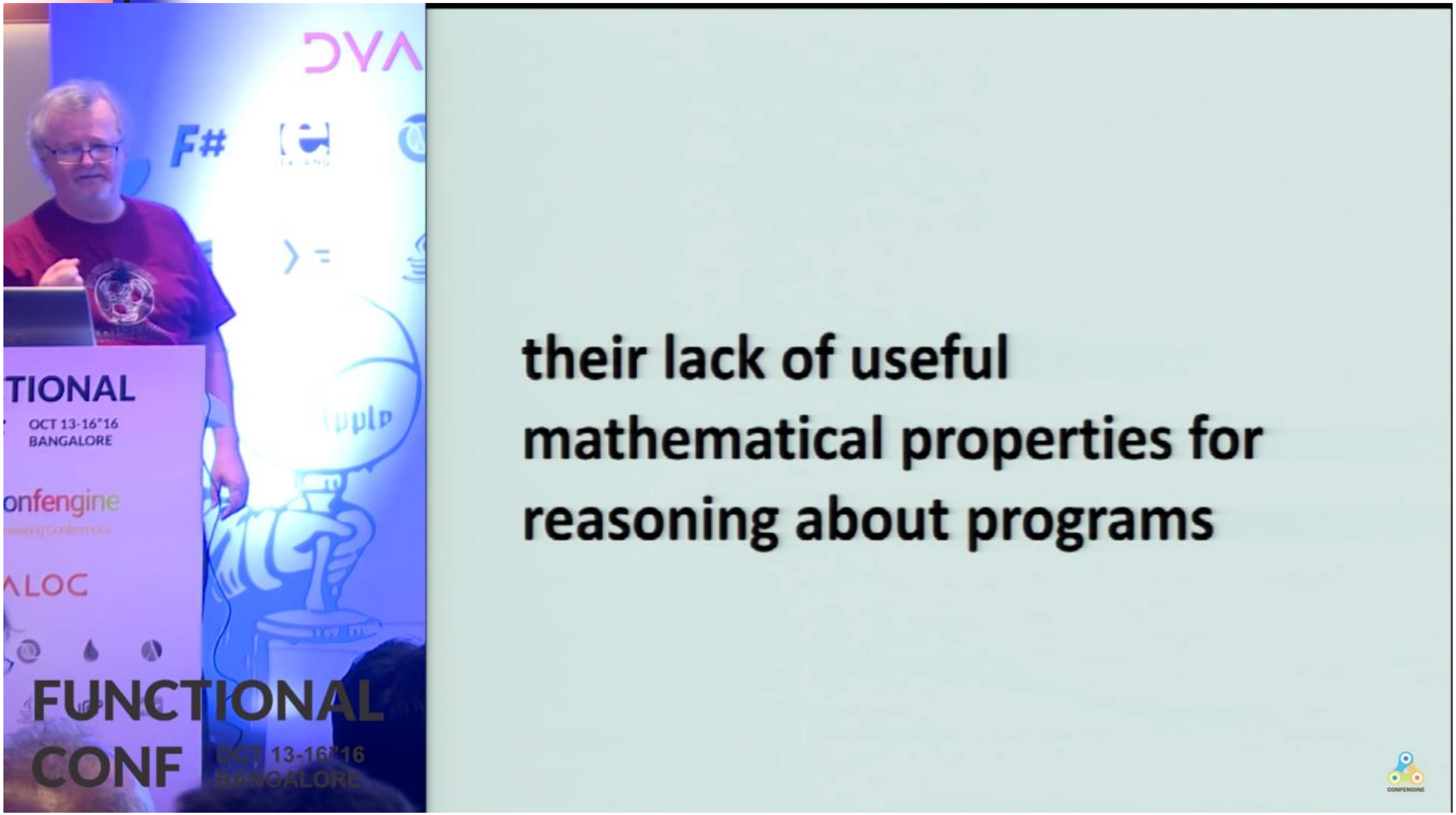

Inherent defects at the most basic level cause them to be both **fat** and **weak:**
NCTIONAL
NF OCT 13-16"16 BANGALORE
Confengine
FUNCTIONAL
CONF OCT 13-16"16 BANGALORE


DYALOG
Word-at-a-time
FUNCTIONAL
CONF
OCT 13-16"16
BANGALORE

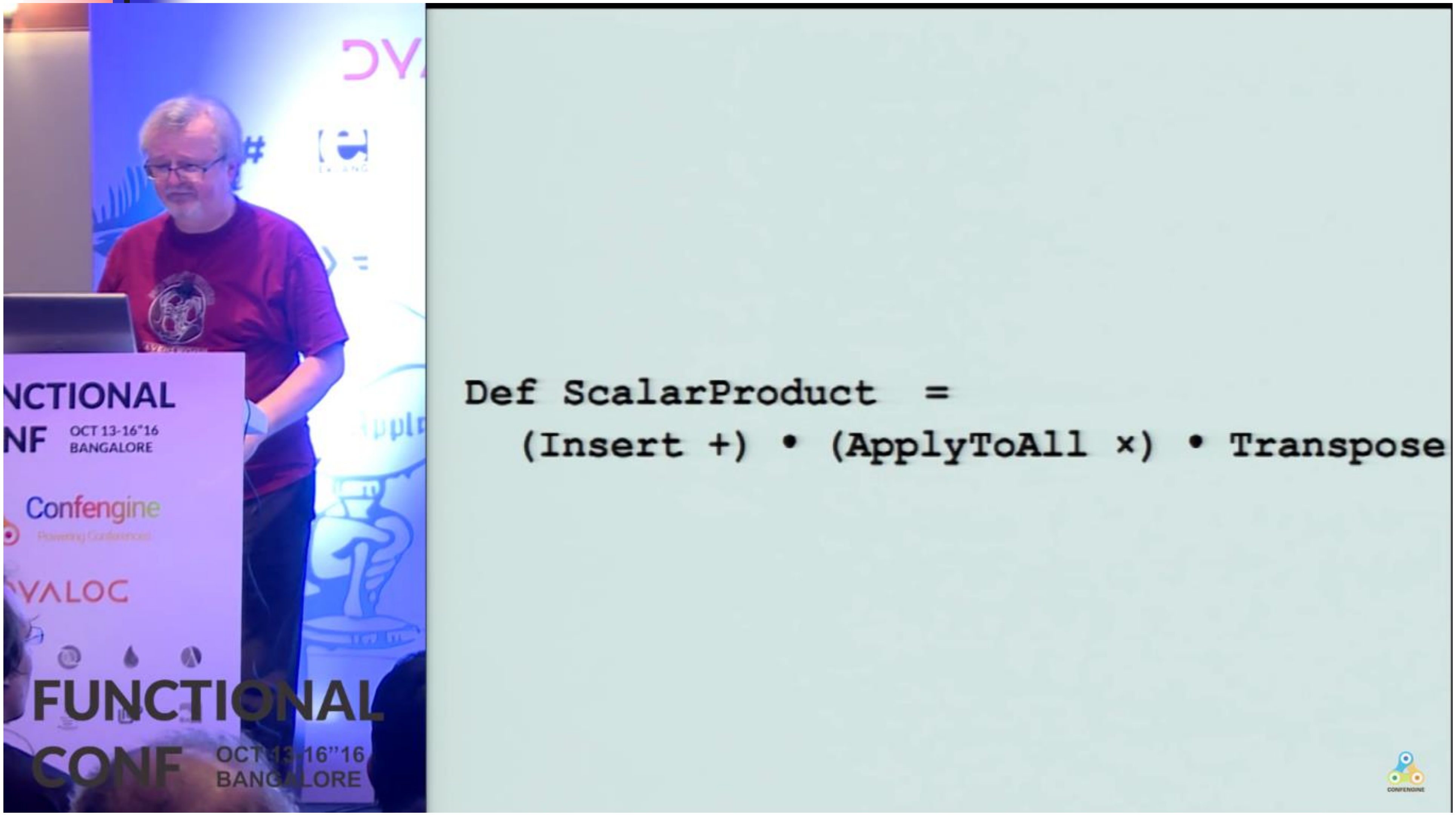

their inability to effectively use
powerful combining forms
for building new programs from
existing ones
FUNCTIONAL
CONF
OCT 13-16"16
BANGALORE
IONAL
OCT 13-16"16
BANGALORE
F#


apply to all
αf
FUNCTIONAL
CONF
OCT 13-16 '16
BANGALORE

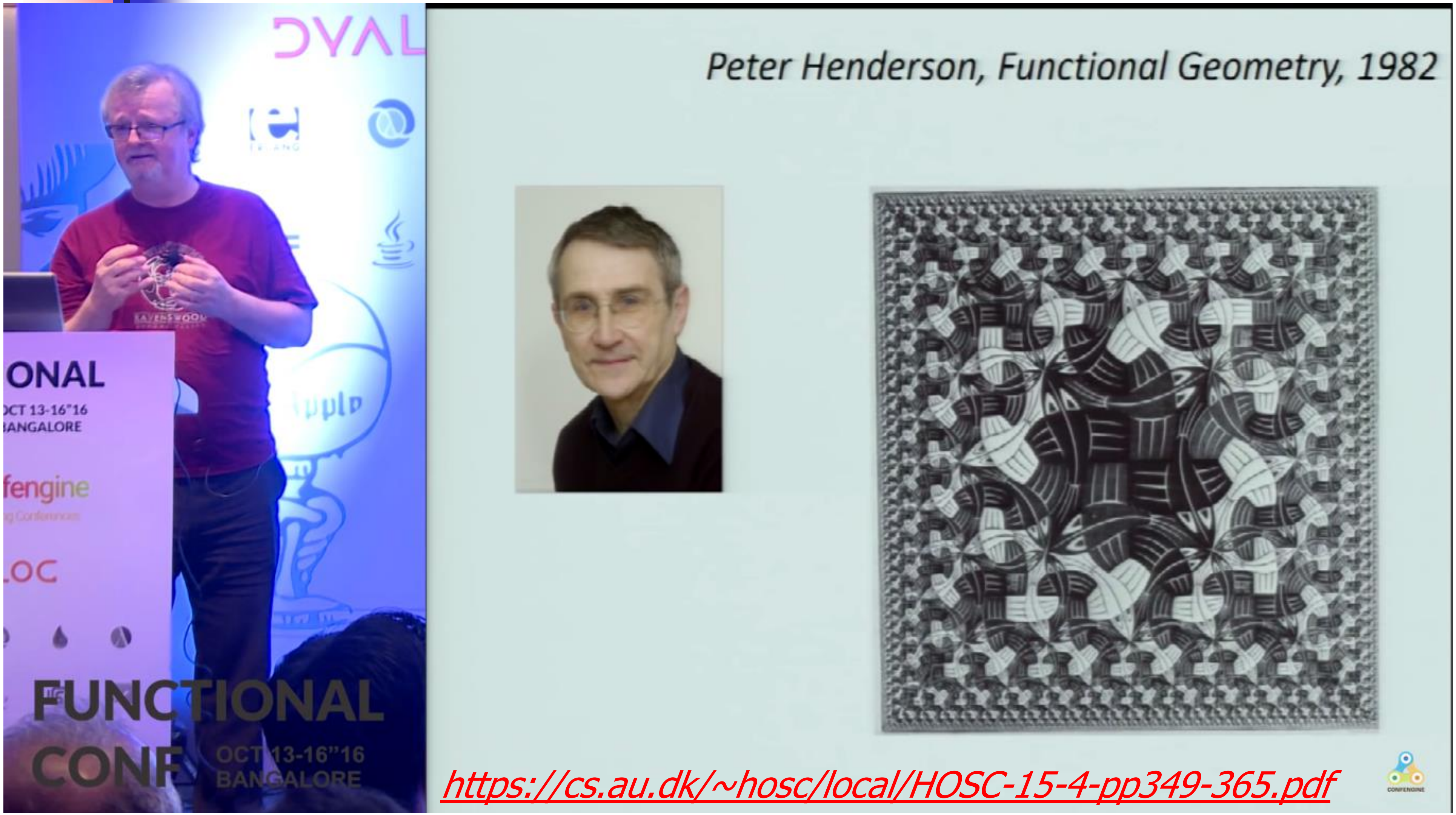
kombinátory

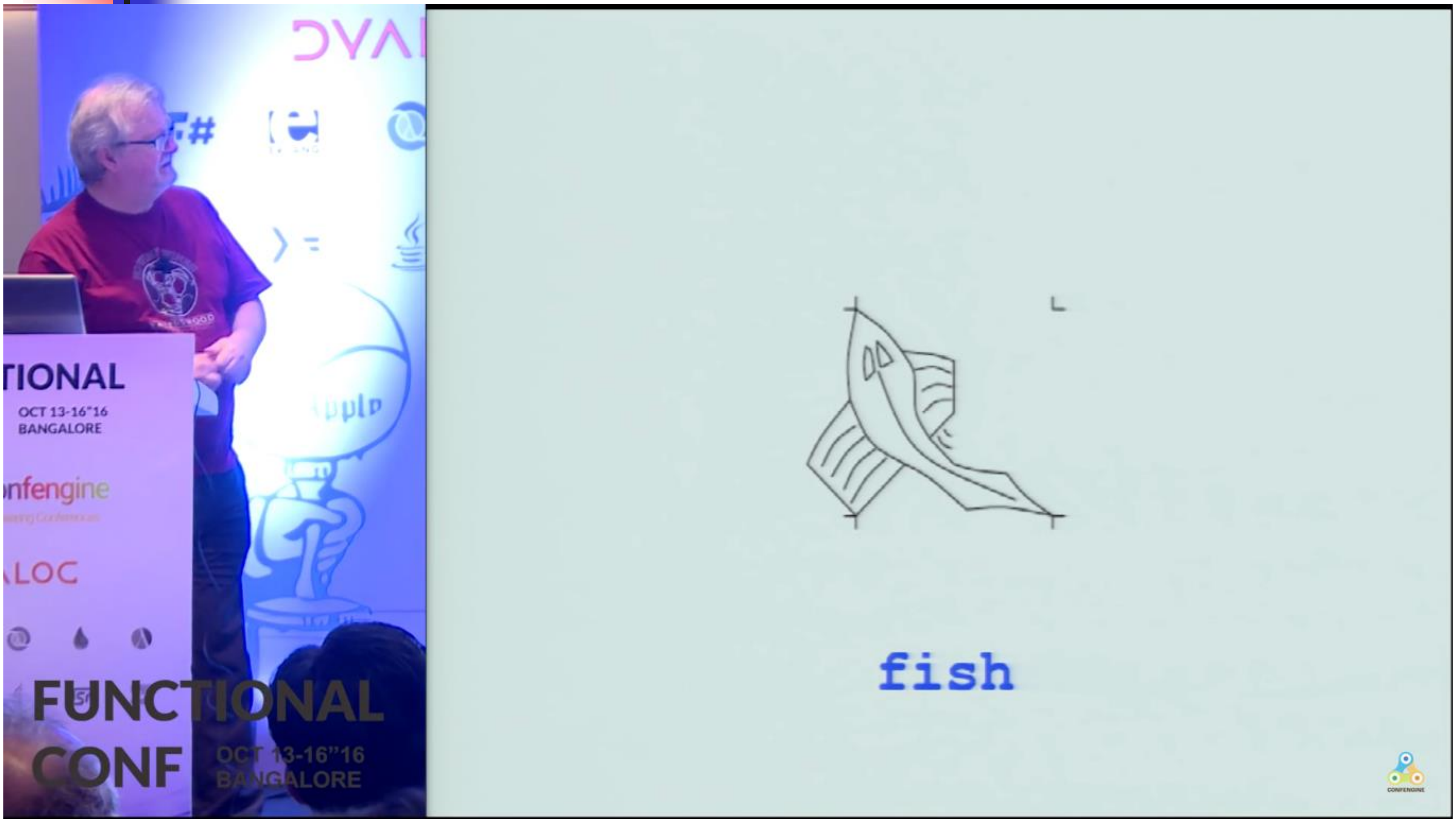

their lack of useful mathematical properties for reasoning about programs
TIONAL
OCT 13-16'16
BANGALORE
onfengine
FUNCTIONAL
CONF
OCT 13-16'16
BANGALORE
F#

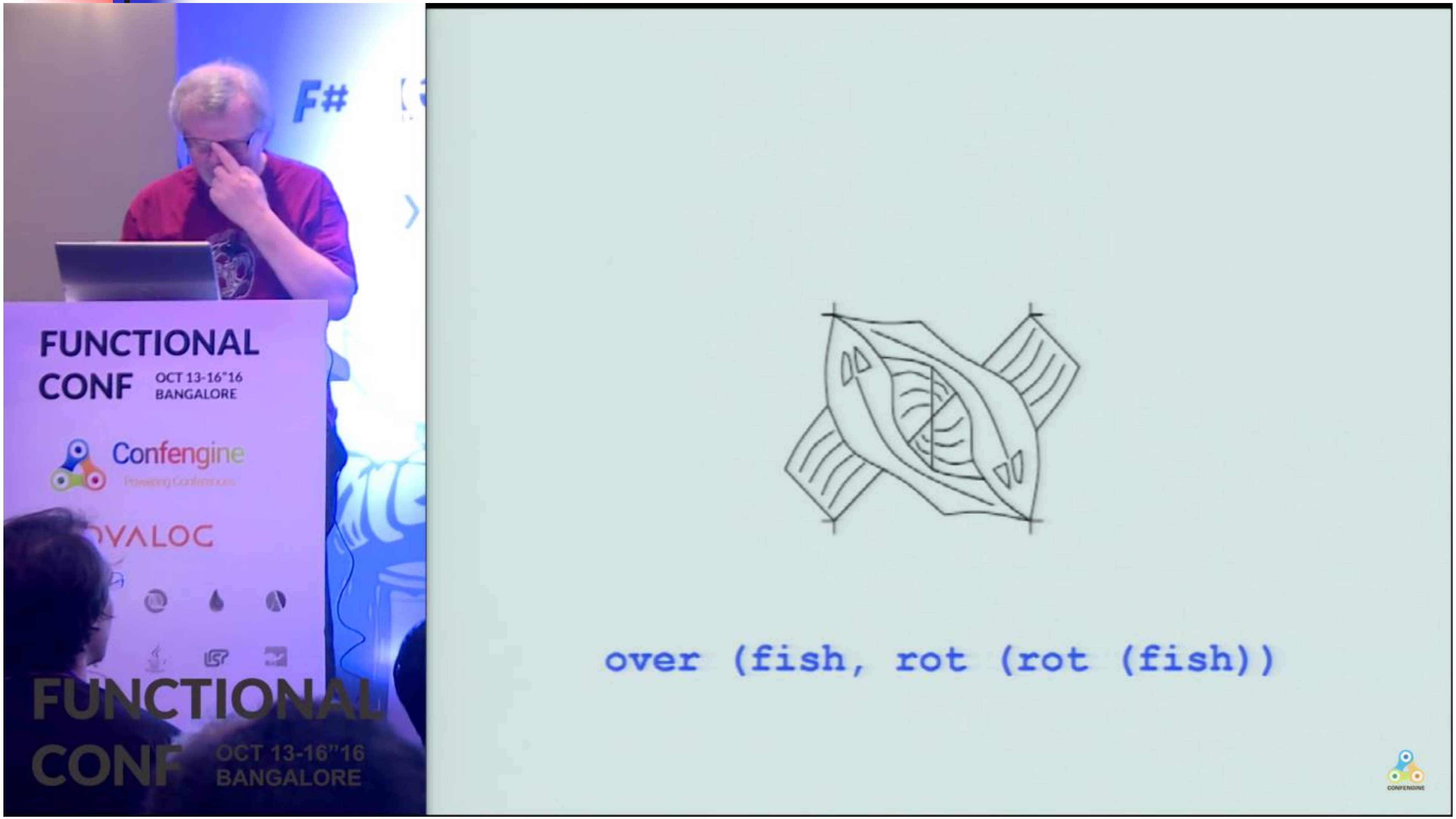
kombinátory

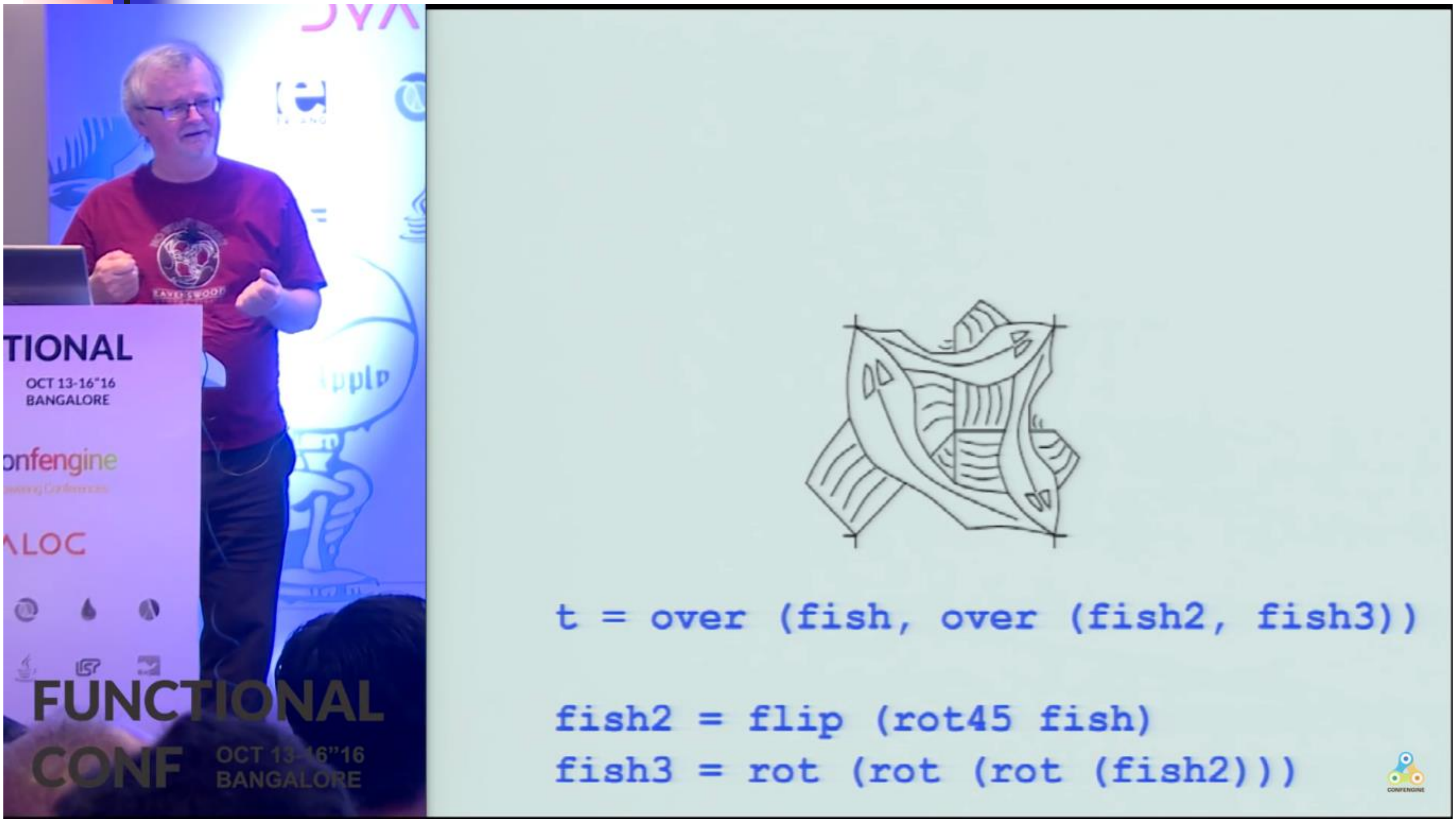

Def ScalarProduct =
(Insert +) • (ApplyToAll ×) • Transpose
FUNCTIONAL
CONF
OCT 13-16"16
BANGALORE
Confengine

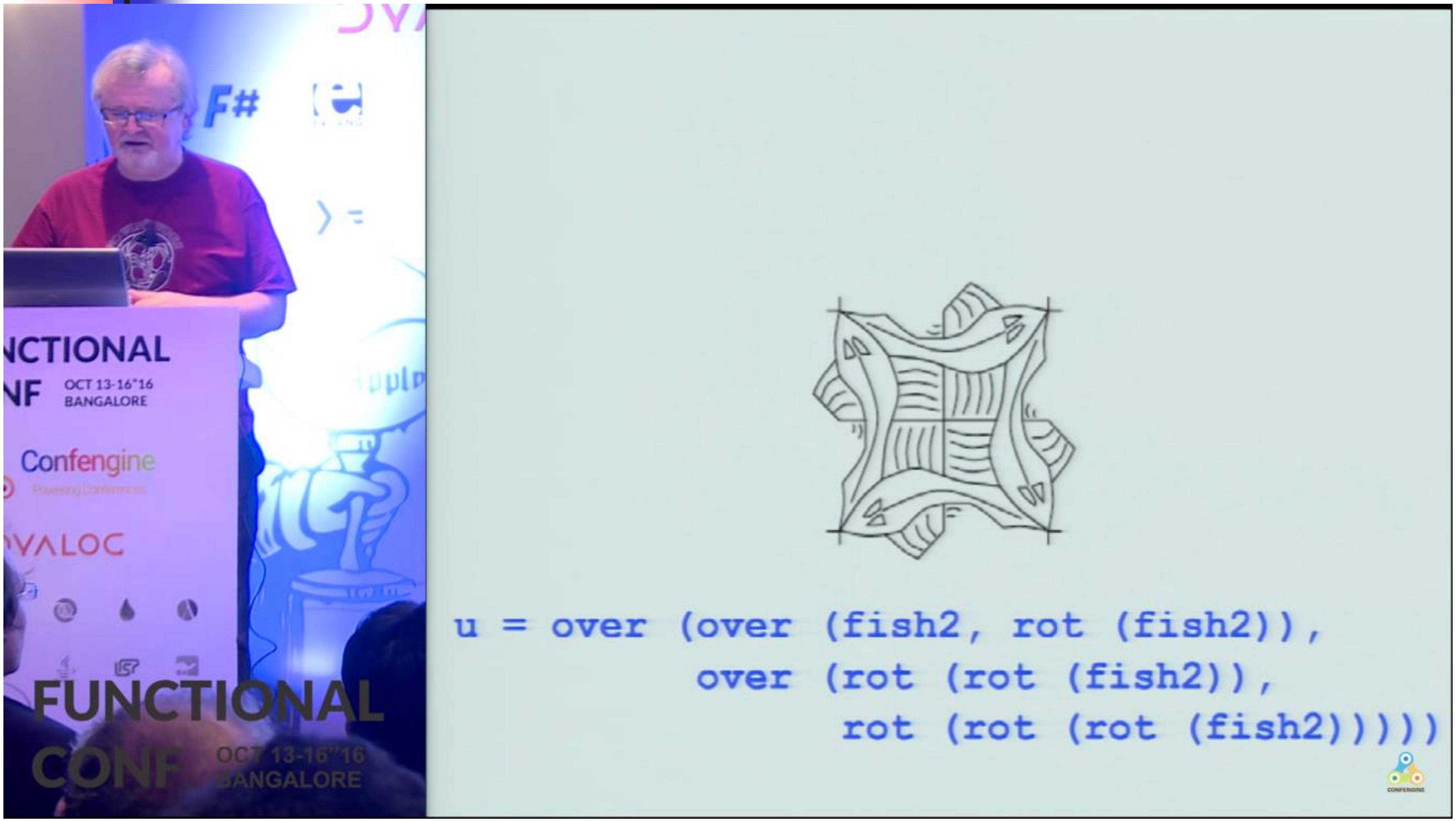
Def SP = (/ +) • (α ×) • Trans

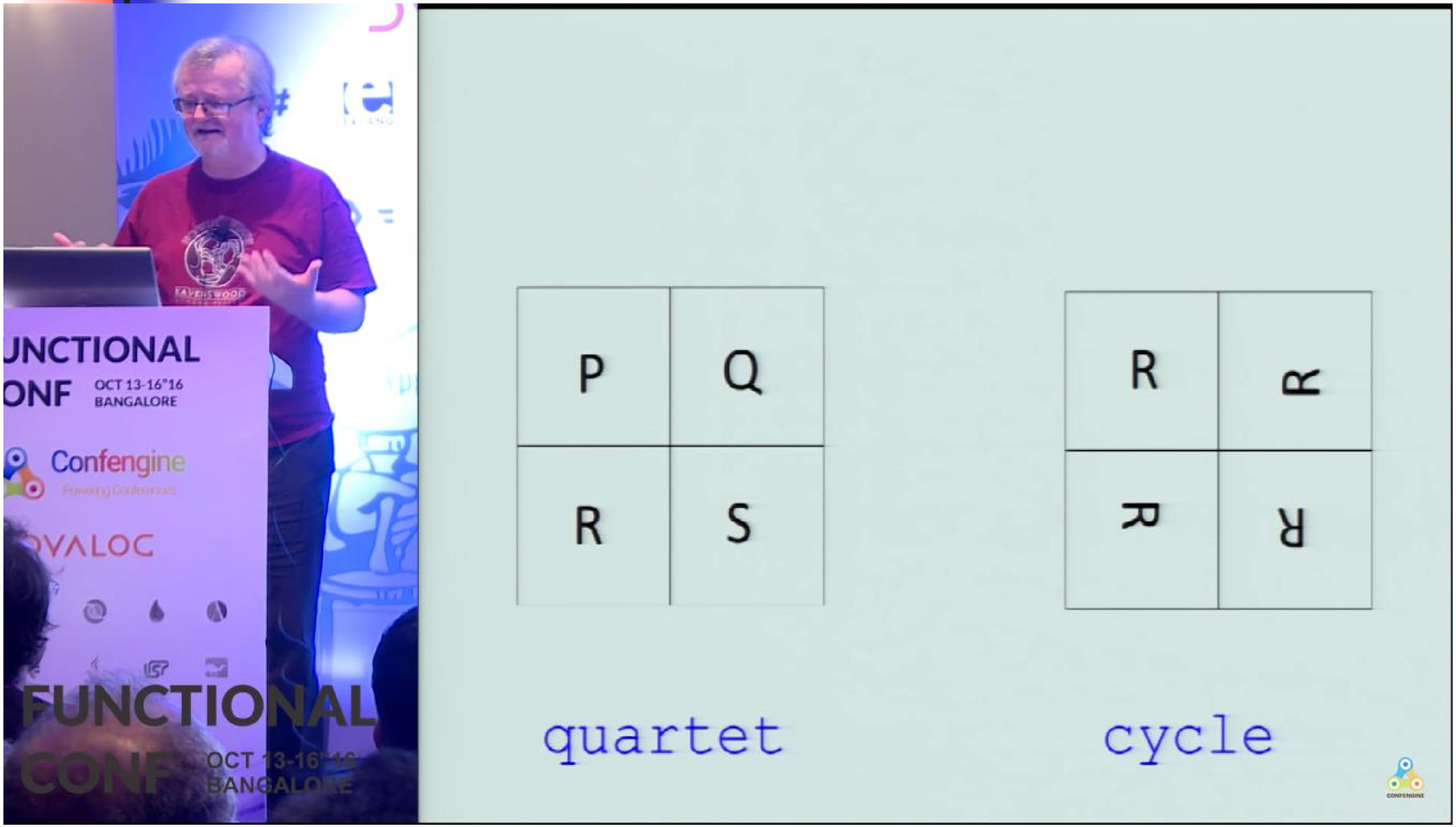

Peter Henderson, Functional Geometry, 1982
https://cs.au.dk/~hosc/local/HOSC-15-4-pp349-365.pdf
FUNCTIONAL
CONF
OCT 13-16"16
BANGALORE

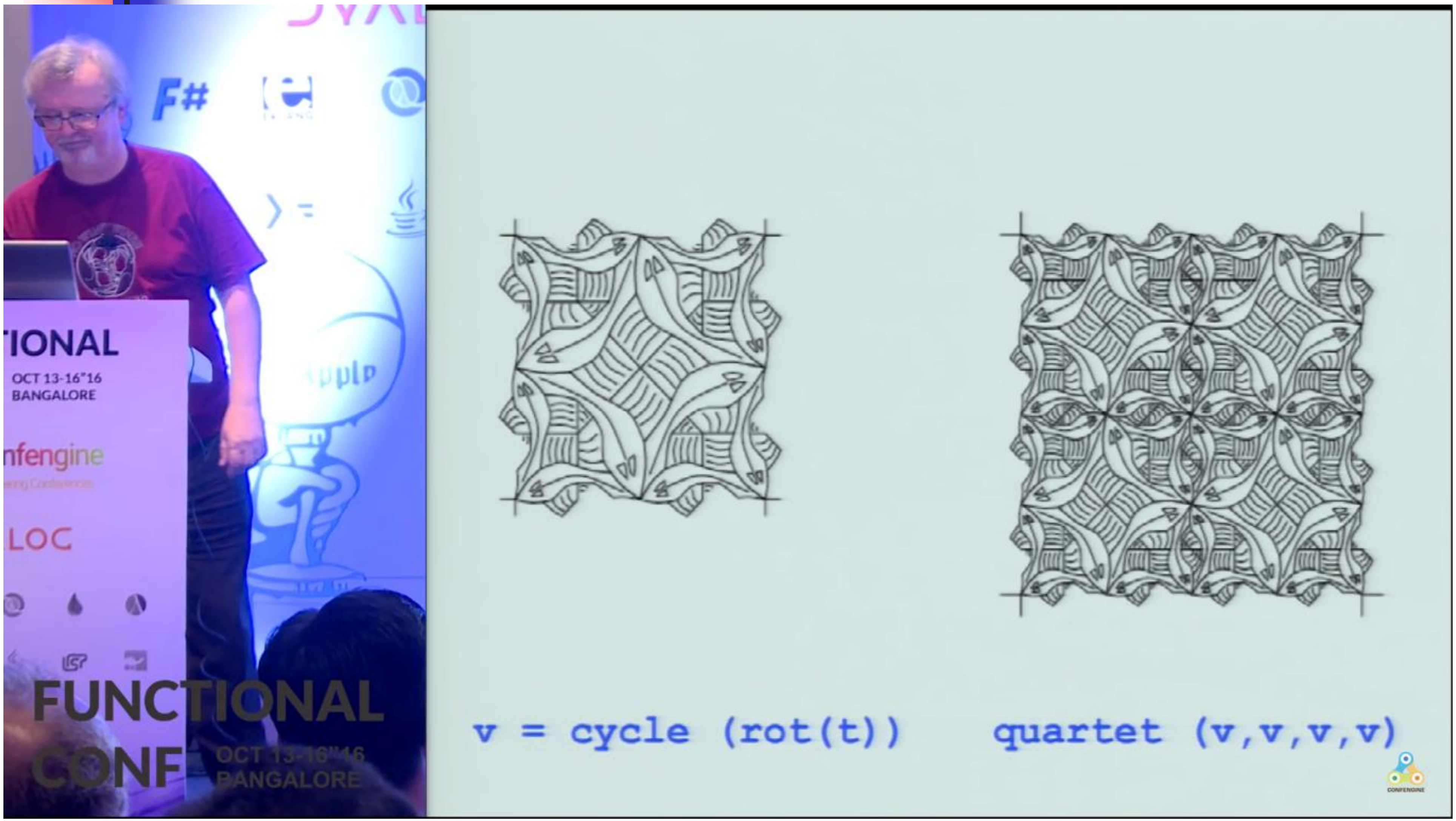

TIONAL
OCT 13-16"16
BANGALORE
FUNCTIONAL
CONF
OCT 13-16"16
BANGALORE
fish

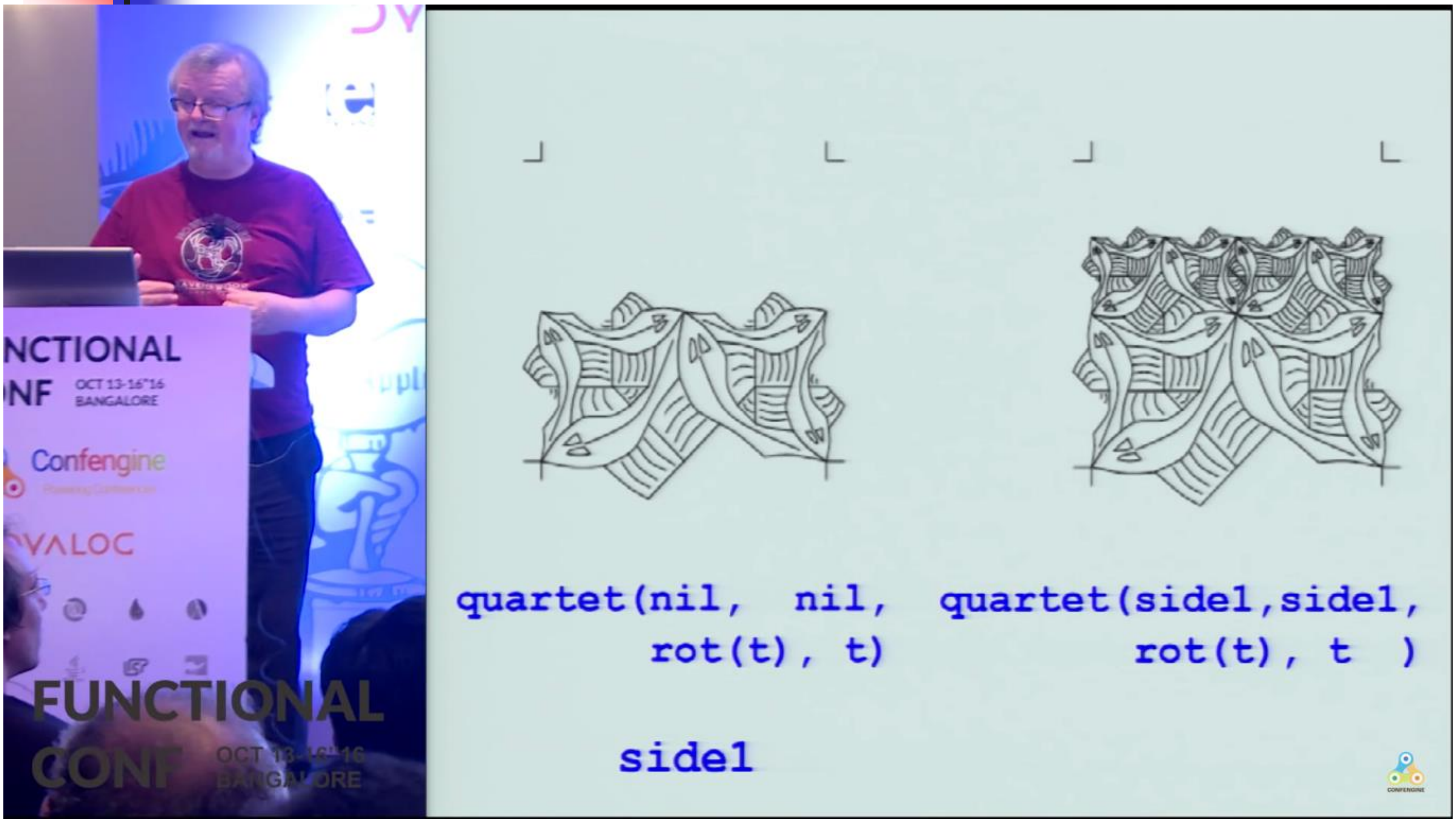

F#
FUNCTIONAL
CONF
OCT 13-16"16
BANGALORE
Confengine
over (fish, rot (rot (fish))
FUNCTIONAL
CONF
OCT 13-16"16
BANGALORE

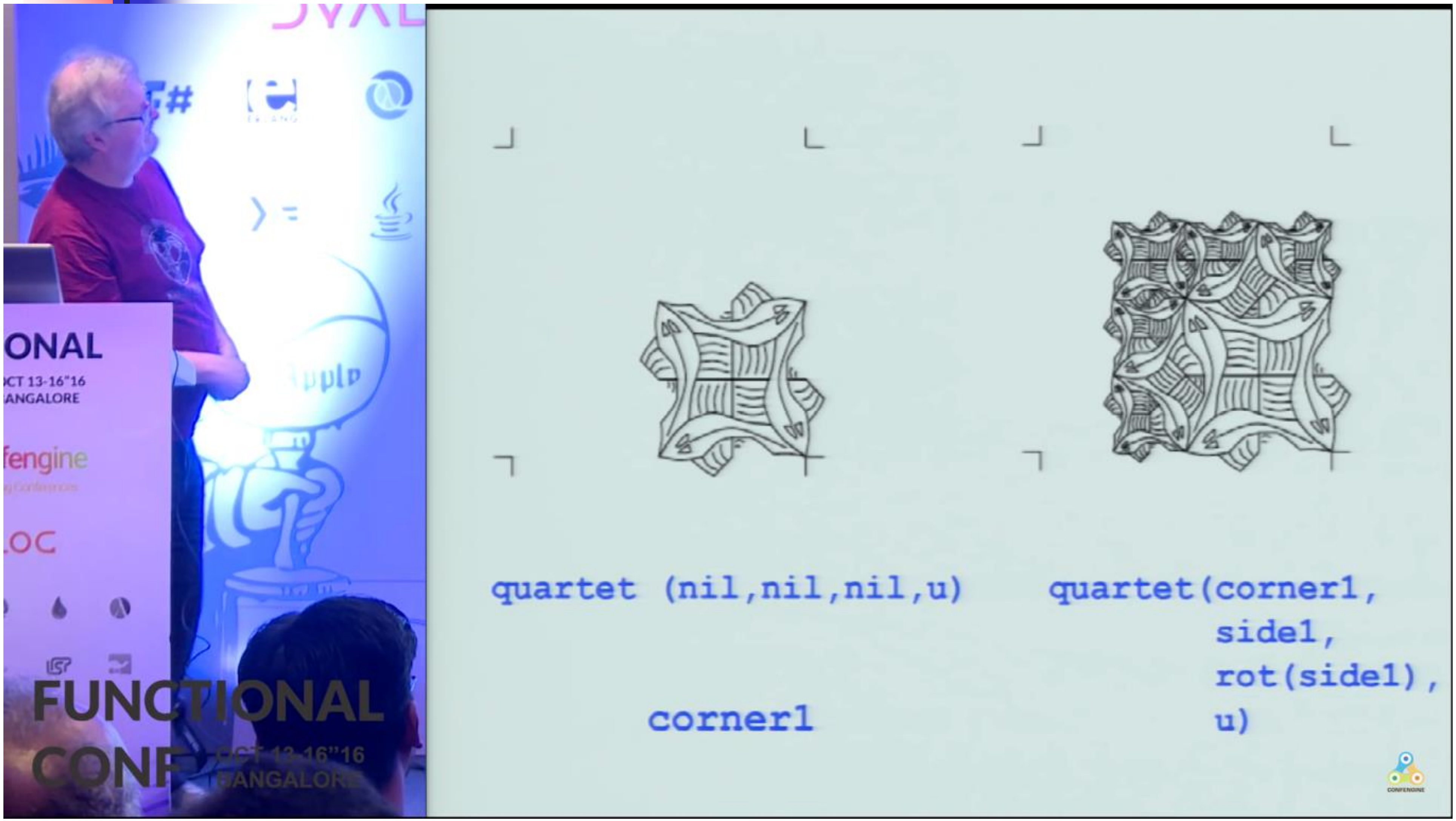

t = over (fish, over (fish2, fish3))
fish2 = flip (rot45 fish)
fish3 = rot (rot (rot (fish2)))
FUNCTIONAL
CONF
TIONAL
OCT 13-16"16
BANGALORE

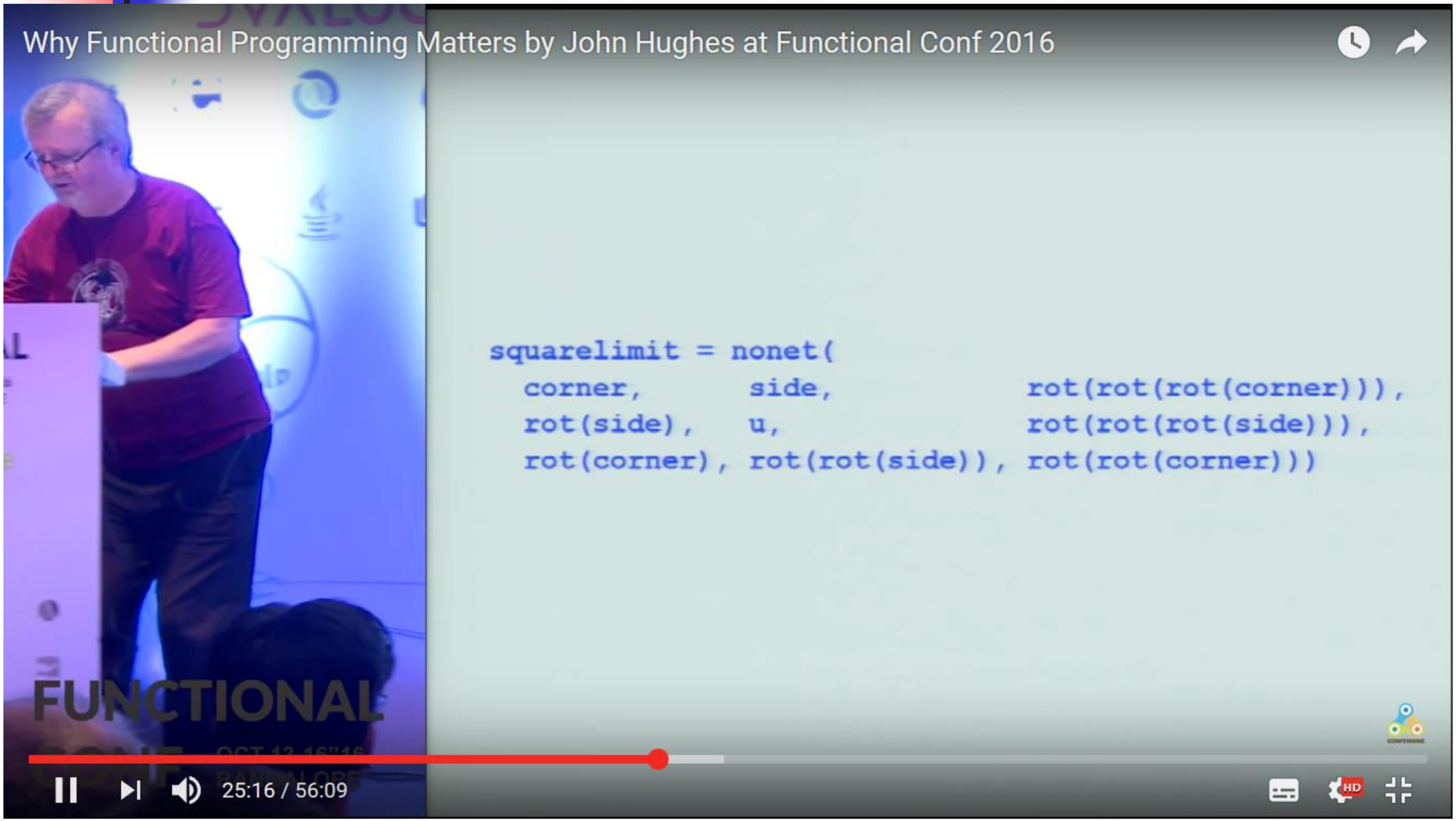

F#
NCTIONAL
NF OCT 13-16"16 BANGALORE
Confengine
VALOC
FUNCTIONAL
CONF OCT 13-16"16 BANGALORE
u = over (over (fish2, rot (fish2)),
over (rot (rot (fish2)),
rot (rot (rot (fish2)))))

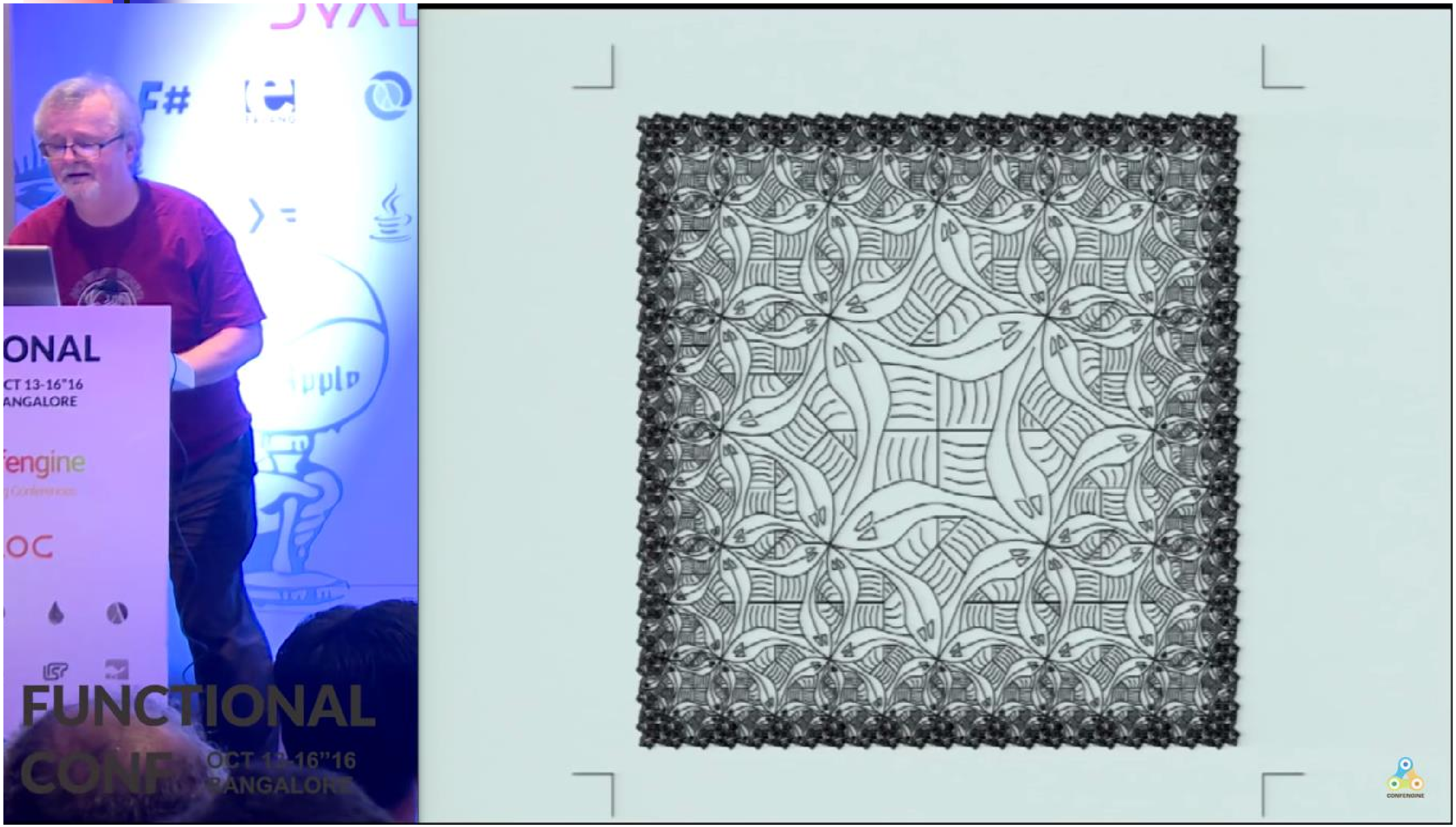

P
Q
R
S
R
quartet
cycle
UNCTIONAL
ONF
OCT 13-16"16
BANGALORE
Confengine
FUNCTIONAL
CONF


v = cycle (rot(t))
quartet (v,v,v,v)
OCT 13-16"16
BANGALORE
FUNCTIONAL
CONF

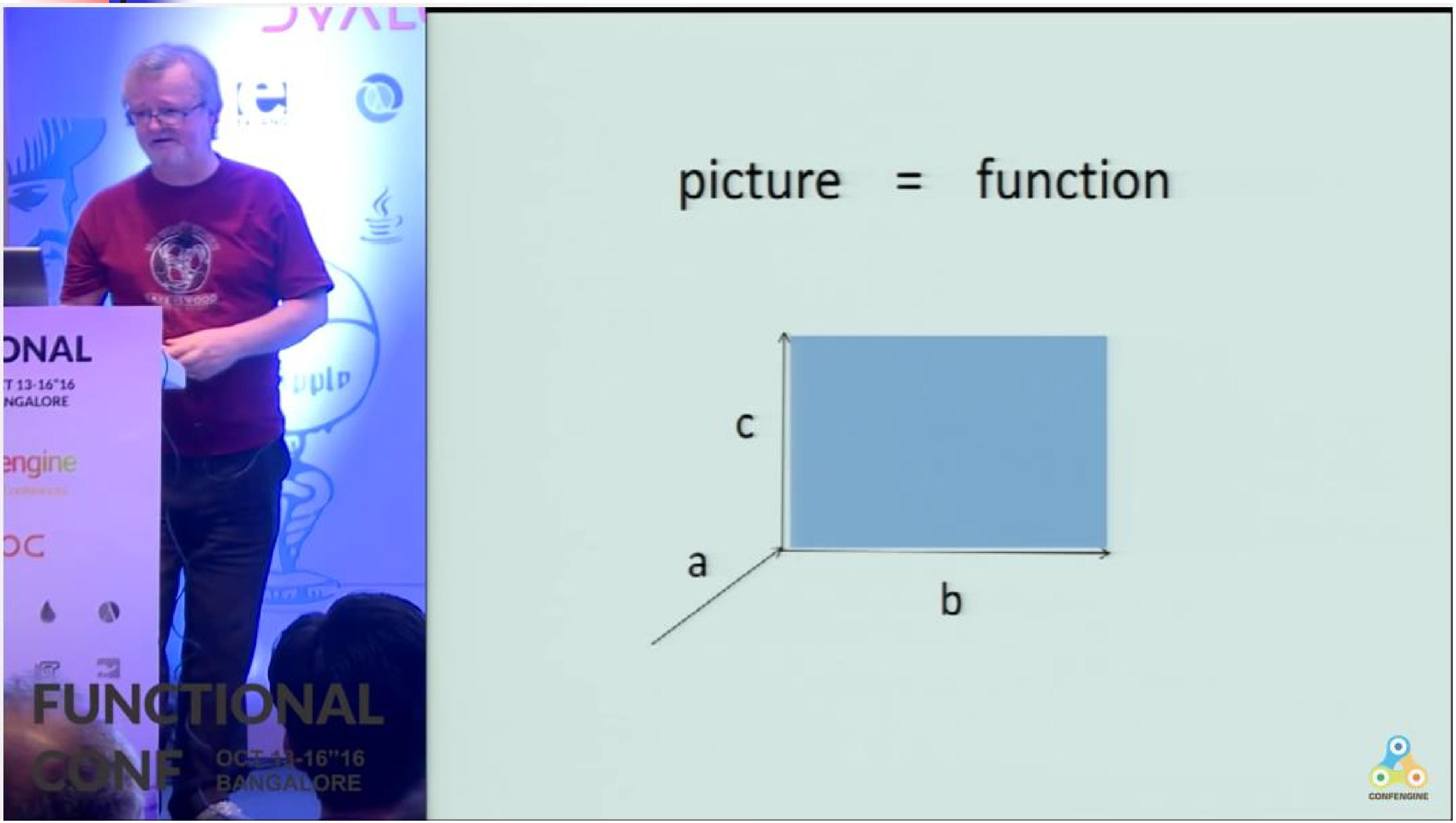

quartet(nil, nil,
rot(t), t)
quartet(side1,side1,
rot(t), t)
side1
FUNCTIONAL
CONF
OCT 13-16'16
BANGALORE
Confengine

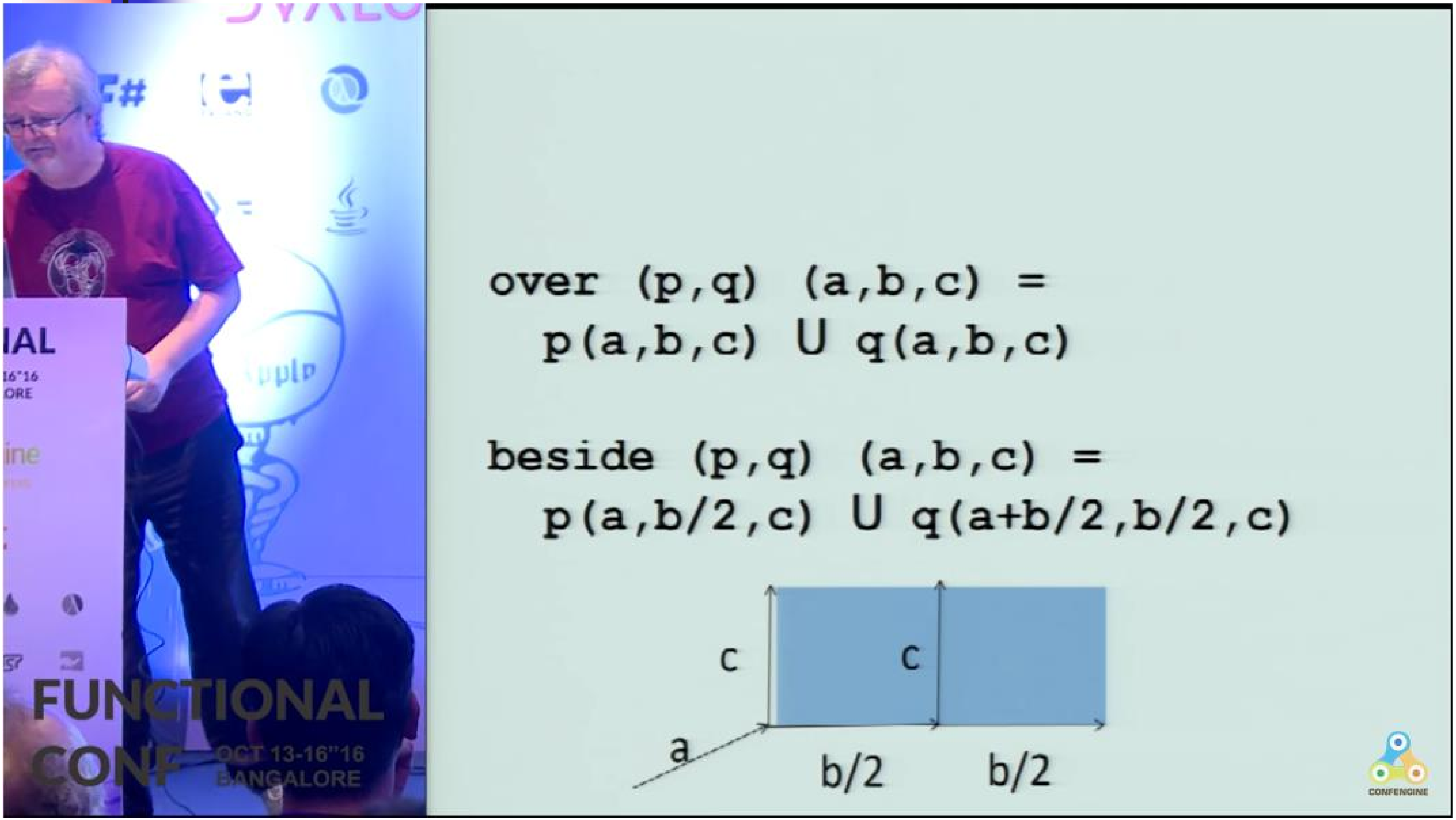

quartet (nil,nil,nil,u)
corner1
quartet(corner1,
side1,
rot(side1),
u)
FUNCTIONAL
CONF
OCT 13-16"16
BANGALORE

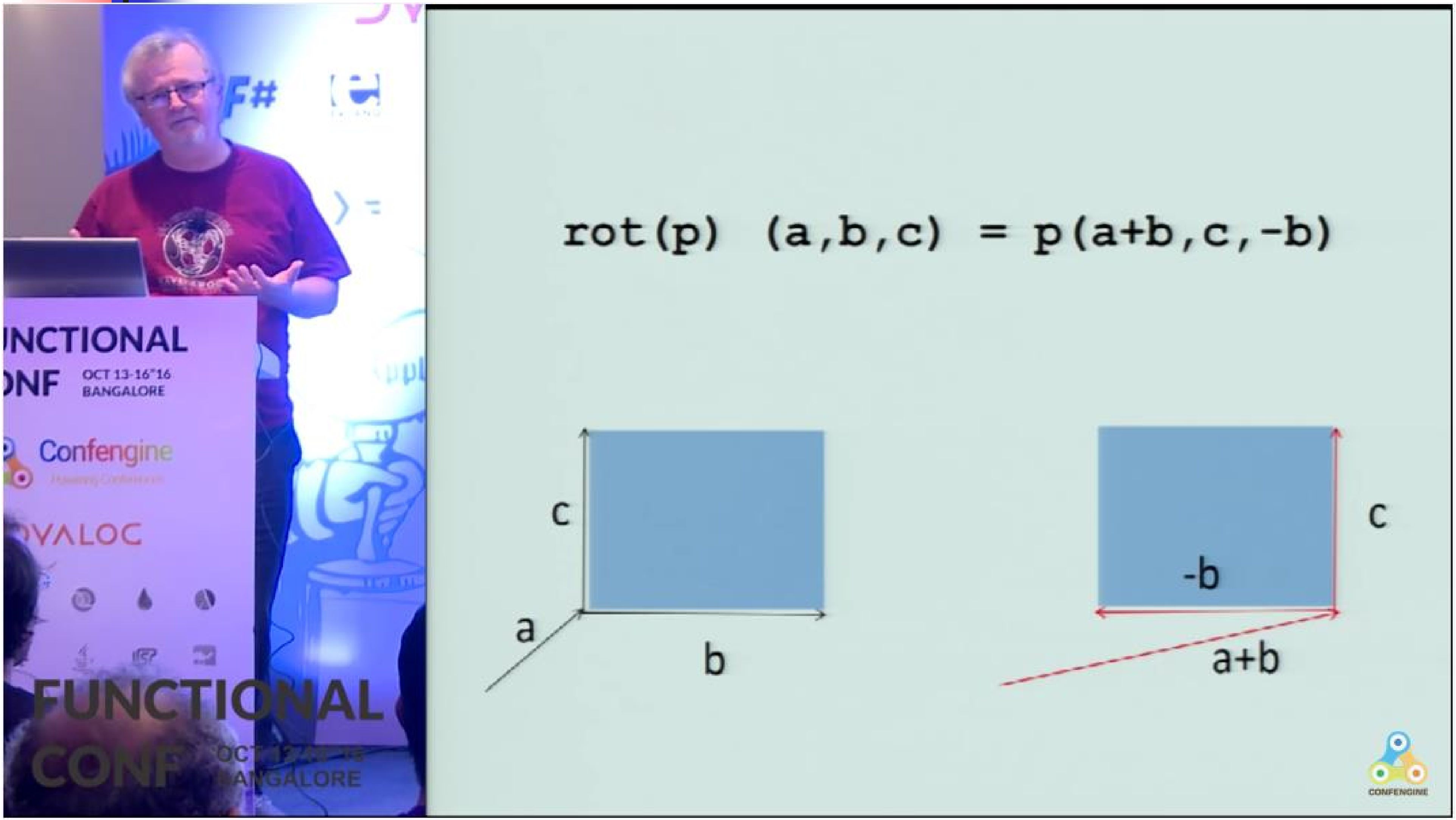

Why Functional Programming Matters by John Hughes at Functional Conf 2016
squarelimit = nonet(
corner, side, rot(rot(rot(corner))),
rot(side), u, rot(rot(rot(side))),
rot(corner), rot(rot(side)), rot(rot(corner)))
FUNCTIONAL
25:16 / 56:09



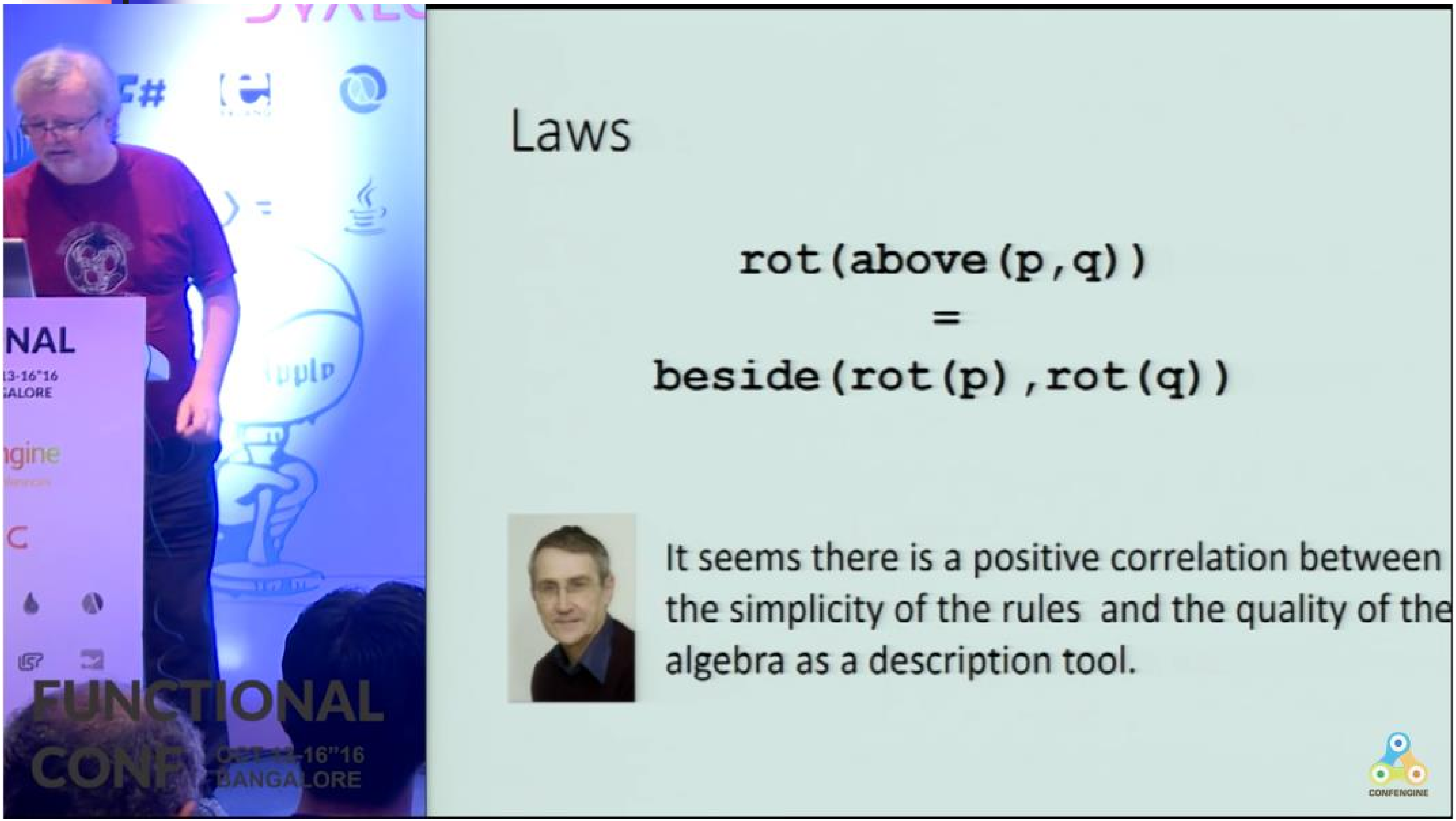
F#
ONAL
CT 13-16"16
ANGALORE
fengine
FUNCTIONAL
CONF
OCT 13-16"16
BANGALORE
CONFENGINE

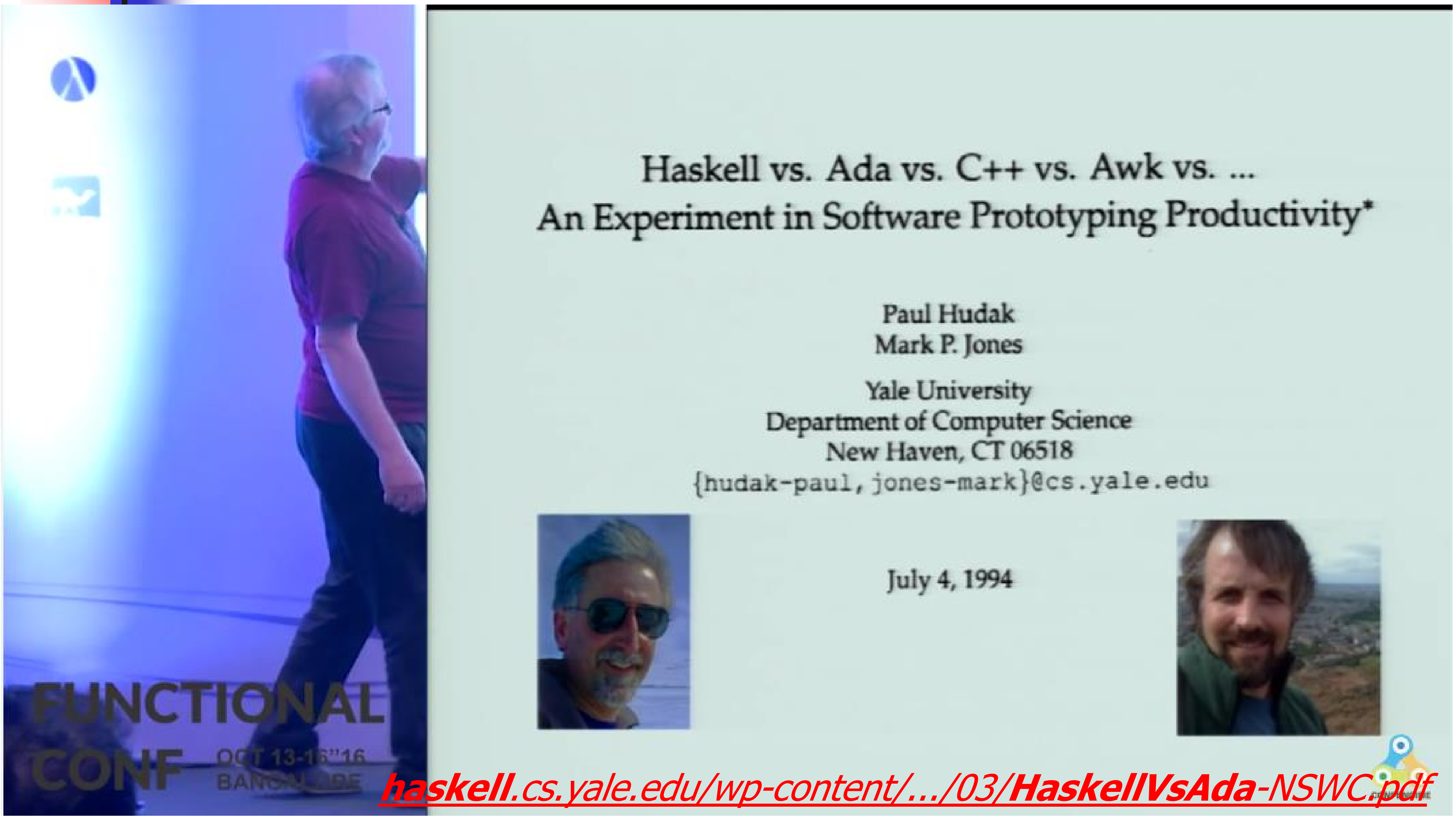

picture = function
c
a
b
FUNCTIONAL
CONF
OCT 13-16''16
BANGALORE
CONFENGINE


over (p,q) (a,b,c) =
p(a,b,c) U q(a,b,c)
beside (p,q) (a,b,c) =
p(a,b/2,c) U q(a+b/2,b/2,c)
c
c
a
b/2
b/2
FUNCTIONAL
CONF
OCT 13-16"16
BANGALORE
CONFENGINE

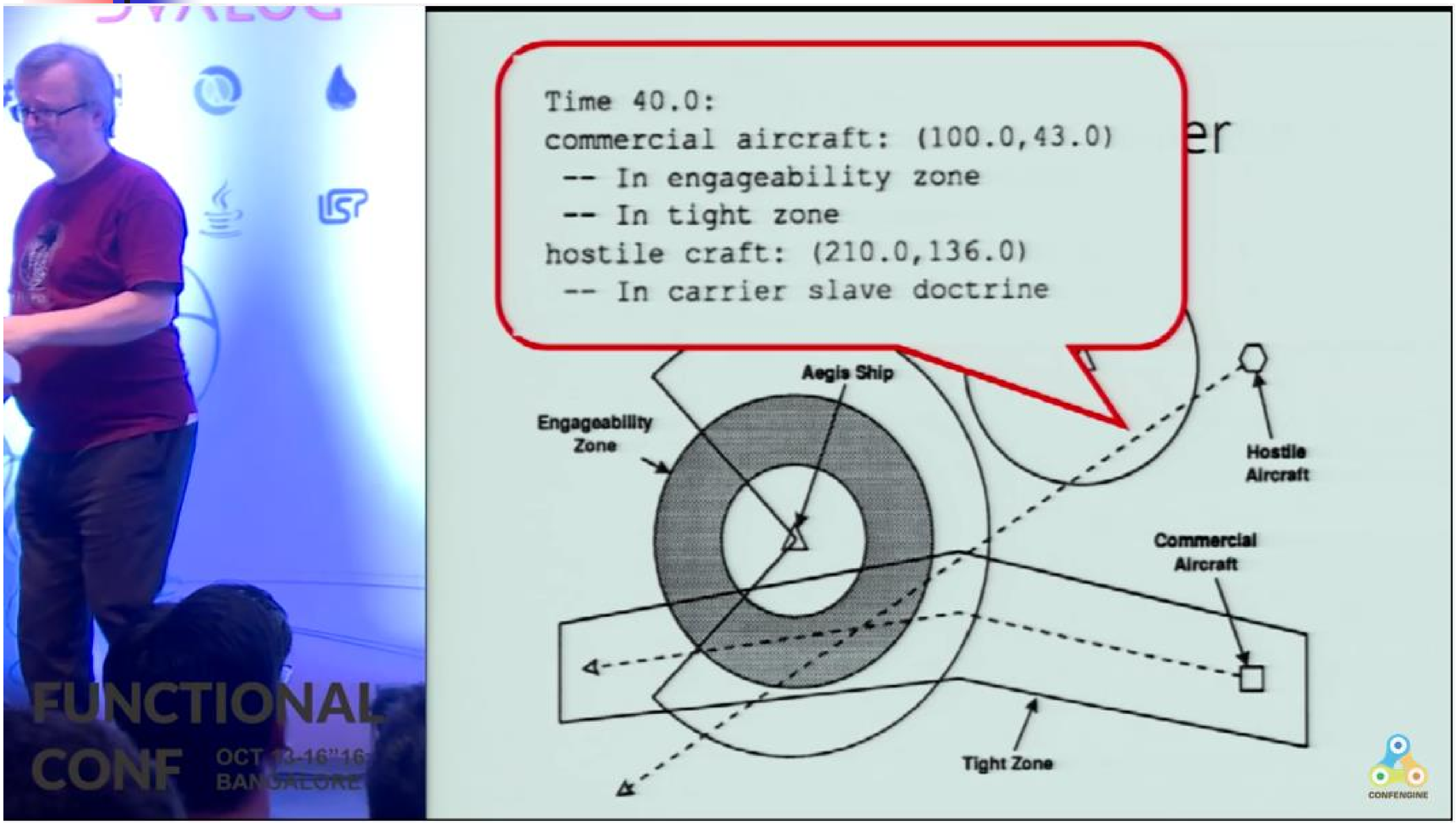


rot(p) (a,b,c) = p(a+b,c,-b)
c
a
b
-b
c
a+b


Laws
rot(above(p,q))
=
beside(rot(p),rot(q))
It seems there is a positive correlation between
the simplicity of the rules and the quality of the
algebra as a description tool.
FUNCTIONAL
CONF
BANGALORE
CONFENGINE



[***haskell***.*cs.yale.edu/wp-content/.../03/****HaskellVsAda****-NSWC.pdf*](haskell.cs.yale.edu/wp-content/.../03/HaskellVsAda-NSWC.pdf)


Time 40.0:
commercial aircraft: (100.0,43.0)
-- In engageability zone
-- In tight zone
hostile craft: (210.0,136.0)
-- In carrier slave doctrine
Aegis Ship
Engageability Zone
Hostile Aircraft
Commercial Aircraft
Tight Zone
FUNCTIONAL
CONF
CONFENGINE

Functions as Data

```
> type Region  =  Point -> Bool
```

Including 29 lines of inferable type signatures/synonyms

A student, given 8 days to learn Haskell, w/o knowledge of Yale group

Language	Lines of code	Lines of docum[illegible]	Development time (hours)
(1) Haskell	85	465	10
(2) Ada	767	[illegible]	23
(3) Ada9X	800	[illegible]	28
(4) C++	1105	[illegible]30	–
(5) Awk/Nawk	250	150	–
(6) Rapide	157	0	54
(7) Griffin	251	0	34
(8) Proteus	293	79	26
(9) Relational Lisp	274	12	3
(10) Haskell	156	112	8

Figure 3: Summary of Prototype Software Development Metrics

Reaction…

"too cute for its own good"

- …higher-order functions just a trick, probably not useful in other contexts

Haskell in FB spam filtering

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.

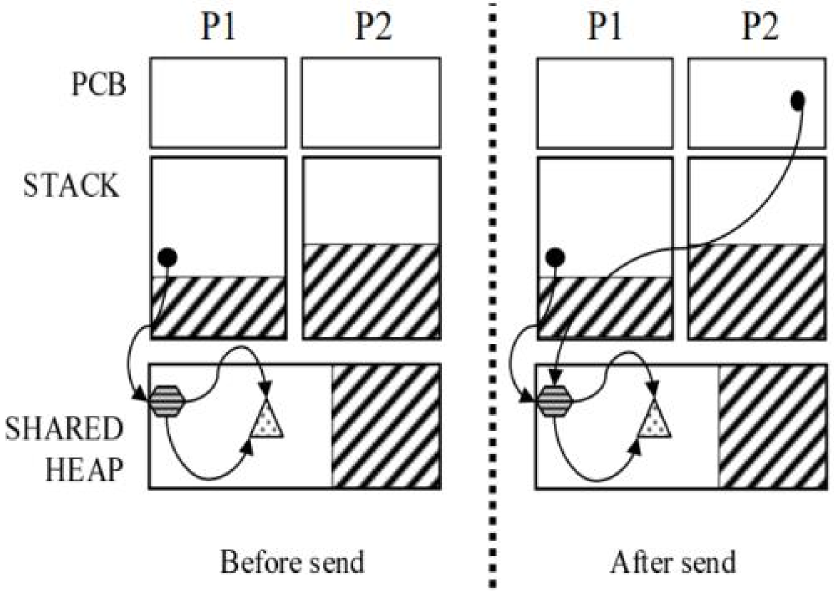
https://code.facebook.com/posts/745068642270222/fighting-spam-with-haskell/

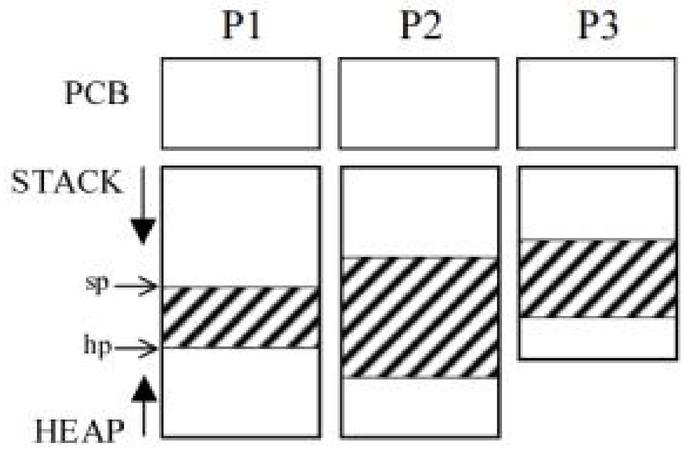
Elixir-Erlang

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

Referencie

- https://www.youtube.com/watch?v=XrNdvWqxBvA
- *www.cse.chalmers.se/~rjmh/Papers/whyfp.pdf*
- *https://www.cs.cmu.edu/~crary/819-f09/Landin66.pdf*
- *https://www.thocp.net/.../papers/**backus**_turingaward_lecture.pdf*
- *https://cs.au.dk/~hosc/local/HOSC-15-4-pp349-365.pdf*
- ***haskell**.cs.yale.edu/wp-content/.../03/**HaskellVsAda**-NSWC.pdf*
- https://code.facebook.com/posts/745068642270222/fighting-spam-with-haskell/
- https://www.fastcompany.com/3026758/inside-erlang-the-rare-programming-language-behind-whatsapps-success