ACS
ICT Educators’ Board

Paul Bailes
VP (Academic)
Australian Computer Society
Mission

• to develop professionalism in ICT teachers in school and more generally in the long term

• to build an integrated community of ICT educators from schools through VET to universities)
What we are and what we aren’t

• Are
  • ICT Teachers in Schools
  • Just like ICT Teachers/Lecturers/Professors in VET/Higher Ed/Universities

• Aren’t
  • Teachers in Schools using ICT
  • “ICT in Education”

• ACCE = ICT Teachers + ICT in Education

• ACS ICTEB = ICT Teachers (Schools + VET + Higher Ed + Universities)
Membership  (chair TBD)

• Katrina Falkner (Adelaide)
• Therese Keane (Swinburne)
• Ralph Leonard (Sabrenet)
• Ken Price (Tas Dept Ed)
• Nick Reynolds (Melbourne, IFIP TC3)

• Paul Bailes (ACS VP)
• Athol Chalmers (ACS NS)
  [Joel Cowey? (CSIRO)]
• Michael Johnson (ACS PSB)
• Karsten Schulz (NICTA)
• Iwona Miliszewska (ACDICT)
• Jason Zagami (ACCE)
• Marilyn Livesly (Secretary)
Current Commission

• to develop professionalism in ICT teachers in schools

e.g.

• through development of professional standards including body of knowledge
Current/imminent activities

• ACARA Digital Technologies Curriculum Development

• Support ACS contributions to Digital Careers
  • “Roadshows”
    • Guidelines for endorsement

• Teacher PD

• Support ICT infrastructure providers in schools

• BoK for ICT Teachers in Schools
“Coding” in Schools

• C

... 

• Computational thinking
  • CMU [http://www.cs.cmu.edu/~CompThink/](http://www.cs.cmu.edu/~CompThink/)
    Vs. (getting better)

... 

• Haskell
generic “categorical” abstraction
http://www.willamette.edu/~fruehr/haskell/evolution.html

-- explicit type recursion with functors and catamorphisms

-- fixed point operator
newtype Mu f = In (f (Mu f))
unIn (In x) = x

-- the generic catamorphism: fmap represents the shape of the type; phi represents the catamorphic operation
cata phi = phi . fmap (cata phi) . unIn

-- base functor and data type for natural numbers, using locally-defined “eliminators”; etc for other types
data N c = Z | S c
instance Functor N where -- supplies the characteristic “fmap” for this type
    fmap g Z = Z
    fmap g (S x) = S (g x)
E.g.

type Nat = Mu N  -- implements recursion for the above
  -- constructors
zero = In Z
suck n = In (S n)  -- “suck” because of name clash with predefined “succ”

-example specific operations
add m = cata phi where  -- supplies the characteristic “phi” for this operation
  phi Z = m  -- base value
  phi (S f) = suck f  -- recursive operation

  -- etc. etc. etc.