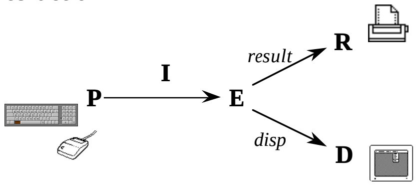
the PIE model

- ‘minimal’ model of interactive system
- focused on external observable aspects of interaction



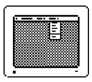
PIE model – user input

- sequence of commands
- commands include:
  - keyboard, mouse movement, mouse click

- call the set of commands C
- call the sequence P
  P = seq C

PIE model – system response

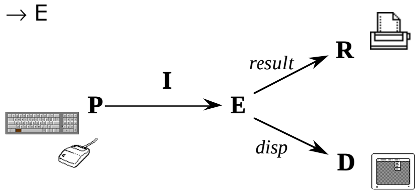
- the ‘effect’
- effect composed of:
  ephemeral display
  the final result
  - (e..g printout, changed file)



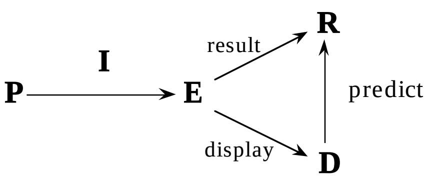
- call the set of effects E

PIE model – the connection

- given any history of commands (P)
- there is some current effect
- call the mapping the interpretation (I)
  I: P → E



properties – WYSIWYG



$\exists$ predict $\in$ (D → R) s.t. predict o display = result

- but really not quite the full meaning

state

- avoided to focus external aspects …

  … but often easier to think about!

- but can add it back
  - either ‘create’ it from I and E
  - or assume E is actually a state

creating the state!

- given a PIE
- say two command sequences as equivalent if they are indistinguishable in the future:
  $p \sim q \quad \text{iff} \quad \forall\, r \in H\!: I(pr) = I(qr)$
- the quotient set of P is a minimal state allowing the same effects

N.B. change notation use H (history) instead of P for sequence of commands

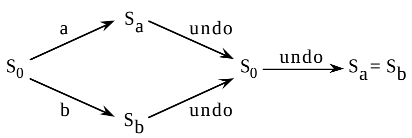
property of state

- for E to be a state it needs a state update function – call it 'doit':
  $\text{doit} : E \times C \to E$
- this needs to 'agree' with I:
  $\forall\, p \in H,\ c \in C\!:\ \text{doit}(I(p),c) = I(pc)$
- if E acts as a state we'll call it S with the initial state $s_0 = I(<>)$

proving things – undo

$\forall\, c : \quad c \text{ undo } \sim \text{ null} \quad ?$

only for $c \neq \text{undo}$



lesson

- undo is no ordinary command!
- other meta-commands:
  back/forward in browsers
  history window

undo as meta-command

- need to think of ordinary commands C plus augmented commands A:
  $C^a = C \cup A \qquad H^a = \text{seq } C^a$
- also augmented system state:
  S^a
- and behaviour:
  $\text{doit}^a\!:\ S^a \times C^a \to S^a$
  $I^a\!:\ H^a \to S^a$

two state (flip) undo

- system keeps two copies of state:
  $S^a = S \times S$
- ordinary commands update state:
  $\text{doit}^a\ (\ (s_{save}, s),\ c\ ) = (\ s,\ \text{doit}(s,c)\ )$
- undo (redo) flips states:
  $\text{doit}^a\ (\ (s_{save}, s),\ \text{undo}\ ) = (\ s,\ s_{save}\ )$

the real system inside

- the augmented system still needs to be the old system inside!
- link new and old with projection:
  proj: $S^a \to S$
- proj(s) is the old state 'inside'

projected state of flip undo ..

$S^a = S \times S$

- projected state simply second component:

  $proj^a\ (\ (s_{save}, s)\ ) = s$

properties of flip undo

- undo really reverses undo:
  undo undo ~ null (strong-uu)
- undo reverses ordinary commands on projected (original) state
  c undo $\sim_{proj}$ null (weak-cu)

$\forall\ s^a \in S^a : proj(doit^a\ (s^a, c\ undo)) = proj(s^a)$

stack (multistep) undo/redo

- augmented state is a stack of states:

  $S^m = S^+ \times Nat$
  $s^m{}_0 = << s_0 >, 1 >$

  $proj^m : S^m \to S$
  where $proj^m(< hs, n >) = hs[n]$

stack (multistep) undo/redo

- obvious (!) update:

  $doit^m(<hs,n>,undo) = <hs,n-1>$
  if n > 1 - otherwise nothing

  $doit^m(<hs,n>,redo) = <hs,n+1>$
  if n < length(hs)

  $doit^m(<hs,n>,c) =$
  $<<hs[1\ldots n],doit(hs[n],c)>,n+1>$

properties of multistep undo/redo

- redo really reverses undo:
  undo redo ~ null (strong-cu)
- undo reverses commands on projected (original) state
  c undo $\sim_{proj}$ null (weak-cu)
- the only way to satisfy these …

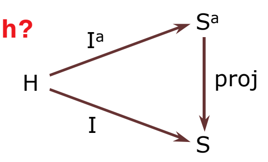
  Prove It !

the real system inside (2)

- behaviour?
- if no A commands ever used, identical:

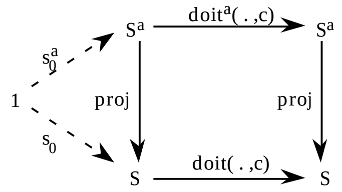
  $\forall\, h \in H : proj(I^a(h)) = I(h)$

enough?

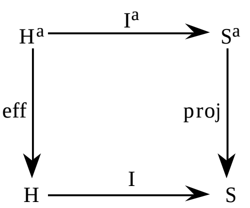


conservativeness of state

$proj(s_0^a) = s_0$
$\forall c \in C, s \in S^a : proj(doit^a(s, c)) = doit(proj(s), c)$

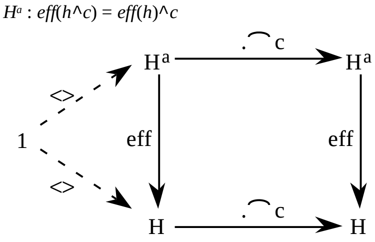


encapsulation



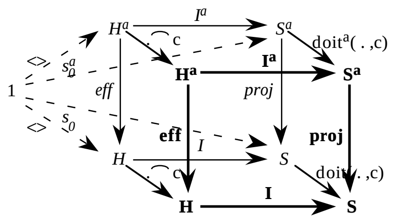
conservativeness of effective history

$eff(\langle\rangle) = \langle\rangle$
$\forall\, c \in C, h \in H^a : eff(h\hat{}c) = eff(h)\hat{}c$



the cube

$eff(\langle\rangle) = \langle\rangle$
$\forall\, c \in C, h \in H^a : eff(h\hat{}c) = eff(h)\hat{}c$



full details ...

R. Mancini (1997).
Modelling Interactive Computing by exploiting the Undo.
Dottorato di Ricerca in Informatica, IX-97-5, Università degli Studi di Roma "La Sapienza"