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
  \[ P = \text{seq } C \]

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)
  \[ I: P \to E \]

properties - WYSIWYG
- 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

\[ \exists \text{ predict } \in (D \to R) \text{ s.t. predict } \circ \text{ display } = \text{ result} \]
- but really not quite the full meaning
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’:
  \[ \text{doit} : E \times C \rightarrow E \]
- 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 \( s_0 = I(<> \rangle) \)

proving things – undo

\[ \forall c : \quad c \text{ undo } \sim \text{ null ?} \]
only for \( c \neq \text{ undo} \)

\[
\begin{array}{ccc}
S_0 & \xrightarrow{a} & S_a \\
& \text{undo} & \\
\downarrow & & \downarrow \\
S_0 & \xrightarrow{b} & S_b \\
& \text{undo} & \\
\end{array}
\]

\[ S_a = S_b \]

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:
  \[ C^a = C \cup A \quad H^p = \text{seq} C^a \]
- also augmented system state:
  \( S^a \)
- and behaviour:
  \[ \text{doit}^a : S^a \times C^a \rightarrow S^a \]
  \[ I^a : H^p \rightarrow S^a \]

two state (flip) undo

- system keeps two copies of state:
  \( S^a = S \times S \)
- ordinary commands update state:
  \[ \text{doit}^a ( (s_{\text{save}}, s), c ) = ( s, \text{doit}(s,c) ) \]
- undo (redo) flips states:
  \[ \text{doit}^a ( (s_{\text{save}}, s), \text{undo} ) = ( s, s_{\text{save}} ) \]
the real system inside
• the augmented system still needs to be the old system inside!
• link new and old with projection:
  \( \text{proj} : S^a \rightarrow S \)
• \( \text{proj}(s) \) is the old state ‘inside’

projected state of flip undo ..
\( S^a = S \times S \)
• projected state simply second component:
  \( \text{proj}^a( (s_{\text{save}},s) ) = s \)

properties of flip undo
• undo really reverses undo:
  \( \text{undo} \text{ undo} \sim \text{null} \) (strong-uu)
• undo reverses ordinary commands on projected (original) state
  \( c \text{ undo} \sim_{\text{proj}} \text{null} \) (weak-cu)
\( \forall s^a \in S^a : \text{proj}(\text{doit}(s^a, c \text{ undo})) = \text{proj}(s^a) \)

stack (multistep) undo/redo
• augmented state is a stack of states:
  \( S^m = S^+ \times \text{Nat} \)
  \( s^m_0 = << s_0 >, 1 > \)
  \( \text{proj}^m : S^m \rightarrow S \)
  where \( \text{proj}^m(<h, n>) = h[n] \)

stack (multistep) undo/redo
• obvious (!) update:
  \( \text{doitm}(<h, n>, \text{undo}) = <h, n-1> \)
  if \( n > 1 \) - otherwise nothing
  \( \text{doitm}(<h, n>, \text{redo}) = <h, n+1> \)
  if \( n < \text{length}(h) \)
  \( \text{doitm}(<h, n>, c) = <<h[1..n], \text{doit}(h[n], c)>, n+1> \)

properties of multistep undo/redo
• redo really reverses undo:
  \( \text{undo} \text{ redo} \sim \text{null} \) (strong-cu)
• undo reverses commands on projected (original) state
  \( c \text{ undo} \sim_{\text{proj}} \text{null} \) (weak-cu)
• the only way to satisfy these ...
  Prove It !
the real system inside (2)

- behaviour?
- if no A commands ever used, identical:
  \[ \forall h \in H : \text{proj}(I_a(h)) = I(h) \]

enough?

conservativeness of state

\[ \text{proj}(s) = s_a \]
\[ \forall c \in C, s \in S : \text{doit}(s, c) = \text{doit}(\text{proj}(s), c) \]

encapsulation

conservativeness of effective history

\[ \text{eff}(\langle \rangle) = \langle \rangle \]
\[ \forall c \in C, h \in H : \text{eff}(h^c) = \text{eff}(h)^c \]

the cube

full details ...

Dottorato di Ricerca in Informatica, IX-97-5, Università degli Studi di Roma "La Sapienza"