modelling state

looking within

what is state

time

modelling state

- describe state using variables
- types of variables:
  - basic type: \( x: \text{Nat} \) - non-negative integer (0,1,2,...)
  - individual item from set: \( \text{shape}: \{\text{circle, line, rectangle}\} \)
  - subset of bigger set: \( \text{selection}: \text{set Nat} \) - set of integers
  - function (often finite): \( \text{objects: Nat \rightarrow shape} \)
  - user defined: \( \text{Point} = x, y: \text{Real} \) - e.g. (1.79,-3.2)
stages

iteratively define:
state - what needs to be remembered
invariants - what is always true
initial state - how it starts
actions - what can happen to the state
(display - what the user sees (hears etc.)

use scenarios to check they are what you want

four function calculator

• formal description of the state
• define the effect of the following actions:
  type_digit(d) - user presses single digit
  equals - user presses '+' button
  op(p) - user presses '*' button

N.B. will not be right first time ... spot the mistakes

calculator state - first attempt

<table>
<thead>
<tr>
<th>state</th>
<th>initial state</th>
<th>display</th>
</tr>
</thead>
<tbody>
<tr>
<td>total: Nat</td>
<td>total = 0</td>
<td>disp</td>
</tr>
<tr>
<td>disp: Nat</td>
<td>disp = 0</td>
<td>- more complex calculator may show formulae</td>
</tr>
</tbody>
</table>
**calculator actions - first attempt**

- **type_digit(d)**
  - add d to the end of disp
  - total unchanged

- **equals**
  - do last operation “+,-,*,/” to disp and total

**what is it!**

**calculator state - second attempt**

- **state**
  - total: Nat – running total (accumulator)
  - disp: Nat – number currently displayed
  - pend_op: \{+,-,*,/\} – pending operation

- **initial state**
  - total = 0
  - disp = 0
  - pend_op = none

**calculator actions - second attempt**

- **type_digit(d)**
  - add d to the end of disp
  - total and pend_op unchanged

- **equals**
  - do pend_op to disp and total
  - put result in both disp and total
  - set pend_op to none

- **op(o)**
  - do pend_op to disp and total
  - put result in both disp and total
  - put o into pend_op
calculator - scenario

- user types: 1 + 2 7 = - 3
- start after 1 + 2

<table>
<thead>
<tr>
<th>action</th>
<th>total</th>
<th>disp</th>
<th>pend_op</th>
</tr>
</thead>
<tbody>
<tr>
<td>type_digit(7)</td>
<td>1 2</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>equals</td>
<td>28</td>
<td>28</td>
<td>none</td>
</tr>
<tr>
<td>op(−)</td>
<td>28</td>
<td>28</td>
<td>−</td>
</tr>
<tr>
<td>type_digit(3)</td>
<td>28</td>
<td>283</td>
<td></td>
</tr>
</tbody>
</table>

calculator state - third attempt

- added ‘typing’ flag
  - user in the middle of typing a number

<table>
<thead>
<tr>
<th>state</th>
</tr>
</thead>
<tbody>
<tr>
<td>total: Nat</td>
</tr>
<tr>
<td>disp: Nat</td>
</tr>
<tr>
<td>pend_op: {+,−,*,/,none}</td>
</tr>
<tr>
<td>typing: Bool</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>action - third attempt</th>
</tr>
</thead>
<tbody>
<tr>
<td>type_digit(d)</td>
</tr>
<tr>
<td>if typing then add d to the</td>
</tr>
<tr>
<td>end of disp</td>
</tr>
<tr>
<td>otherwise clear disp and</td>
</tr>
<tr>
<td>put d in it</td>
</tr>
<tr>
<td>also set typing to true</td>
</tr>
<tr>
<td>total and pend_op unchanged</td>
</tr>
<tr>
<td>equals and op(o):</td>
</tr>
<tr>
<td>- as before except both set</td>
</tr>
<tr>
<td>typing to false</td>
</tr>
</tbody>
</table>
### calculator - scenario revisited

- **user types:** \(1 + 2 \quad 7 = -3\)
- **start after:** \(1 + 2\)

<table>
<thead>
<tr>
<th>action</th>
<th>total</th>
<th>disp</th>
<th>pend_op</th>
<th>typing</th>
</tr>
</thead>
<tbody>
<tr>
<td>type_digit(?)</td>
<td>1</td>
<td>2</td>
<td>+</td>
<td>yes</td>
</tr>
<tr>
<td>equals</td>
<td>28</td>
<td>28</td>
<td>none</td>
<td>no</td>
</tr>
<tr>
<td>op(−)</td>
<td>28</td>
<td>28</td>
<td>−</td>
<td>no</td>
</tr>
<tr>
<td>type_digit(3)</td>
<td>28</td>
<td>2</td>
<td>−</td>
<td>yes</td>
</tr>
</tbody>
</table>

### defining state

two problems:

- **too little state**
  - elements missing from specification
  - may be deliberate
    - e.g. dialogue level spec.
  - forgotten elements
    - e.g. ‘typing’ flag for calculator

- **too much state**
  - too many states, too complex state
  - may be deliberate
    - redundancy, extensibility

### too little state

- **forgotten elements**
  - e.g. ‘typing’ flag for calculator

- **checking:**
  - dialogue state
    - can you work out current dialogue state?
  - action specification
    - do you have enough information?
  - implicit global variables (see also later)
    - suggest state missing
too much state

- unreachable states
  - too few actions (see later)
  - constraints
  - states are not orthogonal
- spare variables: constant/functional dependent
- dependent state
  - e.g. first point of line, number being typed
- indistinguishable states
  - what is observable?

defining actions

- framing problems
  - too little in result state
- unreachable states – insufficient actions
- using 'global' variables
  - implicit in operation definition
- beware extreme cases
  - e.g. empty document, cursor at end of line

internal and external consistency

- invariants preserved?
  - actions complete?
- missing state?
  - makes sense?
- general properties
  - specific examples
- scenarios