

HUMAN-COMPUTER INTERACTION THIRD EDITION
 DIX FINLAY ABOWD BEALE

chapter 16

dialogue notations and design

Dialogue Notations and Design

- Dialogue Notations
 - Diagrammatic
 - state transition networks, JSD diagrams, flow charts
 - Textual
 - formal grammars, production rules, CSP
- Dialogue linked to
 - the semantics of the system – what it does
 - the presentation of the system – how it looks
- Formal descriptions can be analysed
 - for inconsistent actions
 - for difficult to reverse actions
 - for missing actions
 - for potential miskeying errors

what is dialogue?

- conversation between two or more parties
 - usually cooperative
- in user interfaces
 - refers to the *structure* of the interaction
 - syntactic level of human-computer ‘conversation’
- levels
 - lexical – shape of icons, actual keys pressed
 - syntactic – order of inputs and outputs
 - semantic – effect on internal application/data

structured human dialogue

- human-computer dialogue very constrained
- some human-human dialogue formal too ...

Minister: do you *man's name* take this woman ...
Man: I do
Minister: do you *woman's name* take this man ...
Woman: I do
Man: With this ring I thee wed
 (*places ring on womans finger*)
Woman: With this ring I thee wed (*places ring ..*)
Minister: I now pronounce you man and wife

lessons about dialogue

- wedding service
 - sort of script for three parties
 - specifies order
 - some contributions fixed – “I do”
 - others variable – “do you *man's name* ...”
 - instructions for ring concurrent with saying words “with this ring ...”
- if you say these words are you married?
 - only if in the right place, with marriage licence
 - syntax not semantics

... and more

- what if woman says “I don't”?
- real dialogues often have alternatives:

Judge: How do you plead guilty or not guilty?
Defendant: *either* Guilty or Not guilty

 - the process of the trial depends on the defendants response
- focus on normative responses
 - doesn't cope with judge saying “off with her head”
 - or in computer dialogue user standing on keyboard!

HUMAN-COMPUTER INTERACTION

dialogue design notations

- dialogue gets buried in the program
- in a big system can we:
 - analyse the dialogue:
 - can the user always get to see current shopping basket
 - change platforms (e.g. Windows/Mac)
 - dialogue notations helps us to
 - analyse systems
 - separate lexical from semantic
- ... and before the system is built
 - notations help us understand proposed designs

HUMAN-COMPUTER INTERACTION

graphical notations

state-transition nets (STN)
Petri nets, state charts
flow charts, JSD diagrams

HUMAN-COMPUTER INTERACTION

State transition networks (STN)

- circles - states
- arcs - actions/events

```

graph LR
    Start((Start)) --> Menu((Menu))
    Menu -- "select 'circle'" --> Circle1((Circle 1))
    Menu -- "select line" --> Line1((Line 1))
    Circle1 -- "click on centre" --> Circle2((Circle 2))
    Circle1 -- "rubber band" --> Circle2
    Circle2 -- "click on circumference" --> Finish1((Finish))
    Circle2 -- "draw circle" --> Finish1
    Line1 -- "click on first point" --> Line2((Line 2))
    Line1 -- "rubber band" --> Line2
    Line2 -- "double click" --> Finish2((Finish))
    Line2 -- "draw last line" --> Finish2
    Line2 -- "click on point" --> Line2
    Line2 -- "draw a line" --> Line2
  
```

State transition networks - events

• arc labels a bit cramped because:

- notation is 'state heavy'
- the events require most detail

State transition networks - states

• labels in circles a bit uninformative:

- states are hard to name
- but easier to visualise

Hierarchical STNs

• managing complex dialogues

• named sub-dialogues

Concurrent dialogues - I
simple dialogue box

Text Style

example

bold

italic

underline

Concurrent dialogues - II
three toggles - individual STNs

NO bold ← click on 'bold' → bold bold

NO italic ← click on 'italic' → italic italic

NO u'line ← click on 'underline' → u'line underline

Concurrent dialogues - III
bold and italic combined

NO style ← click on 'bold' → bold only

italic only ← click on 'bold' → bold italic

click on 'italic' ↑ ↓ click on 'italic'

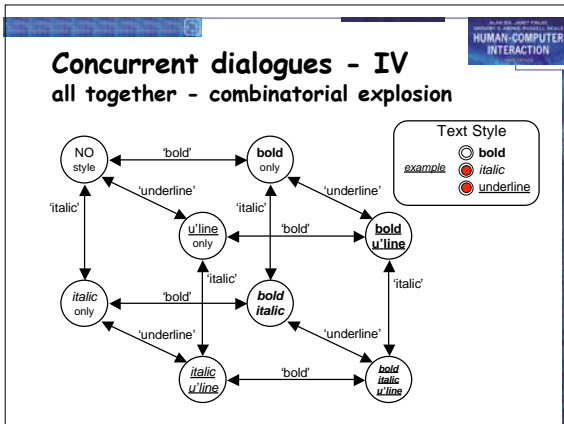
Text Style

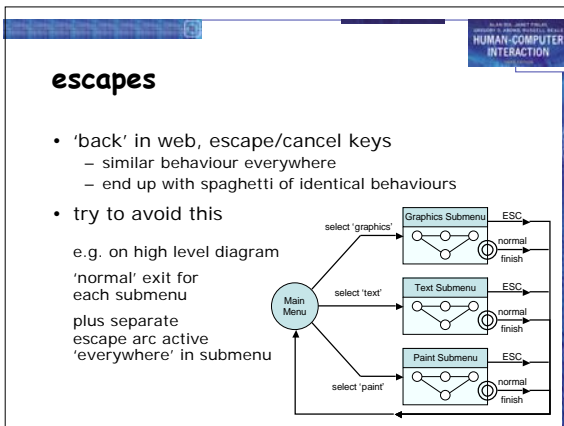
example

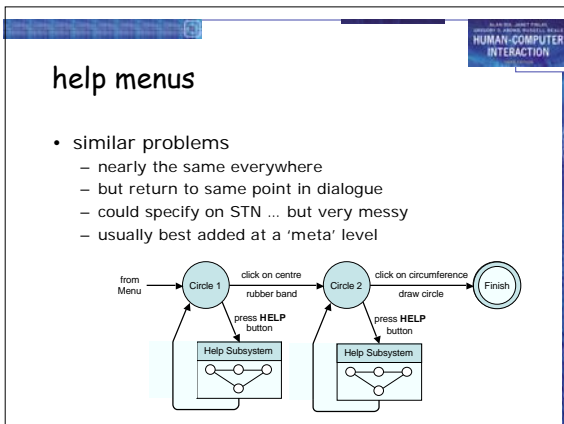
bold

italic

underline



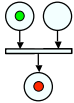




HUMAN-COMPUTER INTERACTION

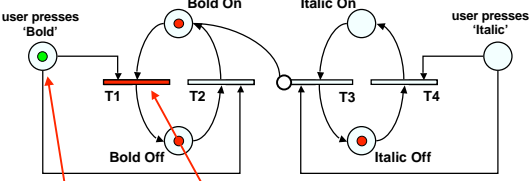
Petri nets

- one of the oldest notations in computing!
- flow graph:
 - places – a bit like STN states
 - transitions – a bit like STN arcs
 - counters – sit on places (current state)
- several counters allowed
 - concurrent dialogue states
- used for UI specification (ICO at Toulouse)
 - tool support – Petshop



HUMAN-COMPUTER INTERACTION

Petri net example



user presses 'Bold'

user presses 'Italic'

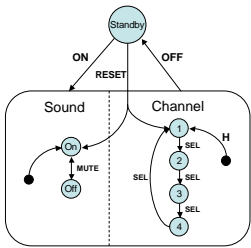
user actions represented as a new counter

transition 'fires' when all input places have counters

HUMAN-COMPUTER INTERACTION

State charts

- used in UML
- extension to STN
 - hierarchy
 - concurrent sub-nets
 - escapes
 - OFF always active
 - history
 - link marked H goes back to last state on re-entering subdialogue



Flowcharts

- familiar to programmers
- boxes
 - process/event
 - not state
- use for dialogue (not internal algorithm)

it works!

- formal notations – too much work?
- COBOL transaction processing
 - event-driven – like web interfaces
 - programs structure ≠ dialogue structure
- used dialogue flow charts
 - discuss with clients
 - transform to code
 - systematic testing
 - 1000% productivity gain
- formalism saves time!!

JSD diagrams

- for tree structured dialogues
 - less expressive
 - greater clarity

textual notations

- grammars
- production rules
- CSP and event algebras

Textual - Grammars

- Regular expressions
`sel-line click click* dble-click`
- compare with JSD
 - same computational model
 - different notation
- BNF

```
expr ::= empty
      | atom expr
      | '(' expr ')' expr
```
- more powerful than regular exp. or STNs
- Still NO concurrent dialogue

Production rules

- Unordered list of rules:

if *condition* then *action*

 - condition based on state or pending events
 - every rule always potentially active
- Good for concurrency
- Bad for sequence

HUMAN-COMPUTER INTERACTION

Event based production rules

```

Sel-line → first
C-point first → rest
C-point rest → rest
D-point rest → < draw line >

```

- Note:
 - events added to list of pending events
 - 'first' and 'rest' are internally generated events
- Bad at state!

HUMAN-COMPUTER INTERACTION

Prepositional Production System

- State based
- Attributes:
 - Mouse: { mouse-off, select-line, click-point, double-click }
 - Line-state: { menu, first, rest }
- Rules (feedback not shown):
 - select-line → mouse-off first
 - click-point first → mouse-off rest
 - click-point rest → mouse-off
 - double-click rest → mouse-off menu
- Bad at events!

HUMAN-COMPUTER INTERACTION

CSP and process algebras

- used in Alexander's SPI, and Agent notation
- good for sequential dialogues


```

Bold-tog = select-bold? → bold-on → select-bold? →
                                             bold-off → Bold-tog

```

```

Italic-tog = . . .
Under-tog = . . .

```
- and concurrent dialogue


```

Dialogue-box = Bold-tog || Italic-tog || Under-tog

```
- but causality unclear

HUMAN-COMPUTER INTERACTION

Dialogue Notations - Summary

- Diagrammatic
 - STN, JSD, Flow charts
- Textual
 - grammars, production rules, CSP
- Issues
 - event base vs. state based
 - power vs. clarity
 - model vs. notation
 - sequential vs. concurrent

HUMAN-COMPUTER INTERACTION

Semantics Alexander SPI (i)

- Two part specification:
 - EventCSP - pure dialogue order
 - EventISL - target dependent semantics
- dialogue description - centralised
- syntactic/semantic trade-off - tollerable

HUMAN-COMPUTER INTERACTION

Semantics Alexander SPI (ii)

- EventCSP


```

Login = login-mess -> get-name -> Passwd
Passwd = passwd-mess -> (invalid -> Login [] valid -> Session)
      
```
- EventISL


```

event: login-mess
  prompt: true
  out: "Login:"
event: get-name
  uses: input
  set: user-id = input
event: valid
  uses: input, user-id, passwd-db
  wgen: passwd-id = passwd-db(user-id)
      
```

HUMAN-COMPUTER INTERACTION

Semantics - raw code

- event loop for word processor
- dialogue description
 - very distributed
- syntactic/semantic trade-off
 - terrible!

```

switch ( ev.type ) {
  case button_down:
    if ( in_text ( ev.pos ) ) {
      mode = selecting;
      mark_selection_start(ev.pos);
    }
    ...
  case button_up:
    if ( in_text ( ev.pos )
        && mode == selecting ) {
      mode = normal;
      mark_selection_end(ev.pos);
    }
    ...
  case mouse_move:
    if ( mode == selecting ) {
      extend_selection(ev.pos);
    }
    ...
} /* end of switch */

```

HUMAN-COMPUTER INTERACTION

Action properties

- completeness
 - missed arcs
 - unforeseen circumstances
- determinism
 - several arcs for one action
 - deliberate: application decision
 - accident: production rules
- nested escapes
- consistency
 - same action, same effect?
 - modes and visibility

HUMAN-COMPUTER INTERACTION

Checking properties (i)

- completeness
 - double-click in circle states?

HUMAN-COMPUTER INTERACTION

Checking properties (ii)

- Reversibility:
 - to reverse select `line`

HUMAN-COMPUTER INTERACTION

Checking properties (ii)

- Reversibility:
 - to reverse select `line`
 - click

HUMAN-COMPUTER INTERACTION

Checking properties (ii)

- Reversibility:
 - to reverse select `line`
 - click - double click

Checking properties (ii)

- Reversibility:
 - to reverse select `line`
 - click - double click - select `graphics`
 - (3 actions)
- N.B. not undo

State properties

- reachability
 - can you get anywhere from anywhere?
 - and how easily
- reversibility
 - can you get to the previous state?
 - but NOT undo
- dangerous states
 - some states you don't want to get to

Dangerous States

- word processor: two modes and exit
 - F1 - changes mode
 - F2 - exit (and save)
 - Esc - no mode change

but ... Esc resets autosave

HUMAN-COMPUTER INTERACTION

Dangerous States (ii)

- exit with/without save \Rightarrow dangerous states
- duplicate states - semantic distinction

```

    graph TD
      edit1((edit)) -- F1 --> menu1((menu))
      menu1 -- F2 --> exit1(((exit)))
      edit2(((edit))) -- F1 --> menu2(((menu)))
      menu2 -- Esc --> exit2(((exit)))
      edit1 -- any update --> edit2
  
```

F1-F2 - exit with save
F1-Esc-F2 - exit with no save

HUMAN-COMPUTER INTERACTION

Lexical Issues

- visibility
 - differentiate modes and states
 - annotations to dialogue
- style
 - command - verb noun
 - mouse based - noun verb
- layout
 - not just appearance ...

HUMAN-COMPUTER INTERACTION

layout matters

- word processor - dangerous states
- old keyboard - OK

```

    graph TD
      edit1((edit)) -- F1 --> menu1((menu))
      menu1 -- F2 --> exit1(((exit)))
      edit2(((edit))) -- F1 --> menu2(((menu)))
      menu2 -- Esc --> exit2(((exit)))
      edit1 -- any update --> edit2
  
```

layout matters

• new keyboard layout

intend F1-F2 (save)
finger catches Esc

layout matters

• new keyboard layout

intend F1-F2 (save)
finger catches Esc
F1-Esc-F2 - disaster!

Dialogue Analysis - Summary

- Semantics and dialogue
 - attaching semantics
 - distributed/centralised dialogue description
 - maximising syntactic description
- Properties of dialogue
 - action properties: completeness, determinism, consistency
 - state properties: reachability, reversibility, dangerous states
- Presentation and lexical issues
 - visibility, style, layout
 - N.B. not independent of dialogue

HUMAN-COMPUTER INTERACTION

Dialogue Analysis - Summary

- Semantics and dialogue
 - attaching semantics
 - distributed/centralised dialogue description
 - maximising syntactic description
- Properties of dialogue
 - action properties: completeness, determinism, consistency
 - state properties: reachability, reversibility, dangerous states
- Presentation and lexical issues
 - visibility, style, layout
 - N.B. not independent of dialogue

HUMAN-COMPUTER INTERACTION

Digital watch - User Instructions

- two main modes
- limited interface - 3 buttons
- button A changes mode

HUMAN-COMPUTER INTERACTION

Digital watch - User Instructions

- dangerous states
 - *guarded*
 - ... by two second hold
- completeness
 - distinguish depress A and release A
 - what do they do in all modes?

