Navigation and Dialogue

HCI Lecture 7

David Aspinall

Informatics, University of Edinburgh

12th October 2007

Outline

Interface Design Roadmap

Conceptual Design	what is the conceptual model?
Physical Design	what physical environment?
Interaction Modes	what styles are appropriate?

Interface Design Roadmap

Conceptual Design	what is the conceptual model?
Physical Design	what physical environment?
Interaction Modes	what styles are appropriate?
Navigation Design	how is the interface structured?
Dialogue Design	how to link interactions?
Information Presentation	how to show feedback/results?
Screen Layout	best grouping/structure/alignment?

Interface Design Roadmap

Conceptual Design	what is the conceptual model?
Physical Design	what physical environment?
Interaction Modes	what styles are appropriate?
Navigation Design	how is the interface structured?
Dialogue Design	how to link interactions?
Information Presentation	how to show feedback/results?
Screen Layout	best grouping/structure/alignment?

- High-level to low-level, task-oriented refinement
- Data and presentation-oriented sometimes better:
  - task focus may suggest long tedious dialogues
  - instead: compact and *interactive* data presentation

Interface Design Roadmap

Conceptual Design	what is the conceptual model?
Physical Design	what physical environment?
Interaction Modes	what styles are appropriate?
Navigation Design	how is the interface structured?
Dialogue Design	how to link interactions?
Information Presentation	how to show feedback/results?
Screen Layout	best grouping/structure/alignment?

- High-level to low-level, task-oriented refinement
- Data and presentation-oriented sometimes better:
  - task focus may suggest long tedious dialogues
  - instead: compact and *interactive* data presentation
- This lecture: *notations* to describe navigation and dialogue design

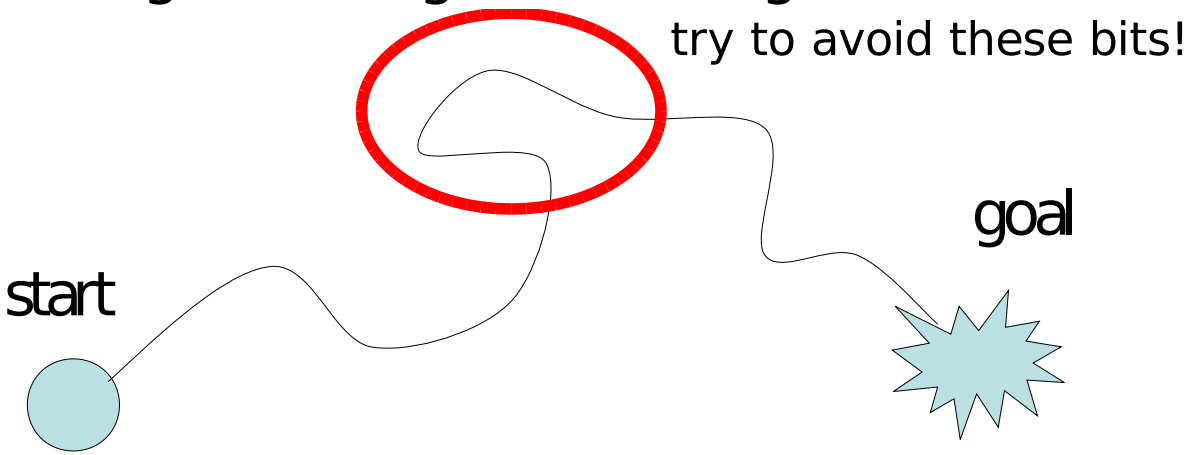
Outline

Navigation Design

- Golden rules — the *Where*^{3}*What* of navigation:
  - **W**here you are
  - **W**here you're going (or what will happen)
  - **W**here you've been (or what has been done)
  - **W**hat you can do now

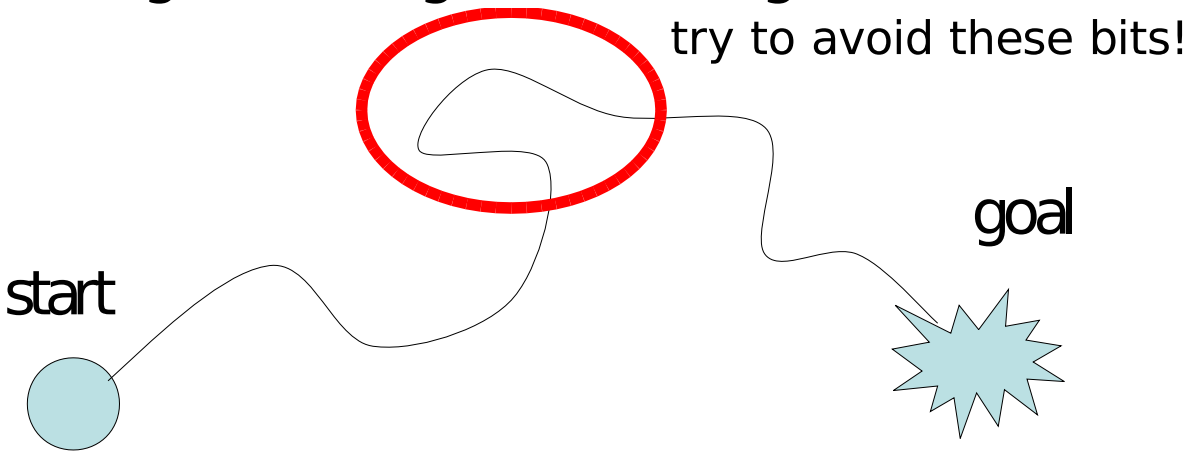
Navigation Design

- Golden rules — the *Where*^{3}*What* of navigation:
  - **W**here you are
  - **W**here you're going (or what will happen)
  - **W**here you've been (or what has been done)
  - **W**hat you can do now
- Often, navigation is goal seeking:



Navigation Design

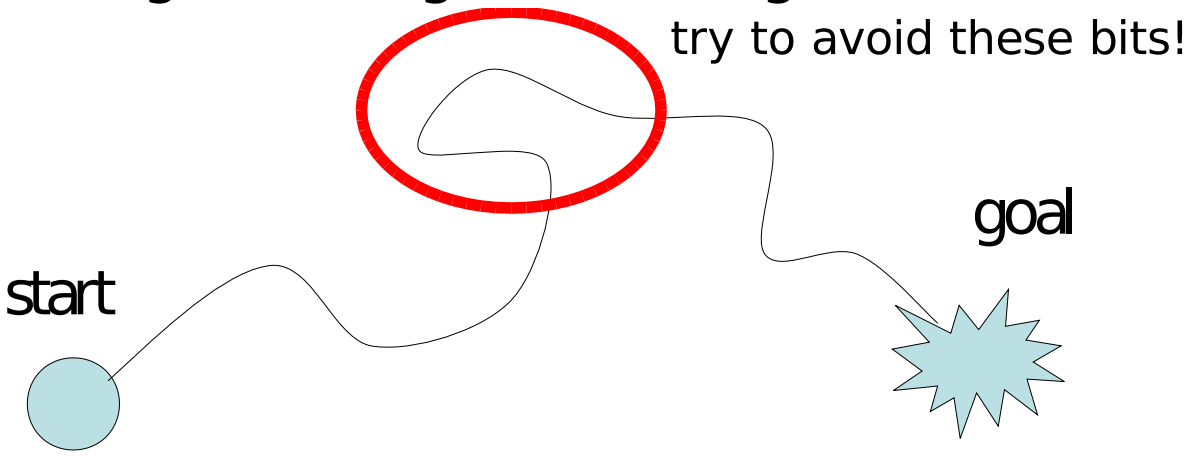
- Golden rules — the *Where*^{3}*What* of navigation:
  - **W**here you are
  - **W**here you're going (or what will happen)
  - **W**here you've been (or what has been done)
  - **W**hat you can do now
- Often, navigation is goal seeking:



- Different levels of structure, according to domain:
  - *app*: widgets; screens; application; environment

Navigation Design

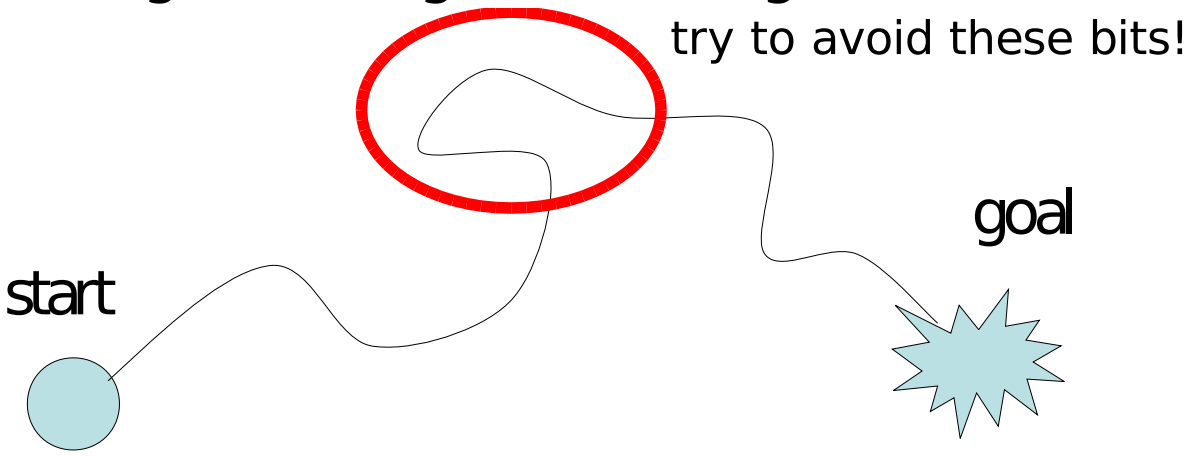
- Golden rules — the *Where*^{3}*What* of navigation:
  - **W**here you are
  - **W**here you're going (or what will happen)
  - **W**here you've been (or what has been done)
  - **W**hat you can do now
- Often, navigation is goal seeking:



- Different levels of structure, according to domain:
  - *app*: widgets; screens; application; environment
  - *web*: HTML; page layout; site; browser+www

Navigation Design

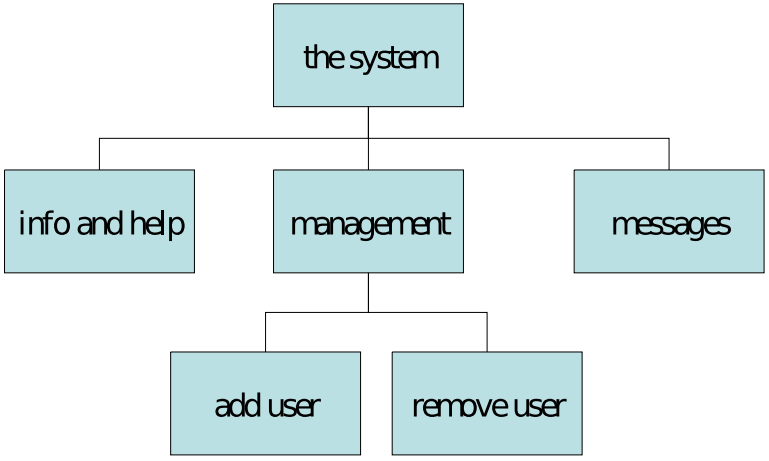
- Golden rules — the *Where*^{3}*What* of navigation:
  - **W**here you are
  - **W**here you're going (or what will happen)
  - **W**here you've been (or what has been done)
  - **W**hat you can do now
- Often, navigation is goal seeking:



- Different levels of structure, according to domain:
  - *app*: widgets; screens; application; environment
  - *web*: HTML; page layout; site; browser+www
  - *device*: controls; physical layout; modes; real world

Static Structure Diagrams

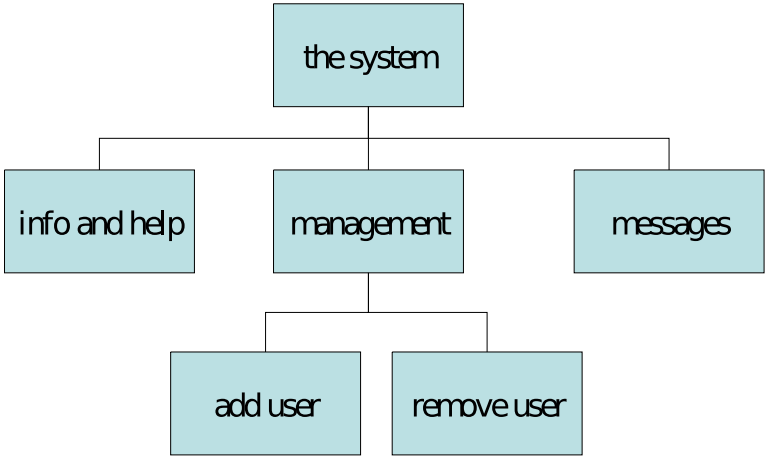
Screen hierarchy



- shows structure/relationship
- system-oriented
- remember: deep is difficult!

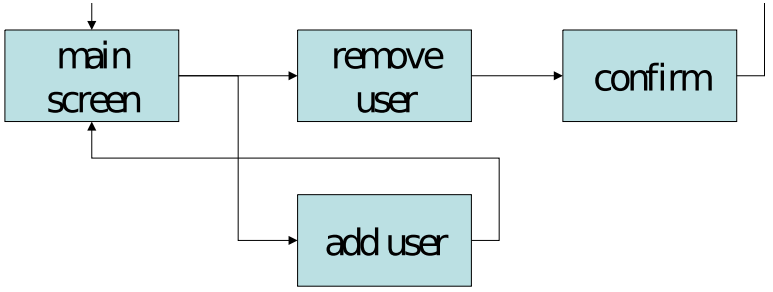
Static Structure Diagrams

Screen hierarchy



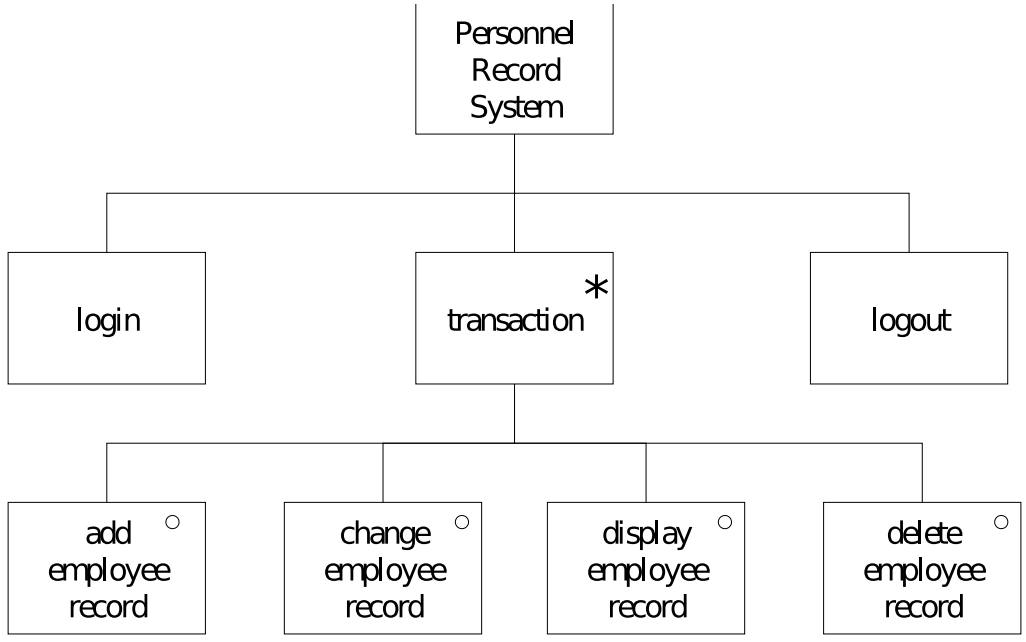
- shows structure/relationship
- system-oriented
- remember: deep is difficult!

Navigation network



- show different paths through system
- including branches
- more task-oriented

JSD Diagram



- Old-fashioned technology and limited
- . . . but easily understood
- close connection to HTA

Outline

Dialogue in UIs

- **Dialogue** is the pattern of interaction between users and system
  - may be schematic (fill in blanks, e.g. names in wedding vows)
  - but course may change according to responses

Dialogue in UIs

- **Dialogue** is the pattern of interaction between users and system
  - may be schematic (fill in blanks, e.g. names in wedding vows)
  - but course may change according to responses
- In UIs, Dialogue Design:
  - refers to *structure* of interaction
  - often low-level (cf cognitive models)
  - not only conversing, also for instructing, manipulating, etc

Dialogue in UIs

- **Dialogue** is the pattern of interaction between users and system
  - may be schematic (fill in blanks, e.g. names in wedding vows)
  - but course may change according to responses
- In UIs, Dialogue Design:
  - refers to *structure* of interaction
  - often low-level (cf cognitive models)
  - not only conversing, also for instructing, manipulating, etc
- Recall levels:
  - *lexical*: key or button presses/releases, icon shapes
  - ***syntactic***: order of inputs/outputs
  - *semantic*: actual effect on application/data

Dialogue Notations

- Dialogue can get buried in the program or designed carelessly “on-demand”

Dialogue Notations

- Dialogue can get buried in the program or designed carelessly “on-demand”
- Instead we may describe it alone, precisely using:
  - diagrammatic notations
  - textual notations
  - specific programming tools

Dialogue Notations

- Dialogue can get buried in the program or designed carelessly “on-demand”
- Instead we may describe it alone, precisely using:
  - diagrammatic notations
  - textual notations
  - specific programming tools
- Formal notations useful for testing, esp if executable

Dialogue Notations

- Dialogue can get buried in the program or designed carelessly “on-demand”
- Instead we may describe it alone, precisely using:
  - diagrammatic notations
  - textual notations
  - specific programming tools
- Formal notations useful for testing, esp if executable
- Also allow **analysis**, e.g., to find:
  - difficult to reverse actions
  - missing actions
  - inconsistent actions
  - unreachable or unrecoverable states
  - likely errors

Dialogue Notations

- Dialogue can get buried in the program or designed carelessly “on-demand”
- Instead we may describe it alone, precisely using:
  - diagrammatic notations
  - textual notations
  - specific programming tools
- Formal notations useful for testing, esp if executable
- Also allow **analysis**, e.g., to find:
  - difficult to reverse actions
  - missing actions
  - inconsistent actions
  - unreachable or unrecoverable states
  - likely errors
- To give semantics, descriptions can be linked (maybe mechanically) to *behaviour* or *presentation*.

Dialogue Notation Formalisms

- **State Transition Networks**
  - graphical notation
  - easy to understand
  - limited expressivity

Dialogue Notation Formalisms

- **State Transition Networks**
  - graphical notation
  - easy to understand
  - limited expressivity
- **Grammars**
  - textual
  - can be harder to understand
  - good expressivity

Dialogue Notation Formalisms

- **State Transition Networks**
  - graphical notation
  - easy to understand
  - limited expressivity
- **Grammars**
  - textual
  - can be harder to understand
  - good expressivity
- **Process Calculi**
  - textual, primarily
  - harder to understand
  - good expressivity, esp for concurrency

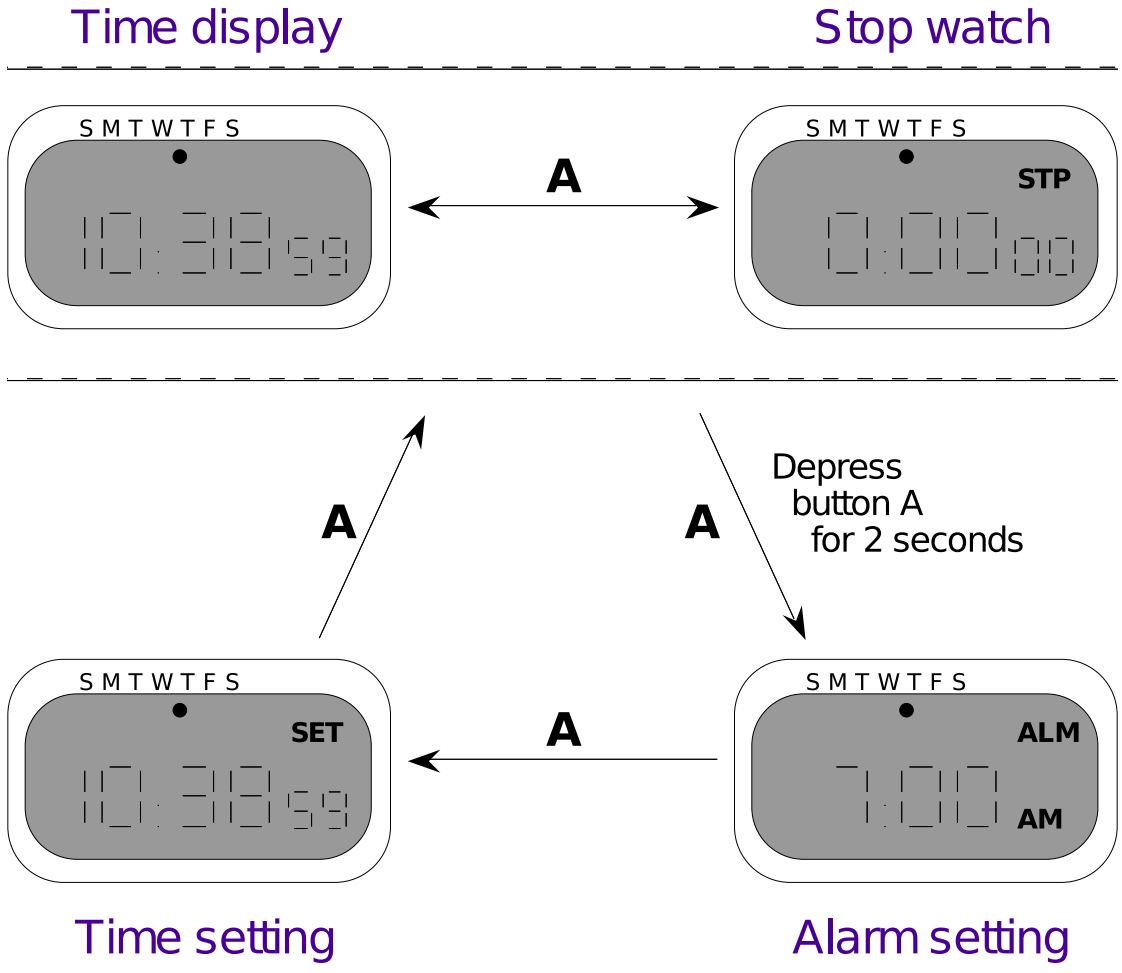
Dialogue Notation Formalisms

- **State Transition Networks**
  - graphical notation
  - easy to understand
  - limited expressivity
- **Grammars**
  - textual
  - can be harder to understand
  - good expressivity
- **Process Calculi**
  - textual, primarily
  - harder to understand
  - good expressivity, esp for concurrency
- Many others
  - flowcharts, **JSD diagrams**
  - production rules (actions guarded by events)
  - Petri Nets
  - State charts, *State and activity diagrams* (UML)

State Transition Network (STN)

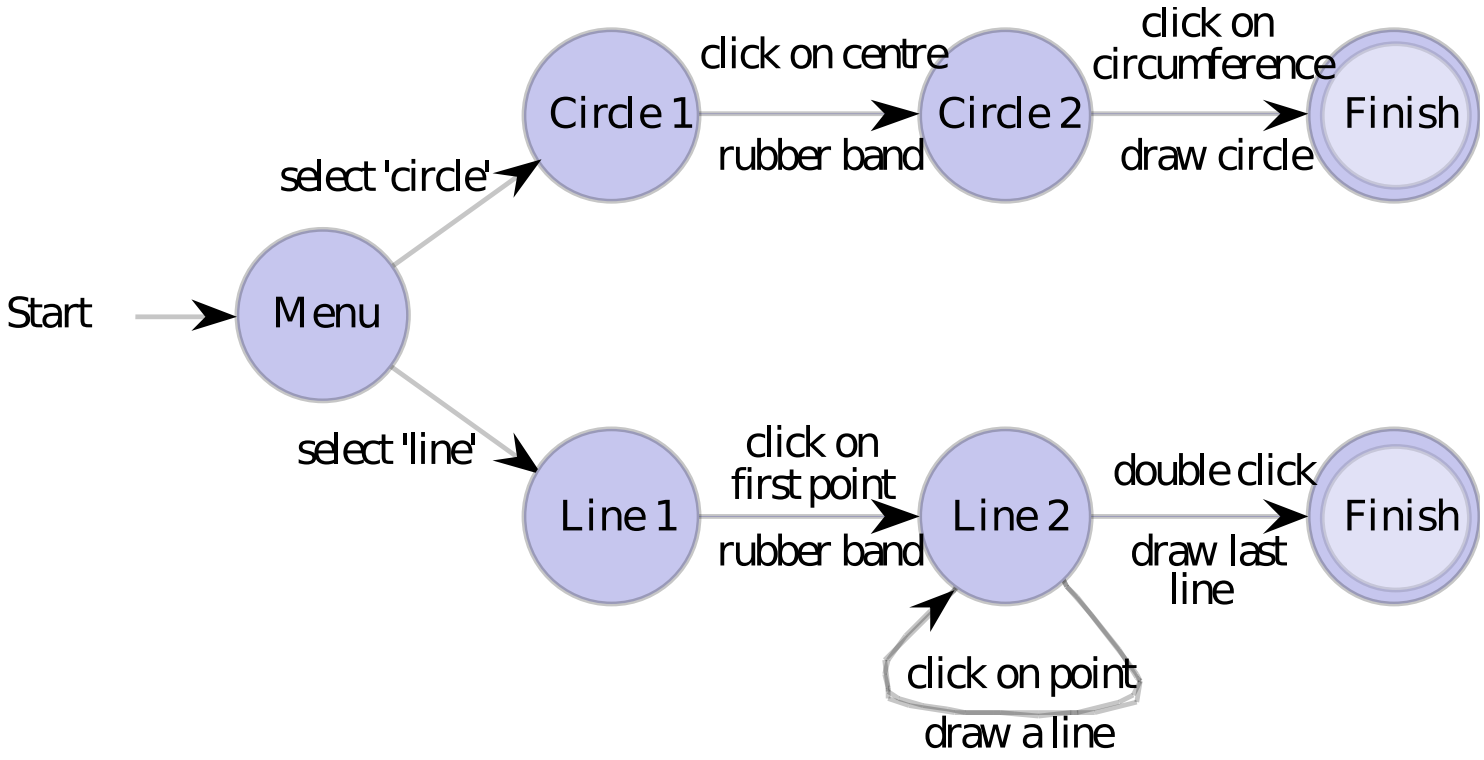
- Like a finite state machine with I/O (a transducer)
  - edges are input events and resulting actions
- Good for capturing sequential behaviour of dialogues
- Poor at capturing concurrency, escape, errors
  - State or edge “blow up”
- Diagrams can become cluttered and obscure
  - clutter: too many states, use **hierarchical STNs**
  - obscure: state names somewhat arbitrary

STN for a Watch with Modes

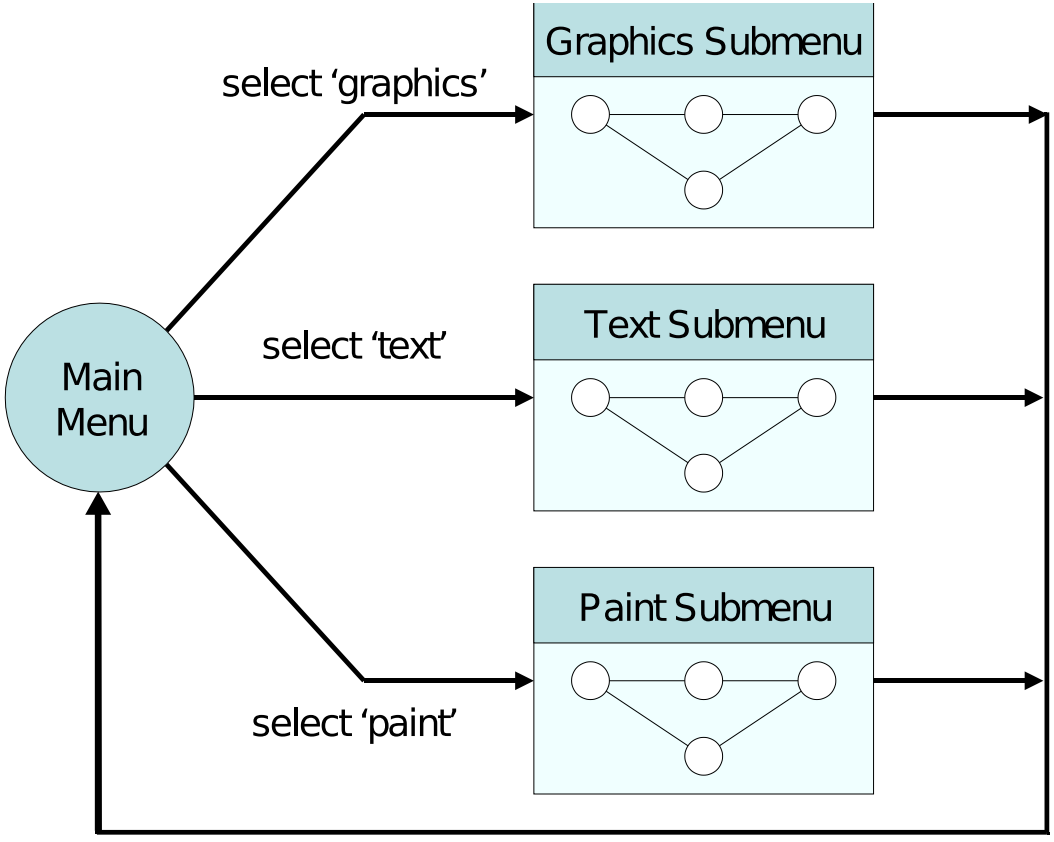


Modes (where control mappings change) are introduced by event timings. Modes have obvious drawbacks but economise on controls.

STN for a Drawing Program

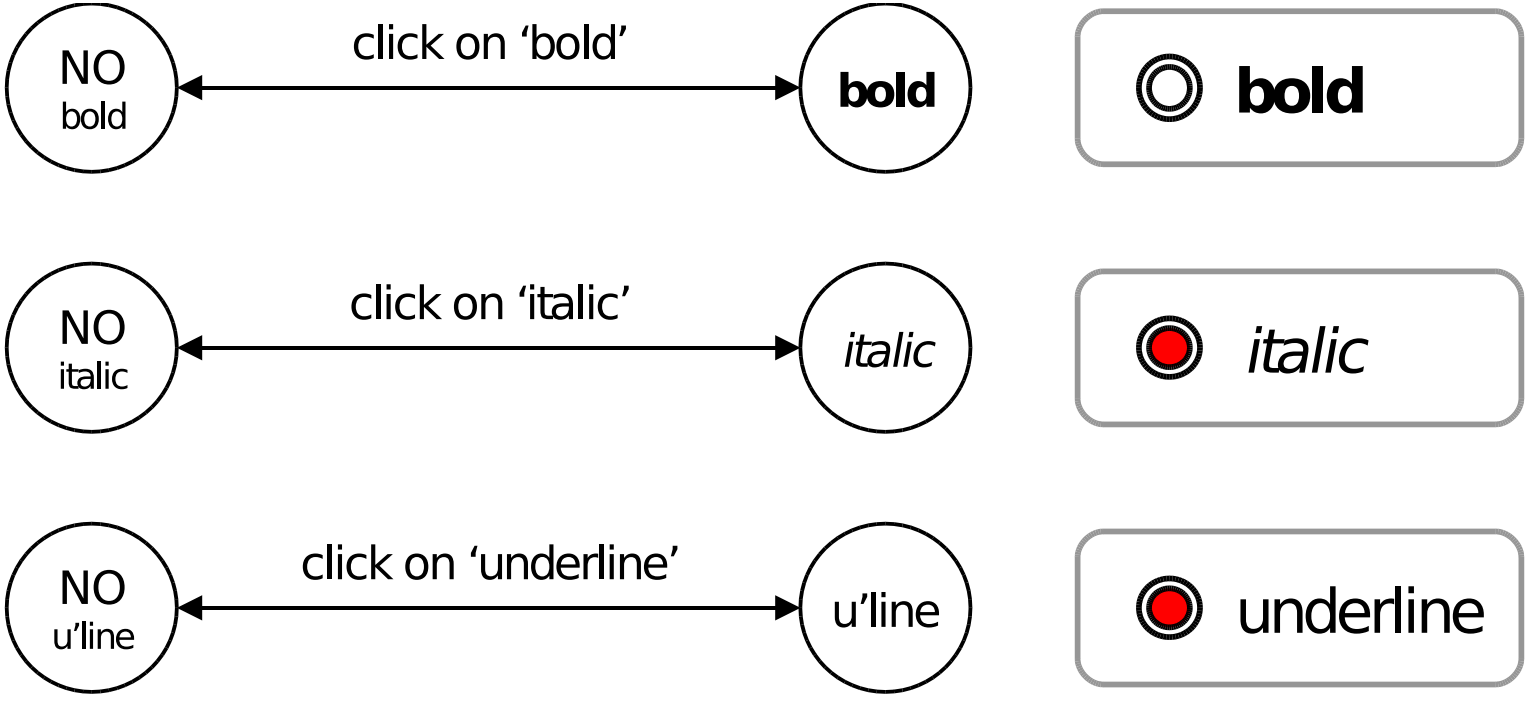


Hierarchical STNs

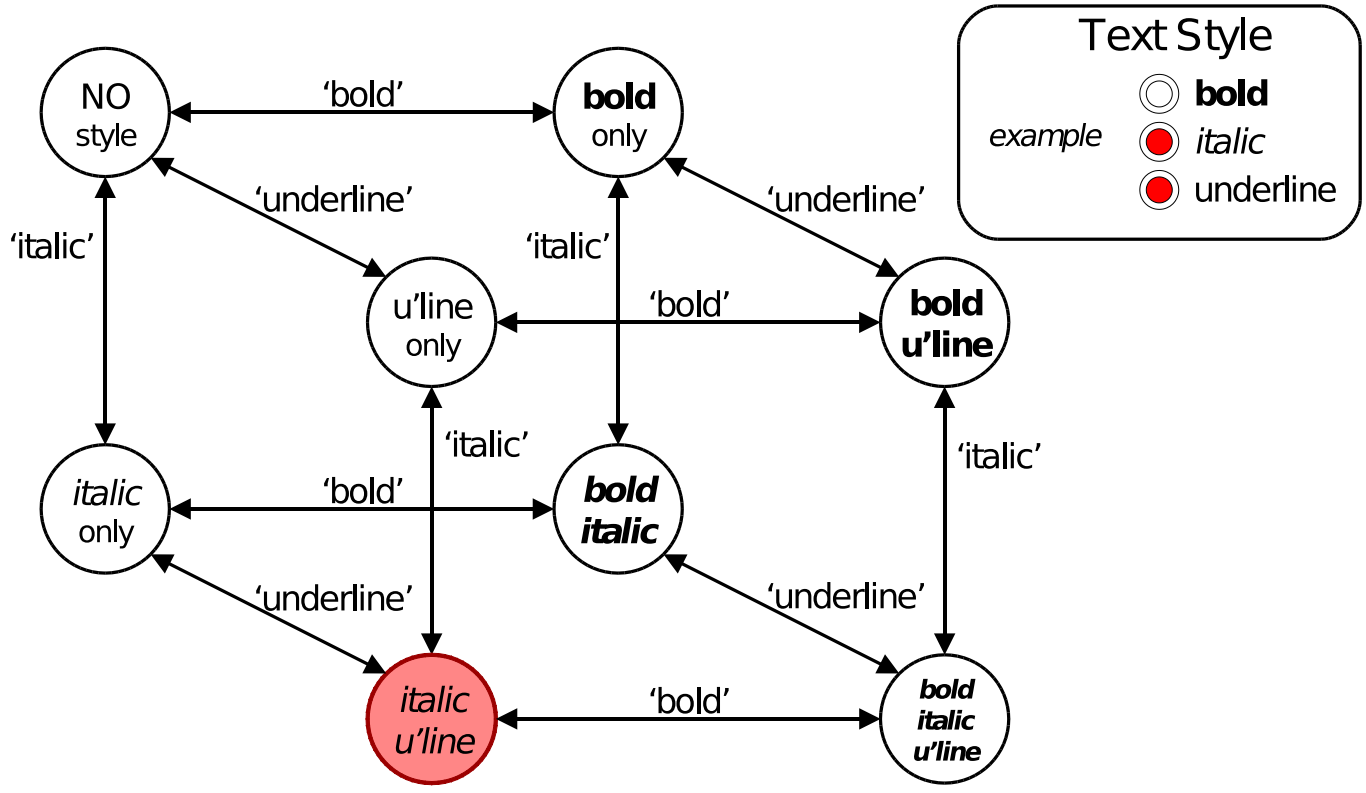


- Combining all operations would give clutter
- Simple structuring solves this, but what are the problems?

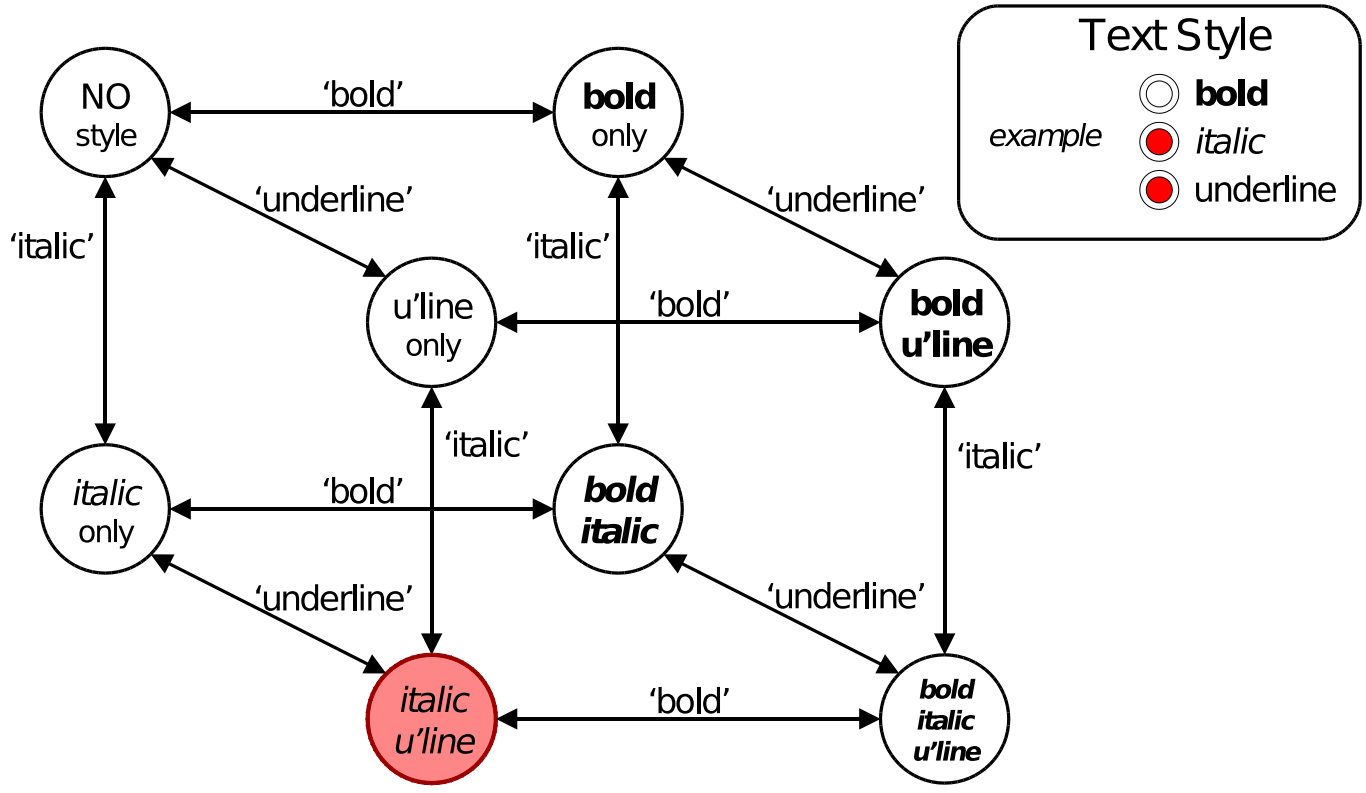
STNs for toggles



STNs: concurrency problem



STNs: concurrency problem



- Representing toggles concurrently leads to state explosion
- Inherent problem: needs a richer notation (e.g. UML state diagrams) or convention (separate STN for "microdialog" in DM).

Grammars

- Regular expressions useful for making compound actions, e.g.

$$selectline + click + click^{*} + doubleclick$$

  Same computation model as JSD.
- BNF and extensions:
  - good for low-level detail, command line syntax
  - more powerful than STNs

BNF with “visual terminals”

```
MENU ITEM SELECT  :=  point to item +
                      mouse down + MENU RESPONSE
MENU RESPONSE     :=  invert item | blink item
```

- Grammars may have cognitive validity
- Still not good for concurrency, pervasive commands

NB: non-standard + used for sequence

Process Calculi

CSP dialogue specification

```
Adder    = add-prompt!  →
             (quit?  → skip []
              zero?  → show(total) → Adder []
              num?   → show(total + num) → Adder)
Database = db-prompt!  →
             (quit?  → skip []
              set?   → Getkey ; Getval []
              get?   → Getkey ; Printval)
Getkey   = key-prompt!  → getkey?
Getval   = val-prompt!  → getval?
System   = Adder ‖ Database
```

→	event guard
;	sequence
[]	choice
‖	parallel

Outline

Dialogue Analysis

We can use descriptions to check some precise properties, of individual states or whole dialogue:

- **Completeness**
  - What happens on event X in state Y?
- **Reversibility**
  - How do we reverse action Z (e.g. “select line”)
  - . . . maybe navigation through dialog; *not* undo
- **Reachability**
  - Can you get anywhere from anywhere?
  - How easily?
- **Dangerous states**
  - Some states *should* be hard to get to
  - Perhaps guarded by warning dialogue
  - . . . although obvious problem if overused

Dialogue Analysis, continued

We can also analyse descriptions informally:

- check style guidelines, usability requirements
- consider *lexical* syntax
  - differentiation and visibility of modes/states
  - verb-noun (menu style) versus noun-verb (direct)
  - physical layout (e.g. key sequence convenience, accidents)
  - not independent of dialogue
- consider semantic intention
  - ways of attaching/checking semantics
  - maximising syntactic description

Outline

Exercise: Dialogue Notation

1. Pick your favourite application program
   — a word processor, drawing program, web browser
2. Considering the high-level static structure
   - give a fragment of a screen hierarchy diagram
3. Considering some low-level interaction structure
   - enumerate some input events and interface reactions
   - produce some hierarchical STNs
4. What did you find difficult to capture? Do your diagrams help you suggest any improvements to the program's interactions?
5. Many programs allow multiple windows (e.g. showing documents, or tool options) at once. Investigate ways of capturing this using dialogue notations.

Outline

References

These slides are mainly based on:

- Dix et al, Chapters 5 (esp. 5.6), 16.