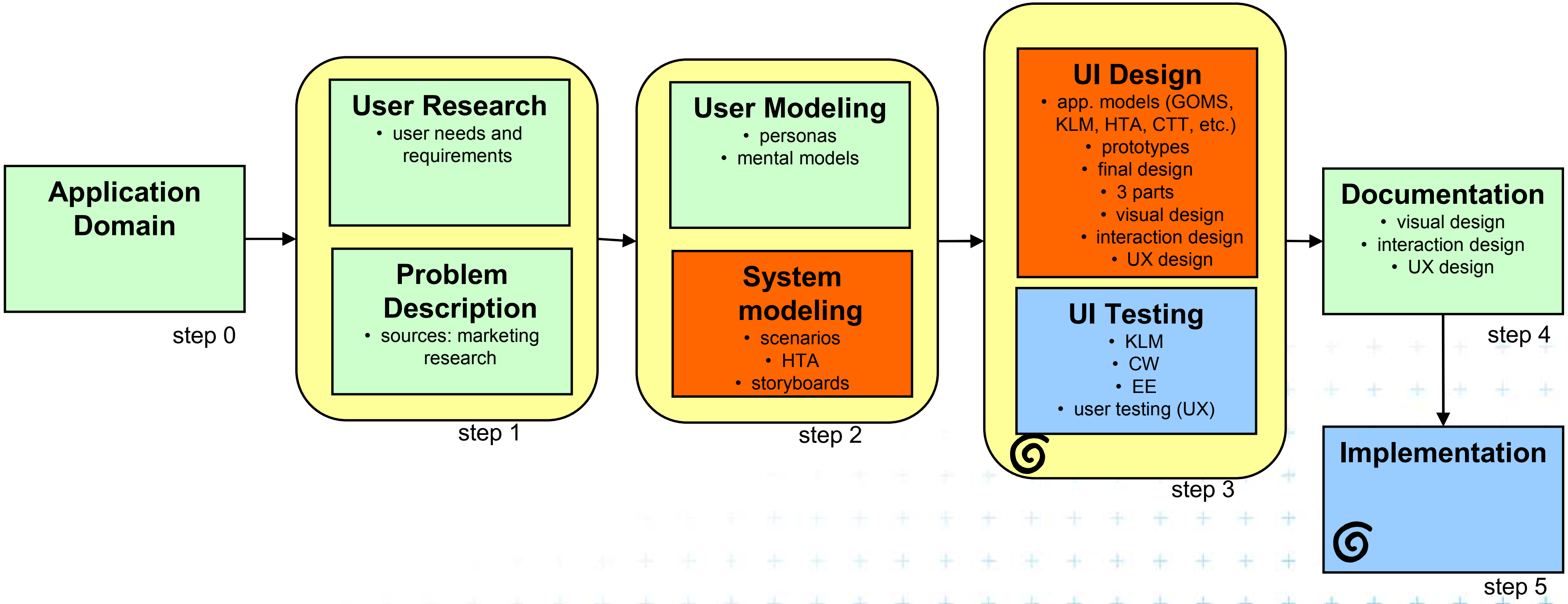
DEPARTMENT OF COMPUTER GRAPHICS AND INTERACTION

Formal description/models of user interfaces

Flow chart, Petri nets, STN, JSD

Models for UI description

- Model 1 ???
- Model 2 ???



What do we need more?

- We need to determine how the communication between computer and human will look like when performing individual steps
- We need to have at disposal formal description of a dialog structure

Dialog modeling

- Dialog can be represented as a set of states with transitions between them
- The advancement in dialog is linked up with the term - *current* state
- Transition between states can be dependent on condition
- It is possible to assign description of actions to individual transitions (e.g. change of the screen content)

Dialog and its structure

What will be discussed in further

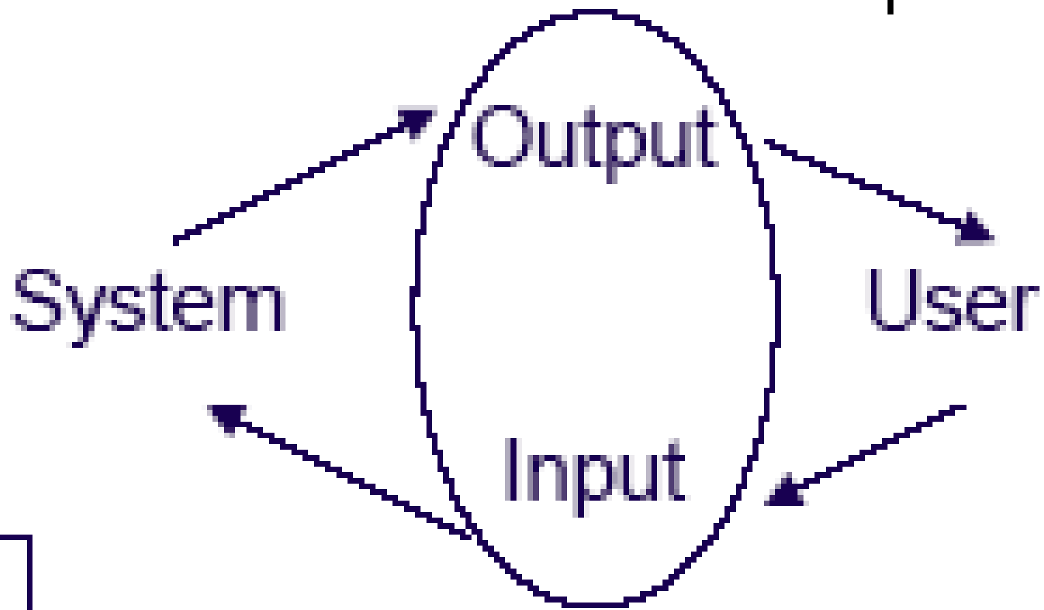
- Let us show that the dialog can have some kind of structure
- We can show that such a structure can be described in a formal way
- We can even show where the benefits of such an approach are

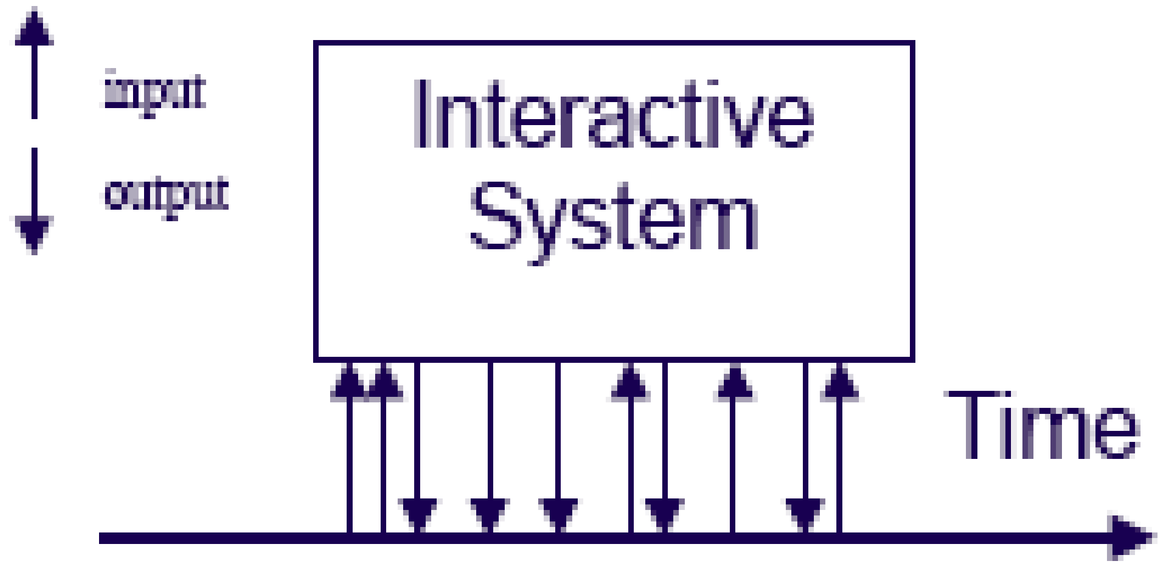
Do we understand what a dialog is?

- Dialog can be understood as syntactic level of communication between human and computer.
- Notation for dialogue description
  - diagrammatic
  - textual
- Dialog is linked-up with
  - Semantics (usually linked up with application)
  - presentation (to the user)
- Advantages of formal description

User and interactive system

I





What is dialog?

- Most of dialogs are not structured (structure = structure of a sentence => not sufficient).
- Examples of structured conversation: movie scenario, wedding ceremony,…
- Real dialog with computer is usually (somehow) structured and limited (not like in *Star Trek*).

Structured dialog between people

- Dialog is usually limited and formal
- Example - marriage

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

Typical features of a dialog

- Wedding ceremony
  - Given scenario for 3 participants
  - Sequence of “actions” is given
  - Some parts are fixed „I do“
  - Some parts are variable– “do you *man’s name* …”
  - What to do with the ring (with words “with this ring …”)
- When telling these words – are we married?
  - Only on the right place with the license (minister)
  - Syntax only – not semantics
  - What if some other answer will be said?

What is dialog?

- *Structure of conversation* between the user and computer system.
- Languages have 3 levels
  - lexical
  - syntactic <-- most of user interfaces
  - semantic
- Description of a language must be linked-up with semantics (because of implementation – example? …… e.g. functionality in CAD system)

Dialog - formal description

Notation for dialog description

- Other branches of computer science (structured dialogue –> specialized language –> formal description –> theory of languages)

- What about to use programming languages for dialog description?

Programming language as a tool for dialogue description

- NO!!
- Why?
- The application part (e.g. simulation of fluid flow) is mixed up with user interface
- Problem?
- YES!!
- E.g. maintenance, modification …..
- This topic is known from **other courses** (Software engineering etc…)
- The UI part will be separated from application (and described formally)

Typical dialog notations

- Textual
- Diagrammatic

Diagrammatic notation

- Frequently used (picture gives us a nice overview)
- Dialog structure – at the first glance
- What to do with large and complex dialogues
- Typical diagrams used
  - STN - State transition networks   (STD)
  - Petri nets
  - Flowcharts
  - JSD diagrams

Textual description

- Non-formal description (in common language)
- Grammars
- Some other theoretically based descriptions (production rules ...)

Why to describe dialog?

- The purpose:
  - Communication with other designers
  - Tool in early phase of design (brainstorming - ideas)

- How to embed semantics?
  - The users can take an active part in discussions
  - The users can suggest extension of functionality
  - We complete the dialog description by intended meaning (semantics) of a new action

Dialog model: State diagrams

How to work with system states

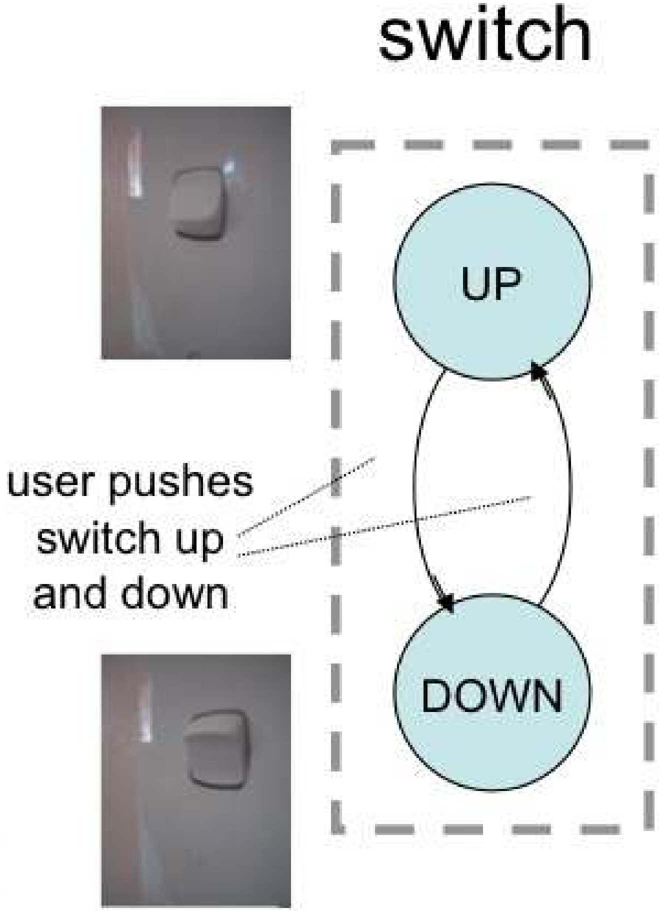
- A lot of formalisms exists
- State diagrams
  – transition diagrams
  – transition networks

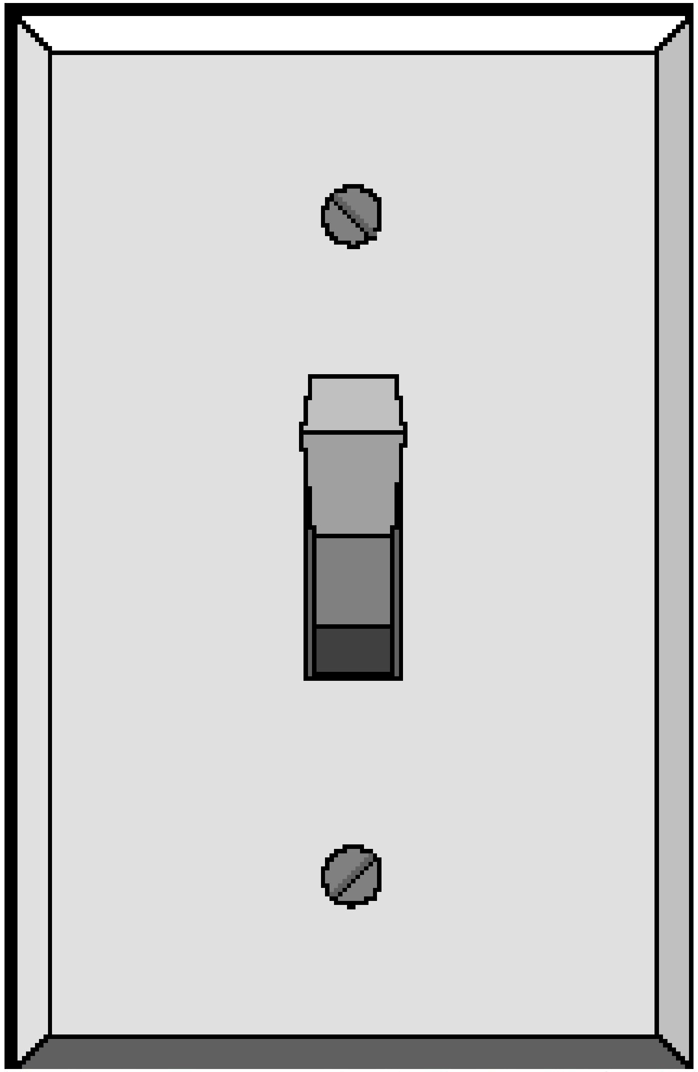
  – Principle: INPUT -> transition from the current state into new one

- Examples from everyday life?

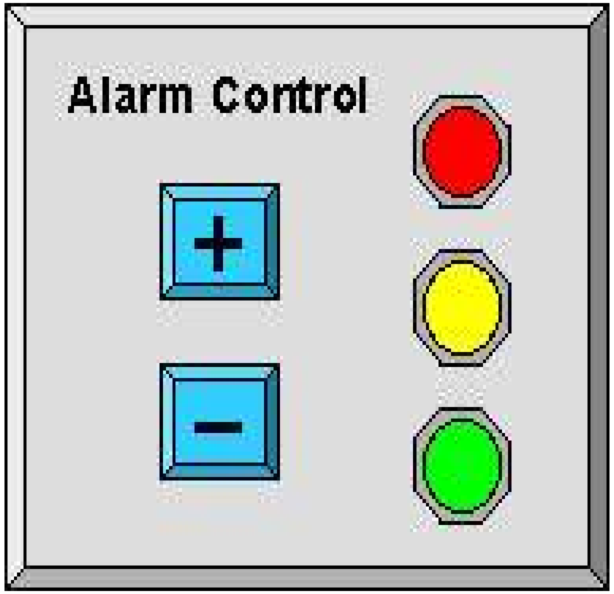
Example state: switch

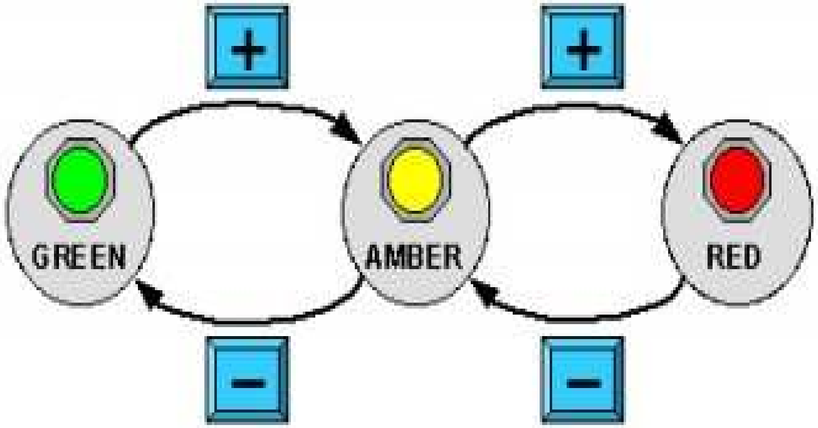
- state: off
- state: on





State diagram - example

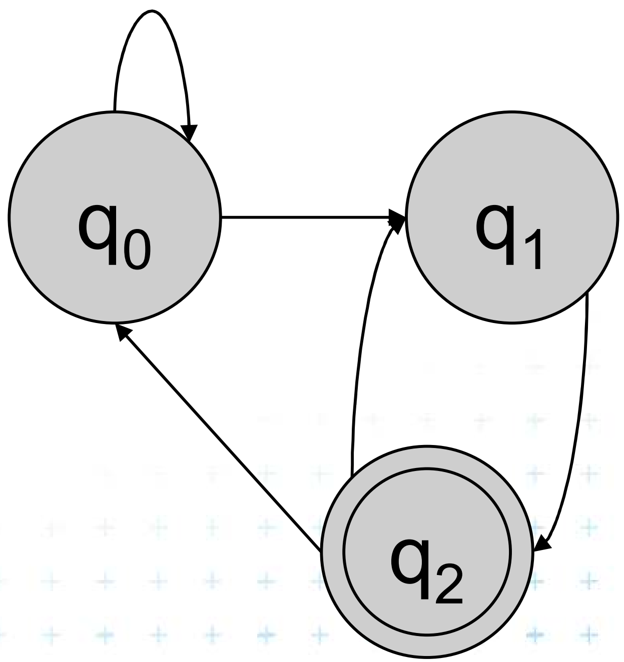




Alan J. Dix

FSA

- FSA is an object that has defined behavior by means of set of states and set of transitions between them
- FSA – picture: what is wrong?

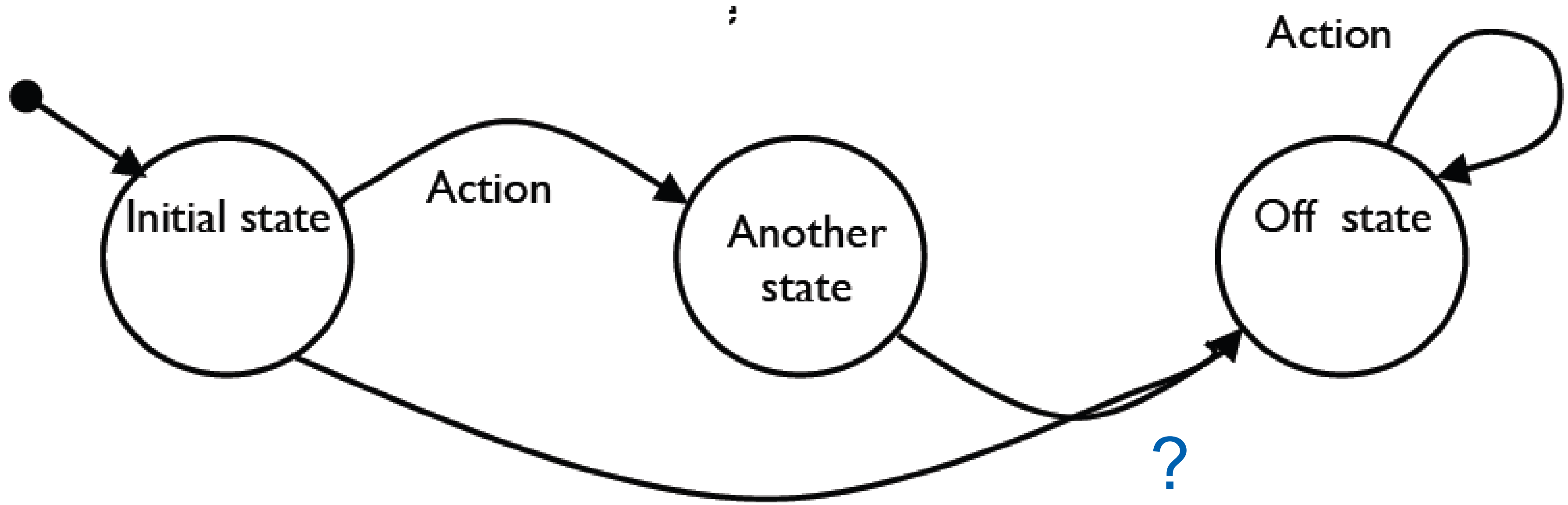


Why state diagrams?

- Formal description of UI behavior
- Dialog is represented as a set of states with transitions between them
- The course of dialog is linked-up with the *current* state
- Transition between states can be conditional
- Manifestation of transitions can be added
  – change of screen

Some rules

- States cannot overlap or intersect
- Have exactly as many arrows from a state as there are possible actions from that state:
- Each arrow is labeled with its action; when action name matches
  - consequent state might omit, e.g. ‘off’ action leads to ‘off’ state
- It may be convenient to merge arrows that go to the same state.

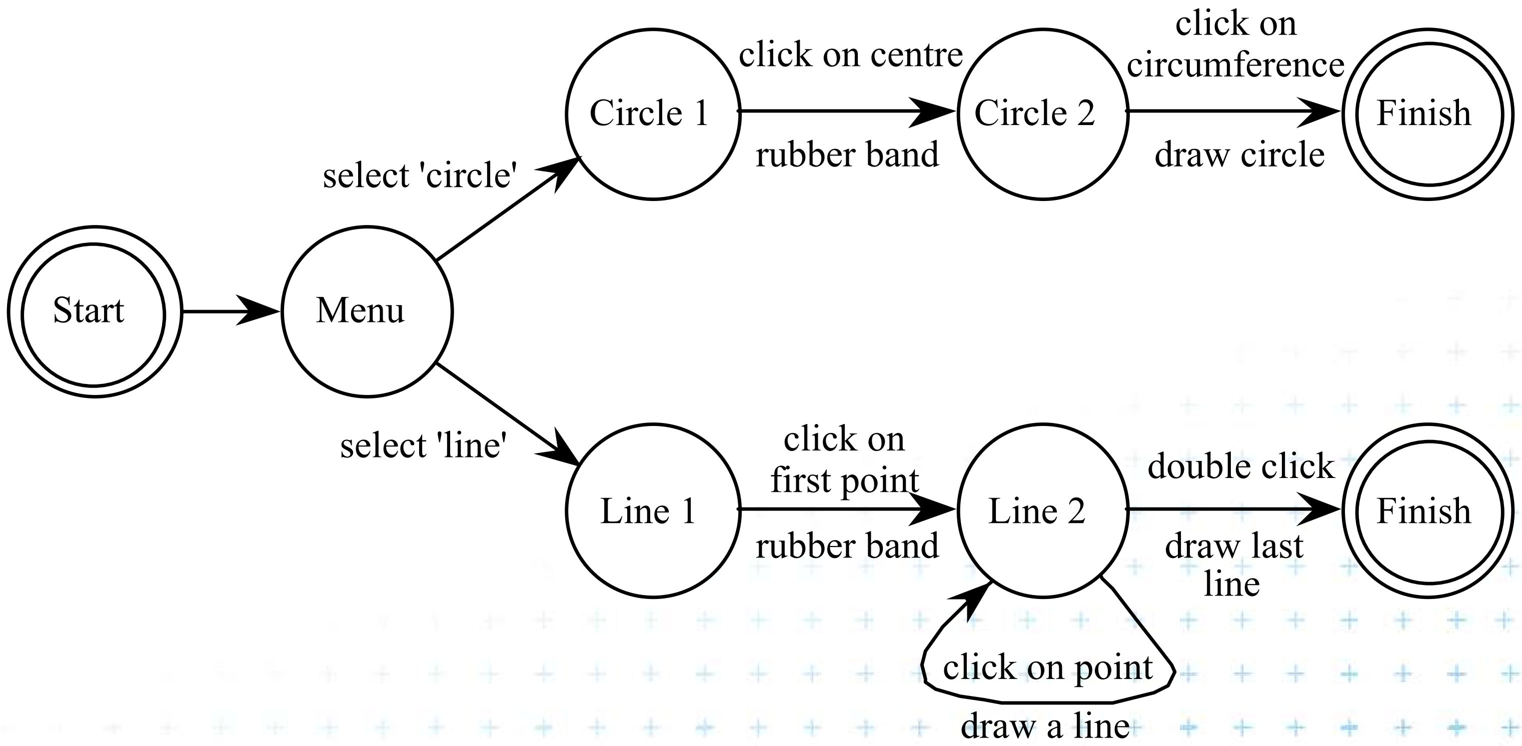


Dialog model

- Experiments with model (see in next slides)
  - distinguish between *control inputs* and *application inputs*
  - change of screen state
- Discovery of all possible paths in the model
  - check if all paths end up in proper states – not in a state that is not a final one and has no output etc..)
  - automatic check (graph algorithm?)
- Possibility of automatic (or semiautomatic) UI creation

State transition networks (STN)

- circles- states
- edges - action/events

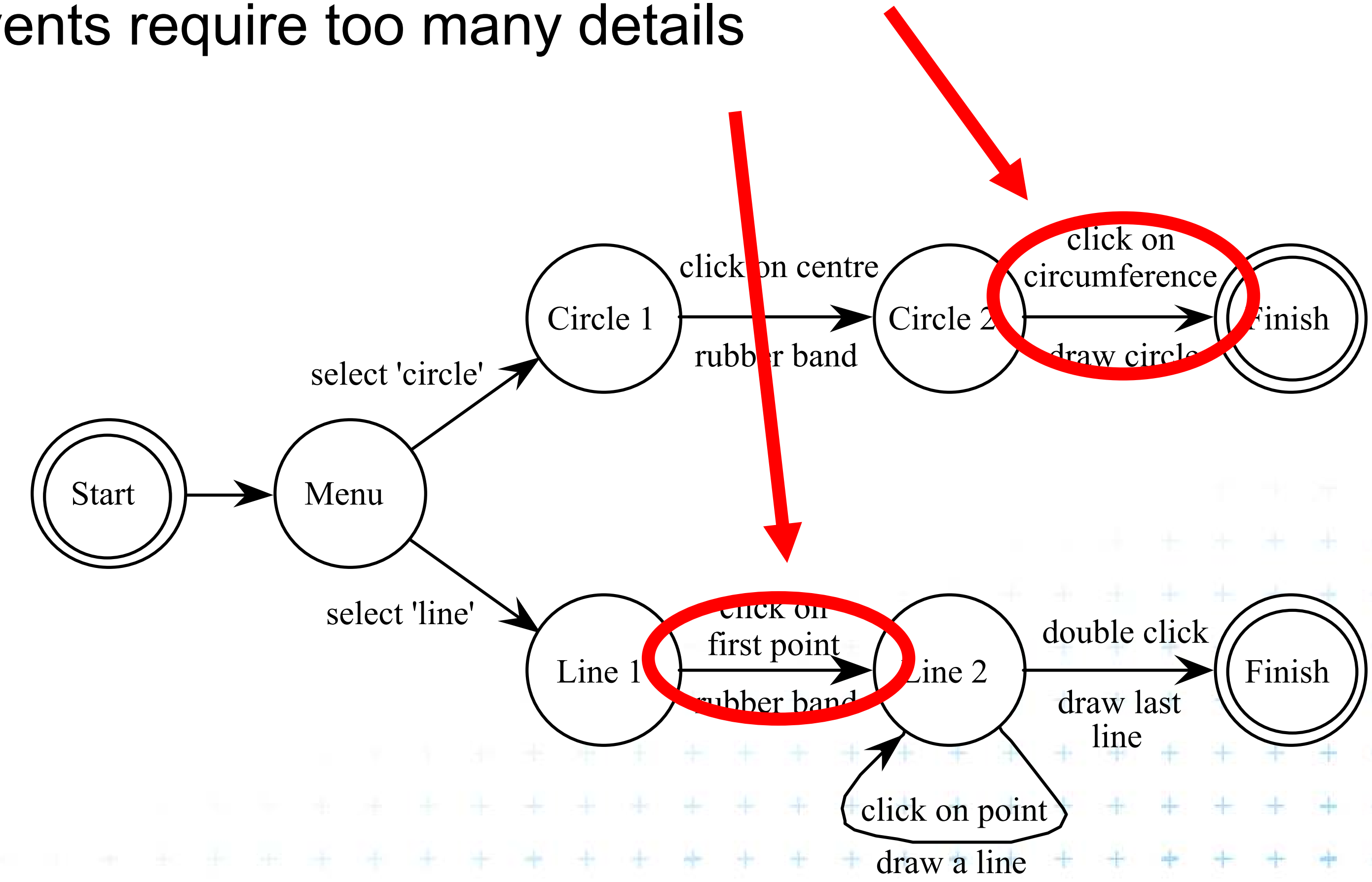


State diagrams = models

- What brings us the use of these models?
- Simulation of interaction
- Check the user ability to cope with UI
- Check the functionality

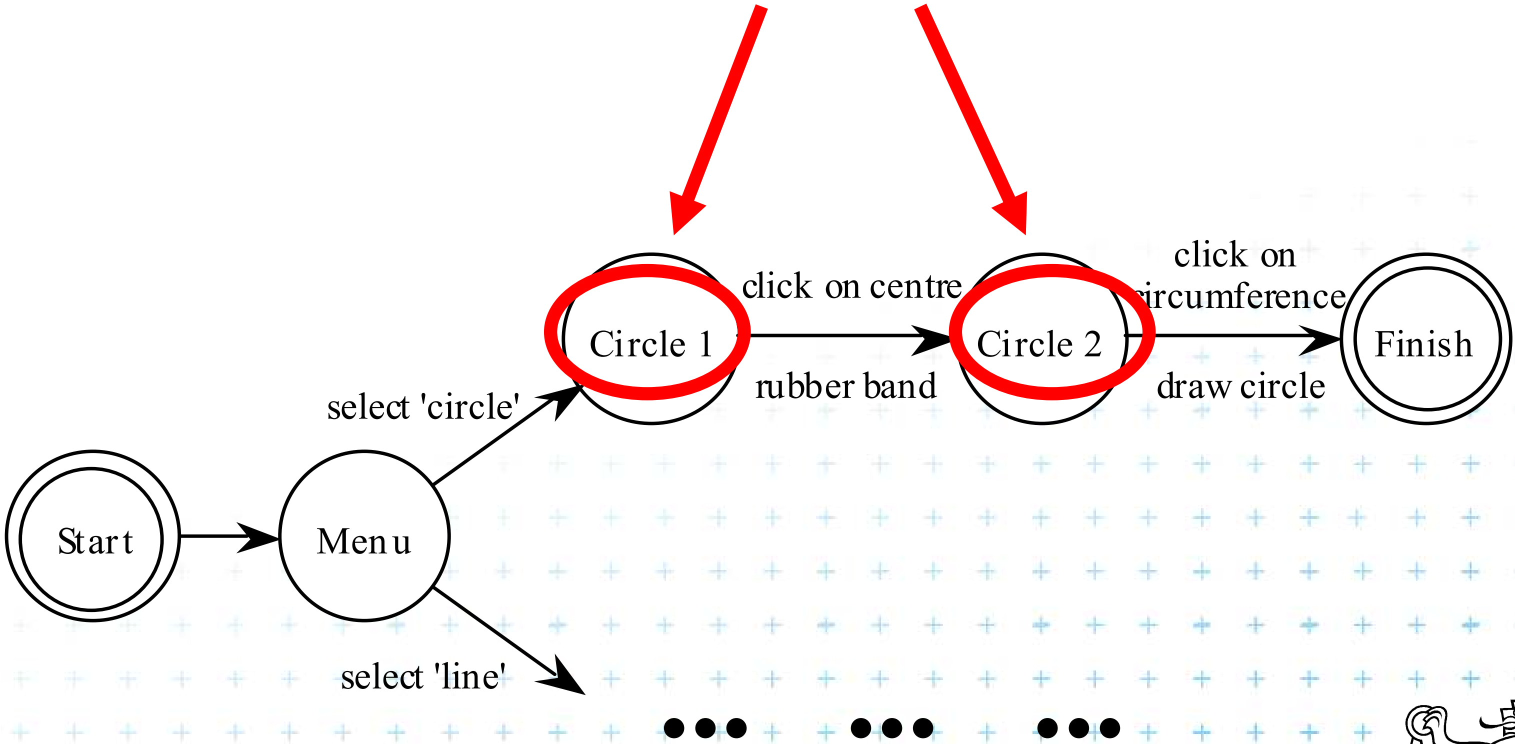
State transition networks - events

- Transitions are hard to read and interpret:
  - Notation includes a lot of states ('state heavy')
  - Events require too many details

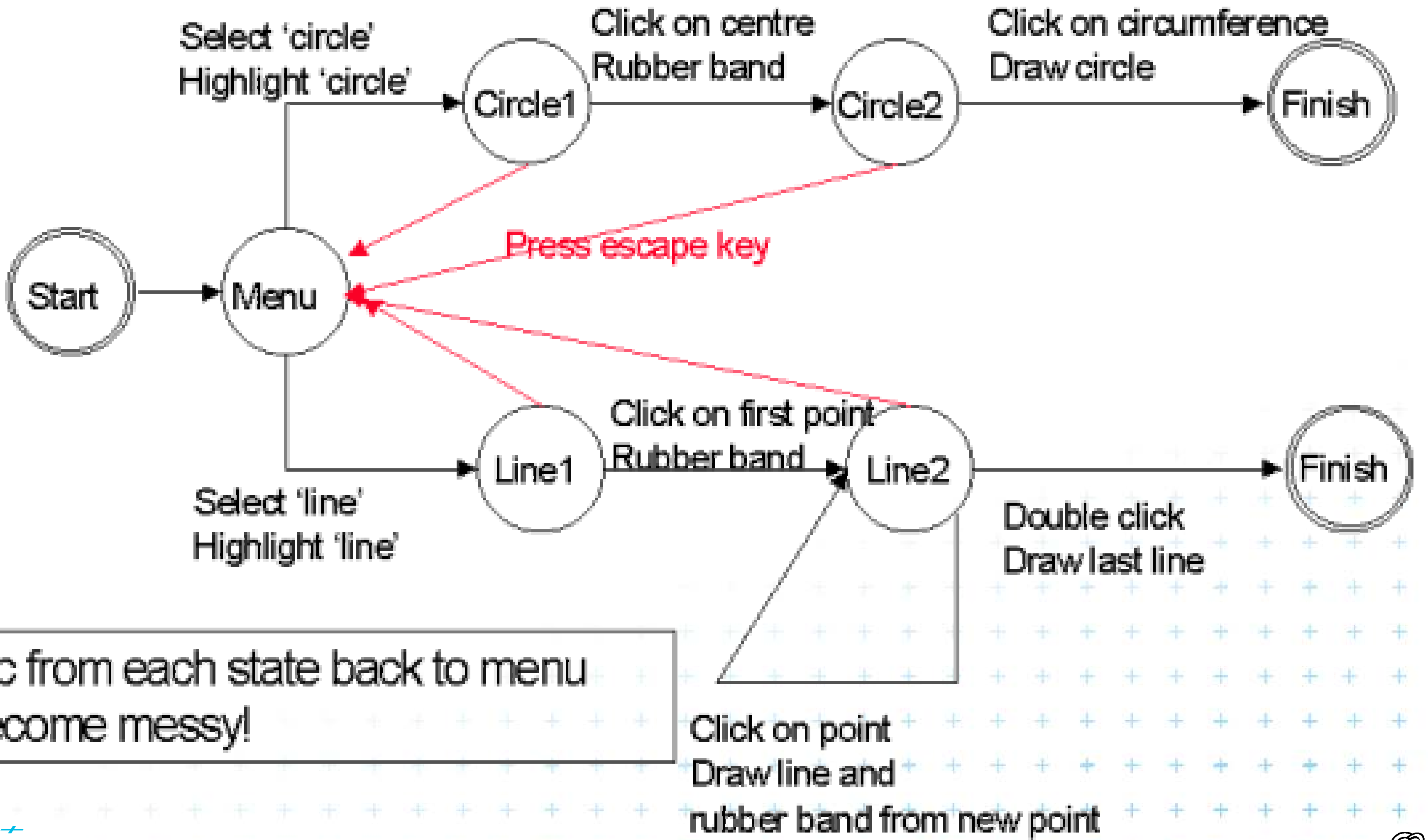


State transition networks - states

- Circle notations are rather unintuitive
  - States are hard to name
  - They can be drawn easily



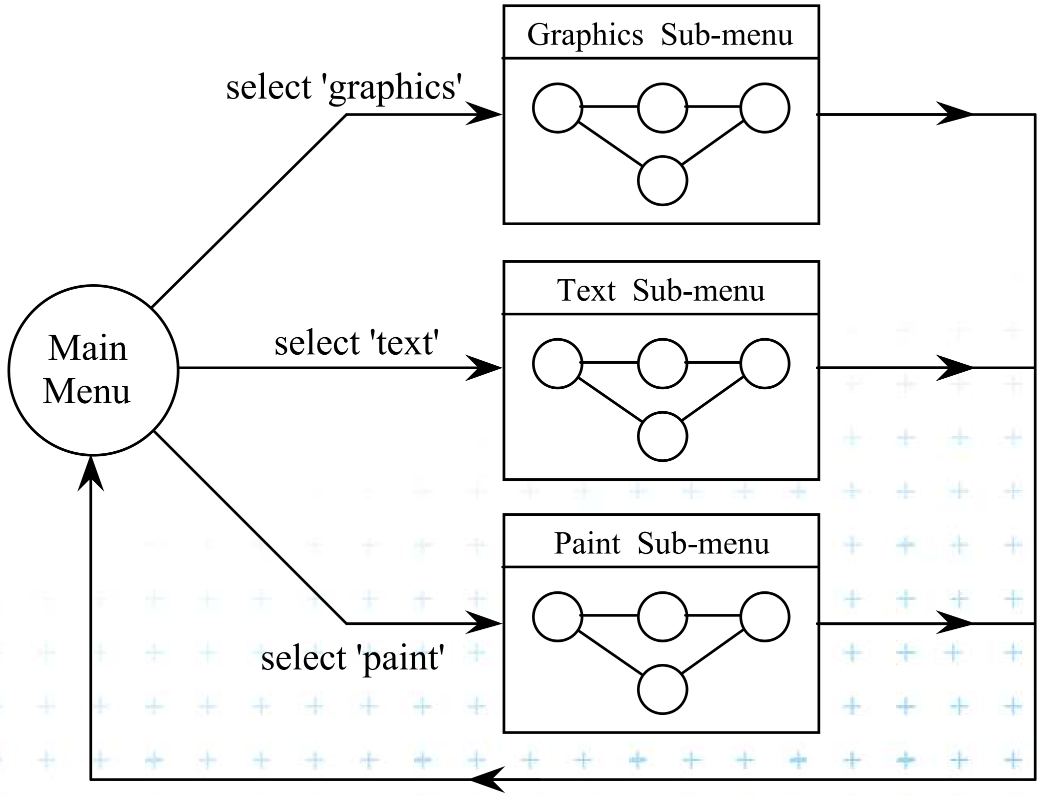
State transition network - transitions



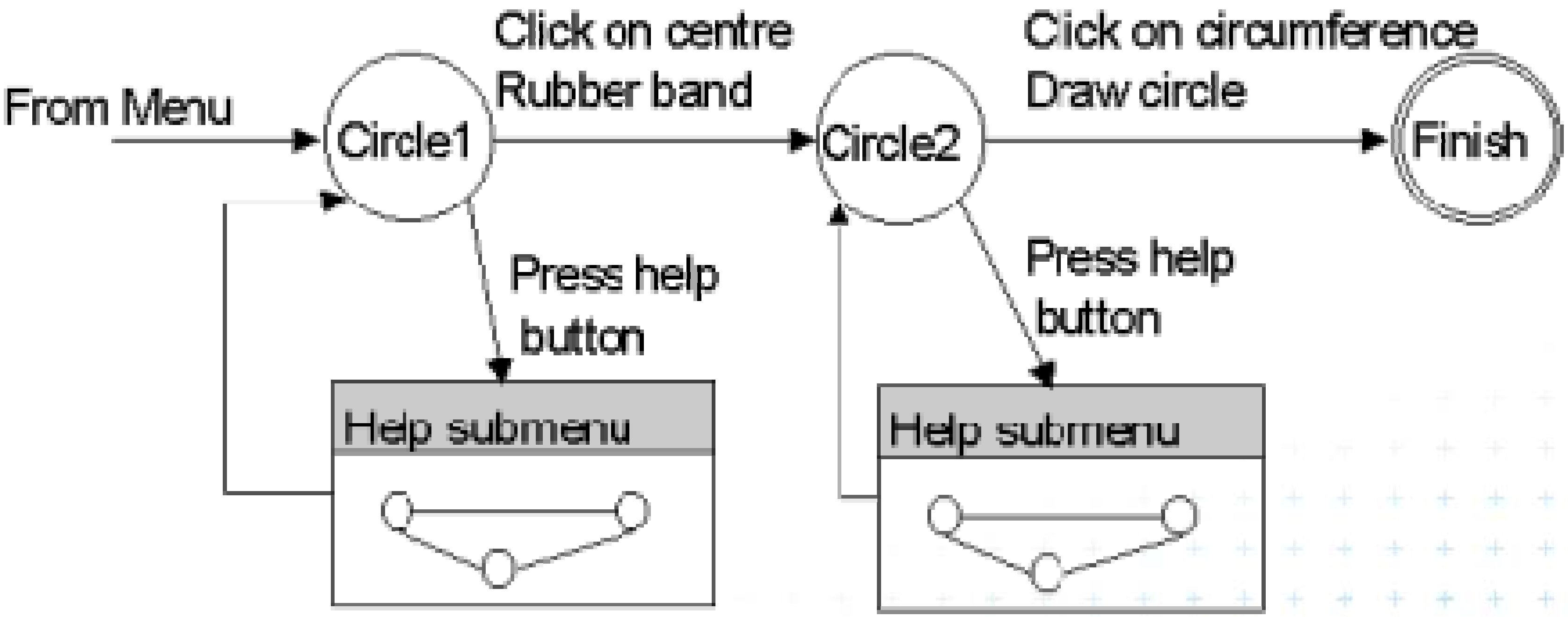
Arc from each state back to menu
Become messy!

Hierarchical STN

- Complex dialogs can be described
- Naming sub-dialogs



Hierarchical state transition network



Augmented transition networks (ATNs)

- Form of STN
- We assume existence of several "registers" that are set before transition (and tested afterwards)
  - if condition is true and event occurs, follow arc
- Example: How many times wrong PIN was used
  - three times – either three inputs or semantic approach: register that is tested each time – till the number 3 has been achieved
- Example: How many times we clicked when drawing the line

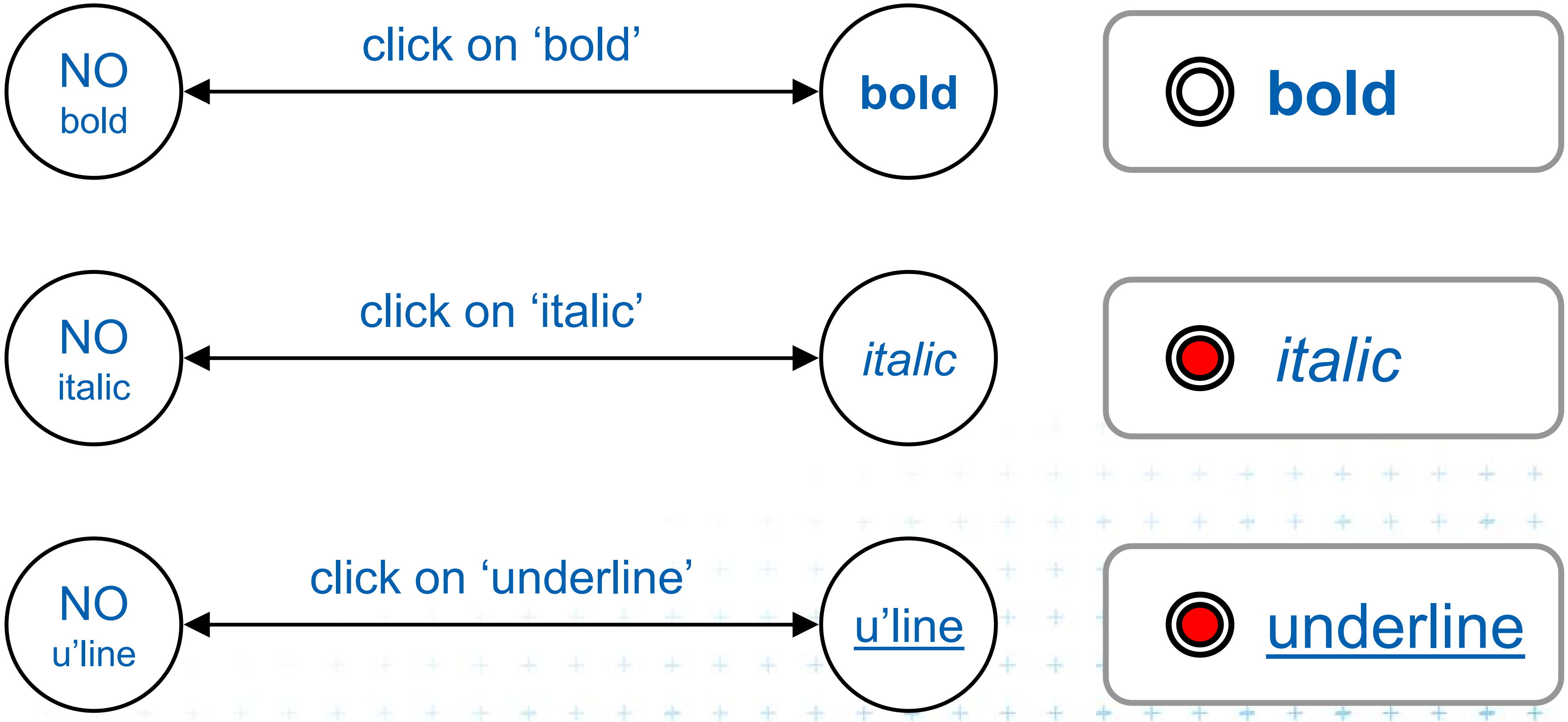
Relations between notations

- What about menu and STN?
- Mutual transformations?

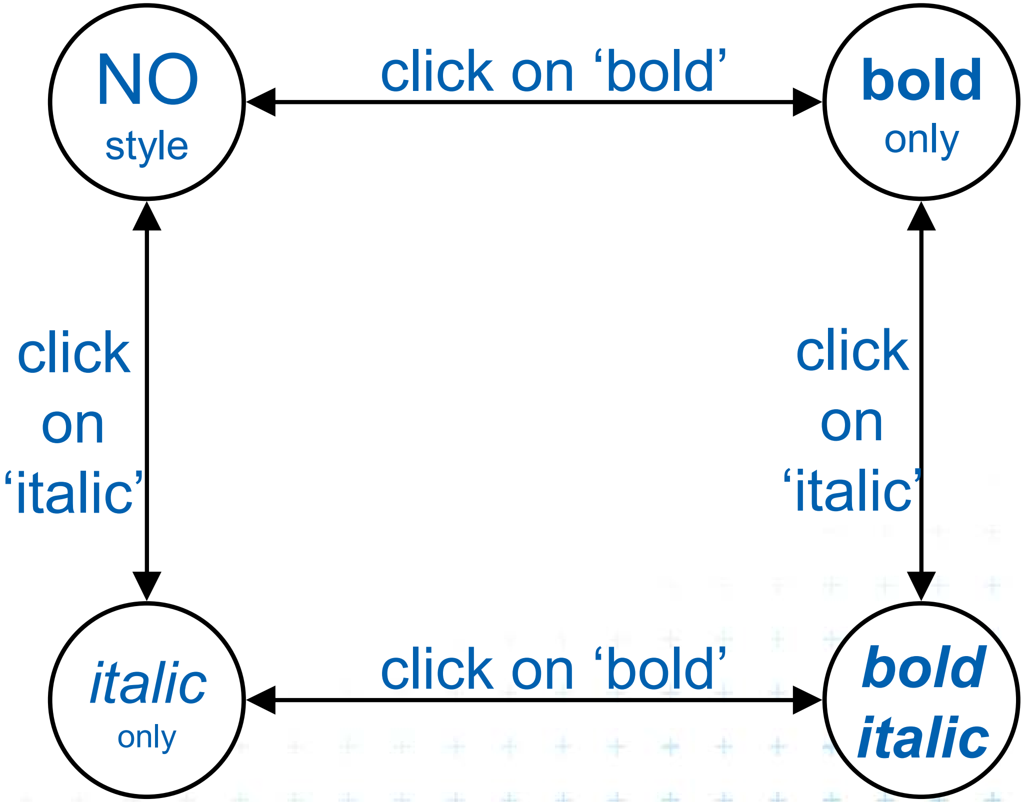
More complex examples

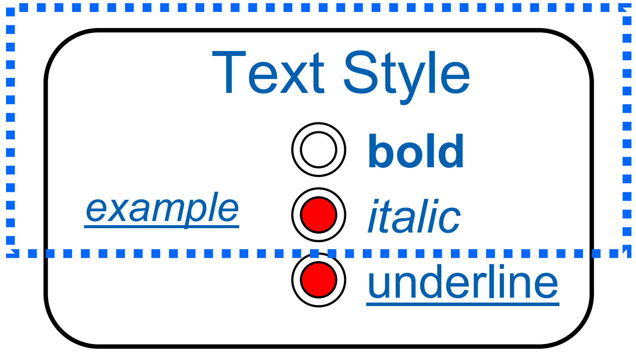
More dialogs in parallel

Concurrent dialogs - individual STNs

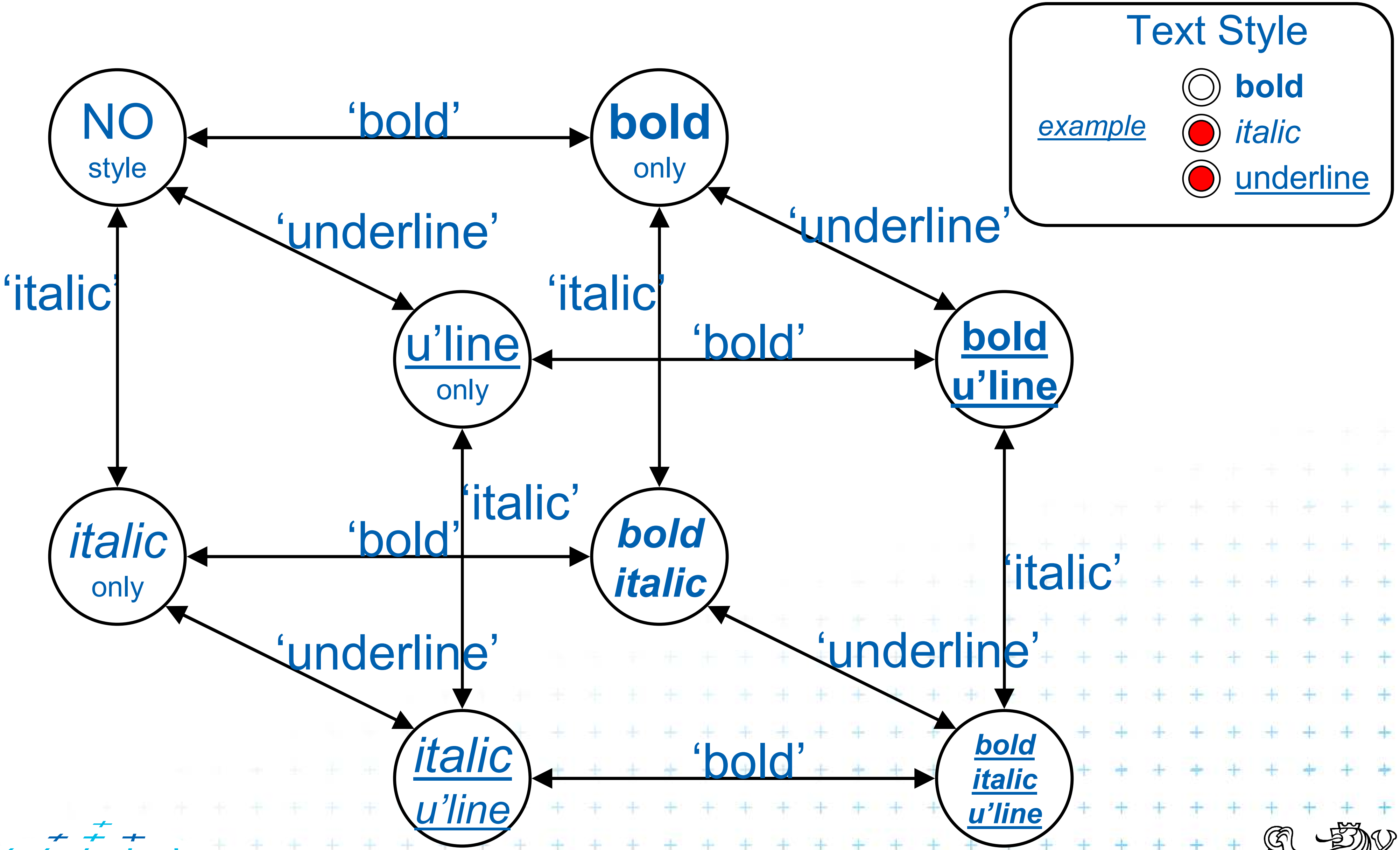


Concurrent dialogs – bold and italic combined





Concurrent dialogs - all together - combinatorial explosion



State properties

- Availability
  – Can we get from anywhere to anywhere?
  – How easy is it?
- Reversibility
  – Can we get to the previous state?
  – Not an UNDO
- Dangerous states
  – We do not want to get there

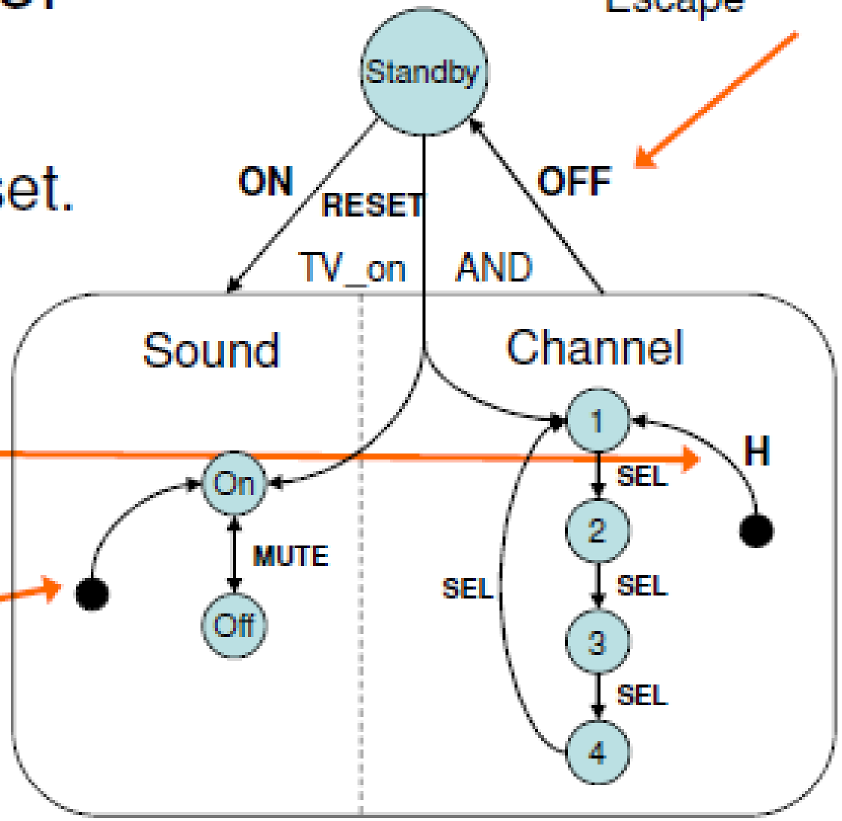
What about switching between dialogs?

- Example: TV remote control
- How to remember state (e.g. TV program number)?

TV remote control – History feature

- Example: TV controller
  - 5 buttons:
    - on, off, mute, sel e reset.

Works as Escape

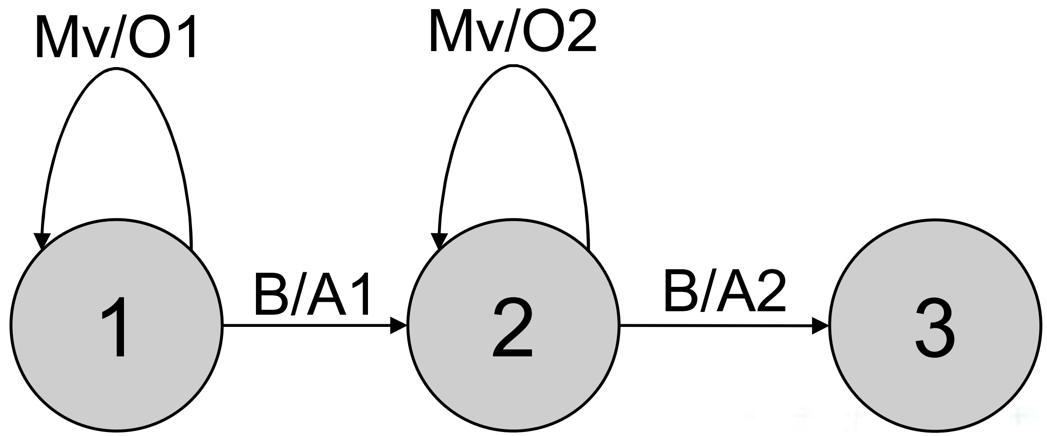


"history":
-1st time (or after "reset") starts on 1
- goes to former selected channel.

"start node" in the STN – represents the "default"

Home work: what interaction technique is this?

- Input events
  - B: mouse button click
  - Mv: cursor movement
- Application operation
  - A1: store start point
  - A2: store end point
- Output
  - O1: track cursor
  - O2: draw line from start to end point



Another homework

What is it? How to convert it into STN?

- Grammars
  - BNF (Backus-Naur Form)
    - Dialog syntactic level
    - Used widely to specify the syntax of computer programming languages.

sequence

  - Example: line-drawing function

draw-line	::=	*select-line + choose-point + last-point*
select-line	::=	*position-mouse + CLICK-MOUSE*
choose-point	::=	*choose-one*
		| choose-one + choose-point
choose-one	::=	*position-mouse + CLICK-MOUSE*
Last-point	::=	*position-mouse + DOUBLE-CLICK-MOUSE*
position-mouse	::=	*empty | MOVE-MOUSE + position-mouse*

choice

recursive definition

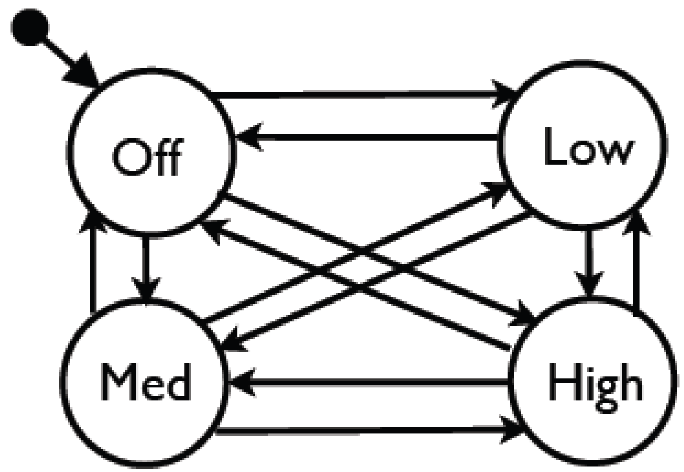
DCGI

Applying graph theory

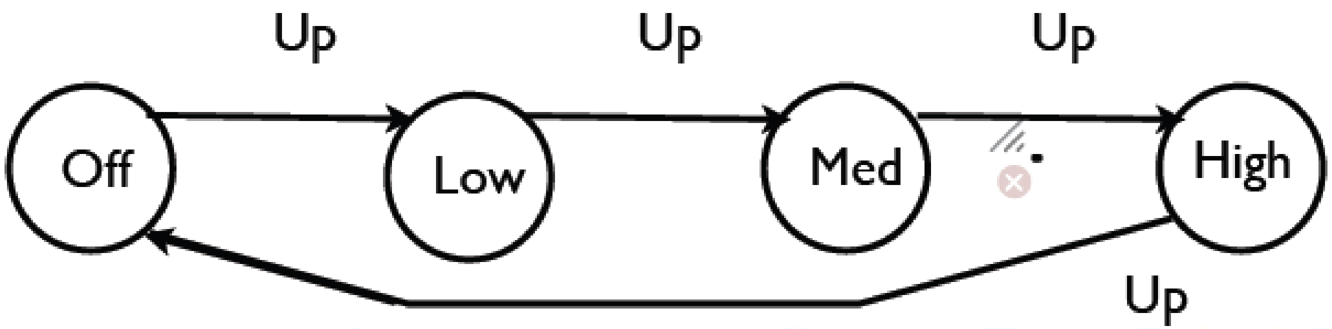
- An STN is a graph, so we can apply graph theory to analyse it
- Shortest route around graph that includes every arc
- Can use to efficiently check every action works as specified; and has a corresponding description in the manual
- Length of tour is a measure of how hard the device will be to test, document, understand or explore

Applying graph theory

- Connectivity:
- For most systems, a user should be able to get from any state to any other, i.e., the STN should be *strongly connected*



Cyclic graph – just one button.
We go through all states



Automating usability checks

- A state transition network is a finite state machine
- We can describe the device in a computer program:

  List of states

  List of actions
- Matrix of actions x states describing transitions

  Can automatically generate the transition diagram

  Can automatically find shortest paths
- Provide user instructions; generate the help manual

  Can check if some path lengths are unreasonably long

  Can make frequently used actions easier (e.g. larger buttons)

Testing STD

- Test to find errors in the design and the implementation

  The state transitions should be made visible during testing
- Check the action carefully
- Check the state carefully
- Check for dead states
- Ensure events that are not supposed to be *possible*, really cannot happen

Testing NOKIA

6 *Harold Thimbleby*

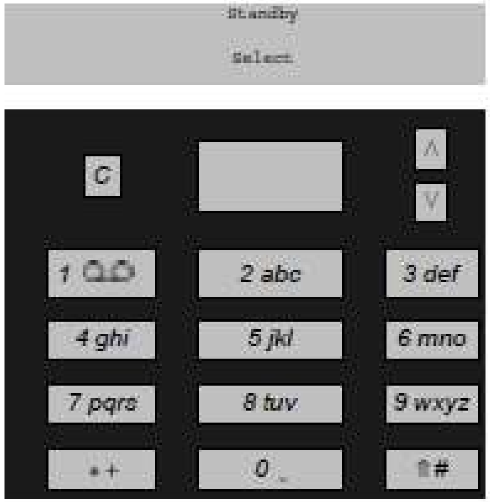


Figure 3: Simulation of the Nokia handset. The picture (which is full size in the *Mathematica* version of this paper) is active code and works when it is clicked.

Testing NOKIA

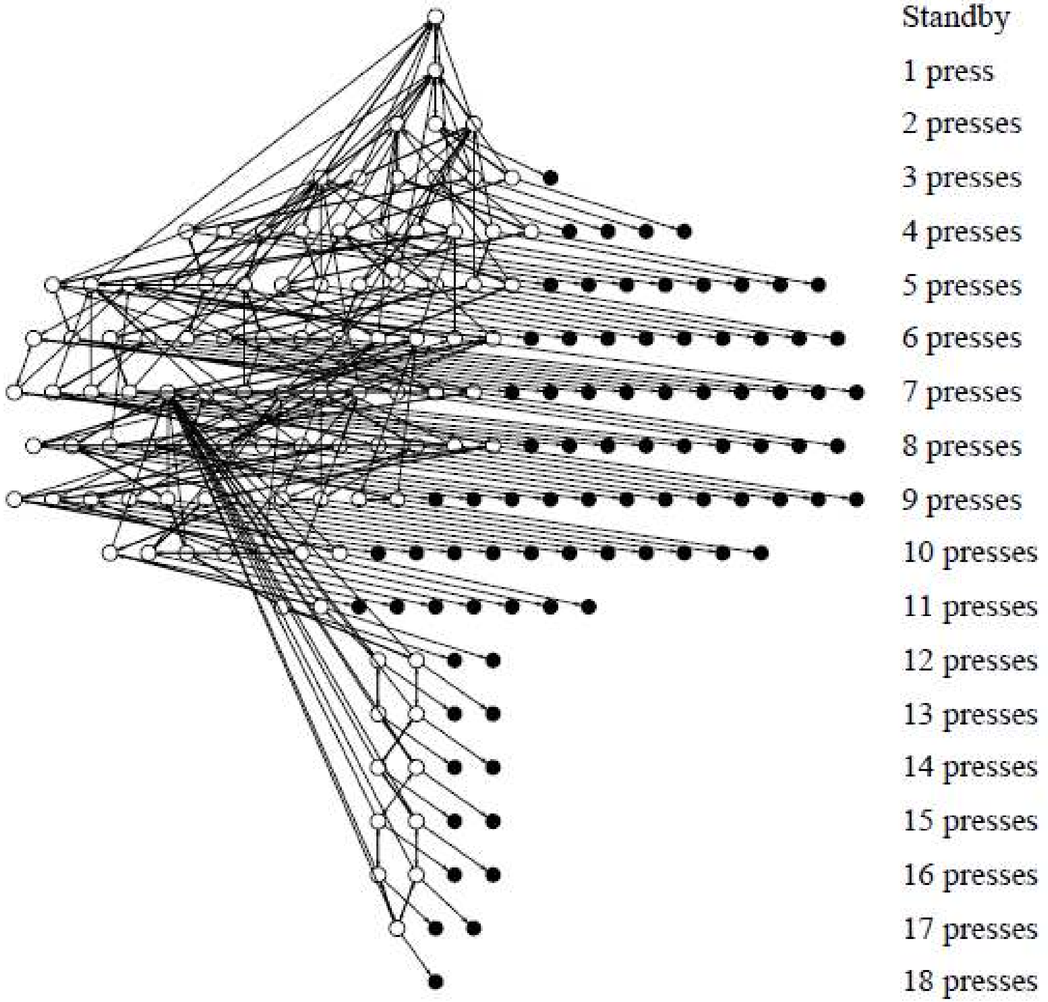
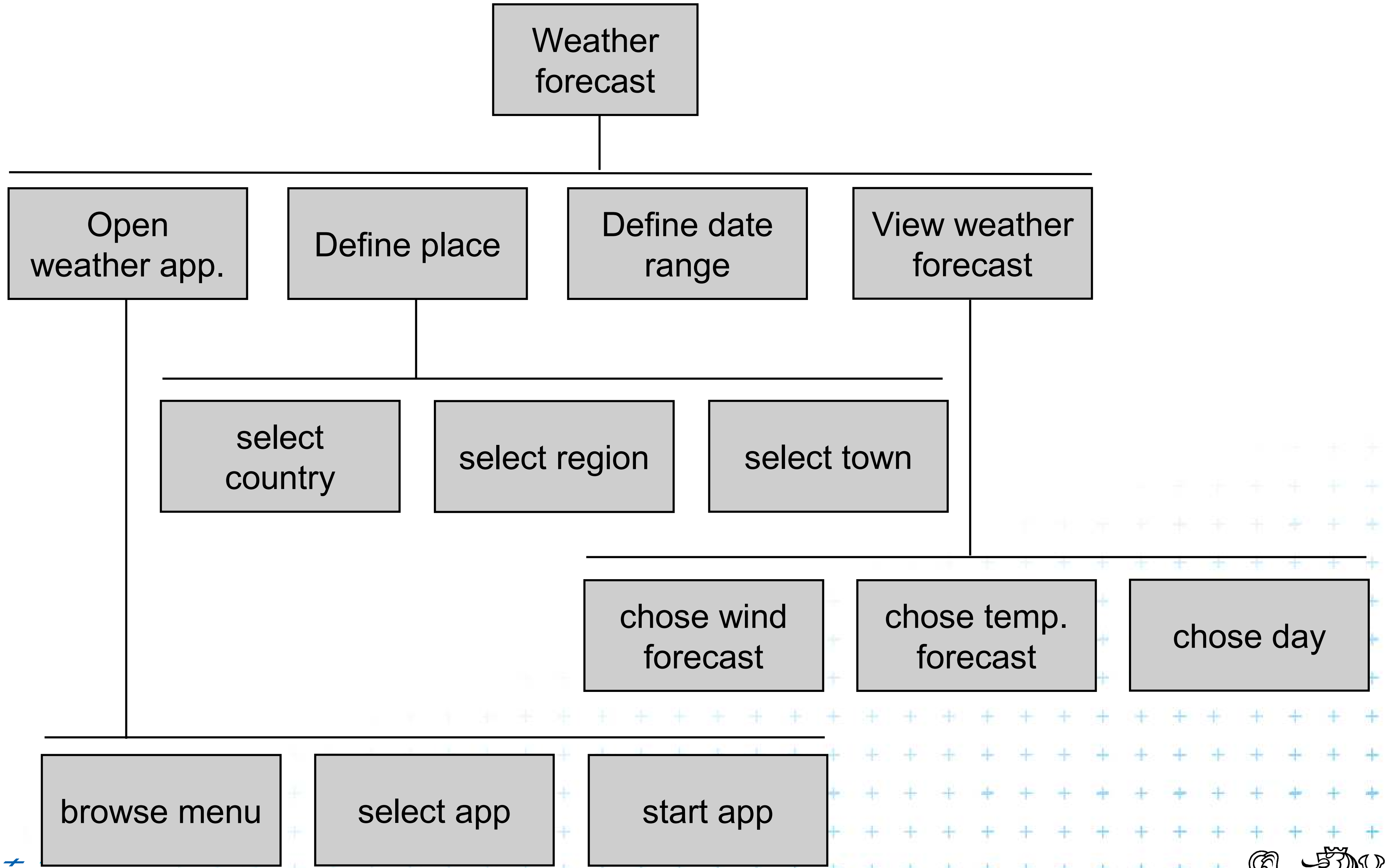
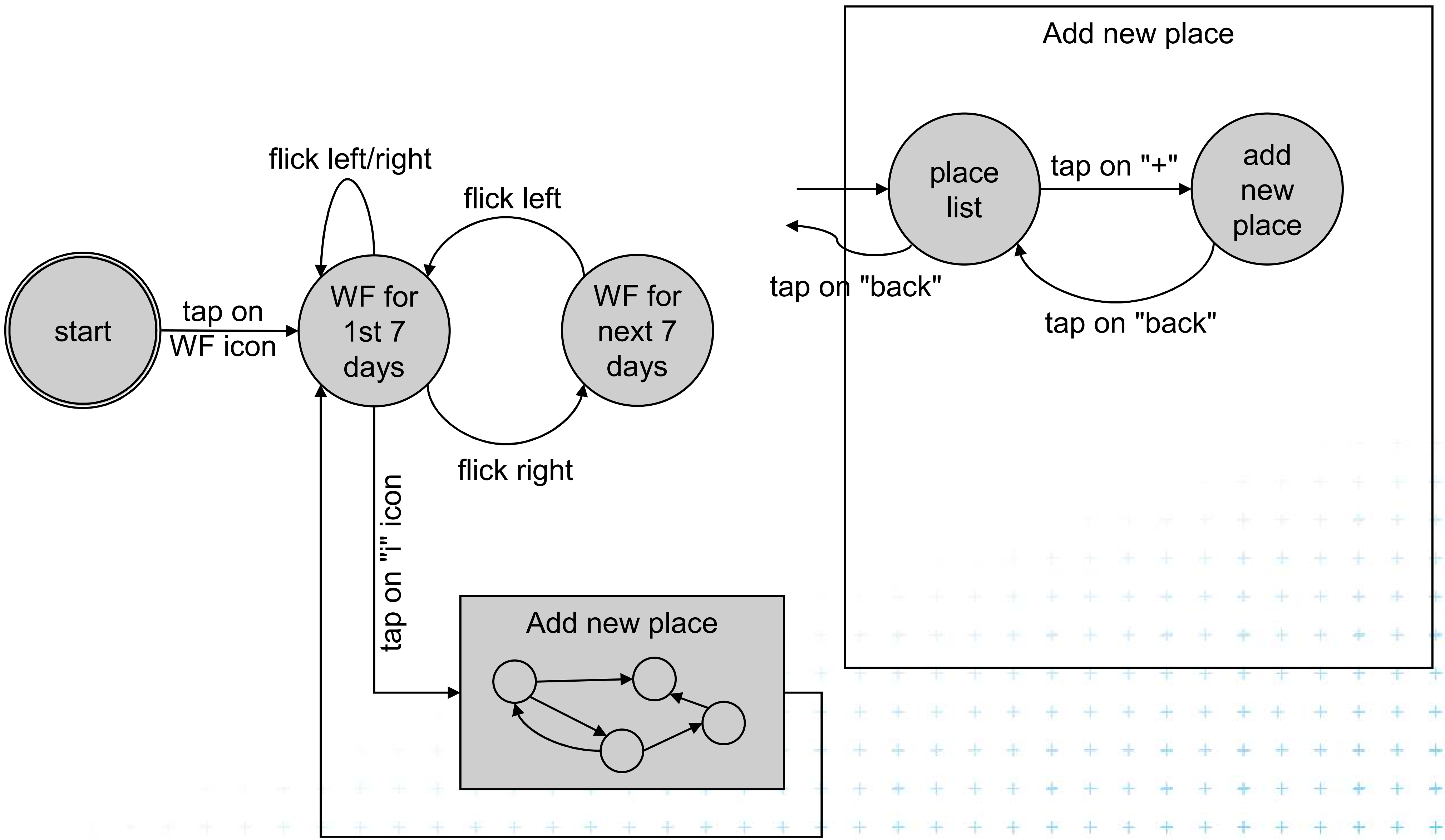


Figure 2: Visualising the error-free cost of accessing Nokia menu functions. For clarity arrows *from* functions (black dots) are not shown.

Weather forecast HTA



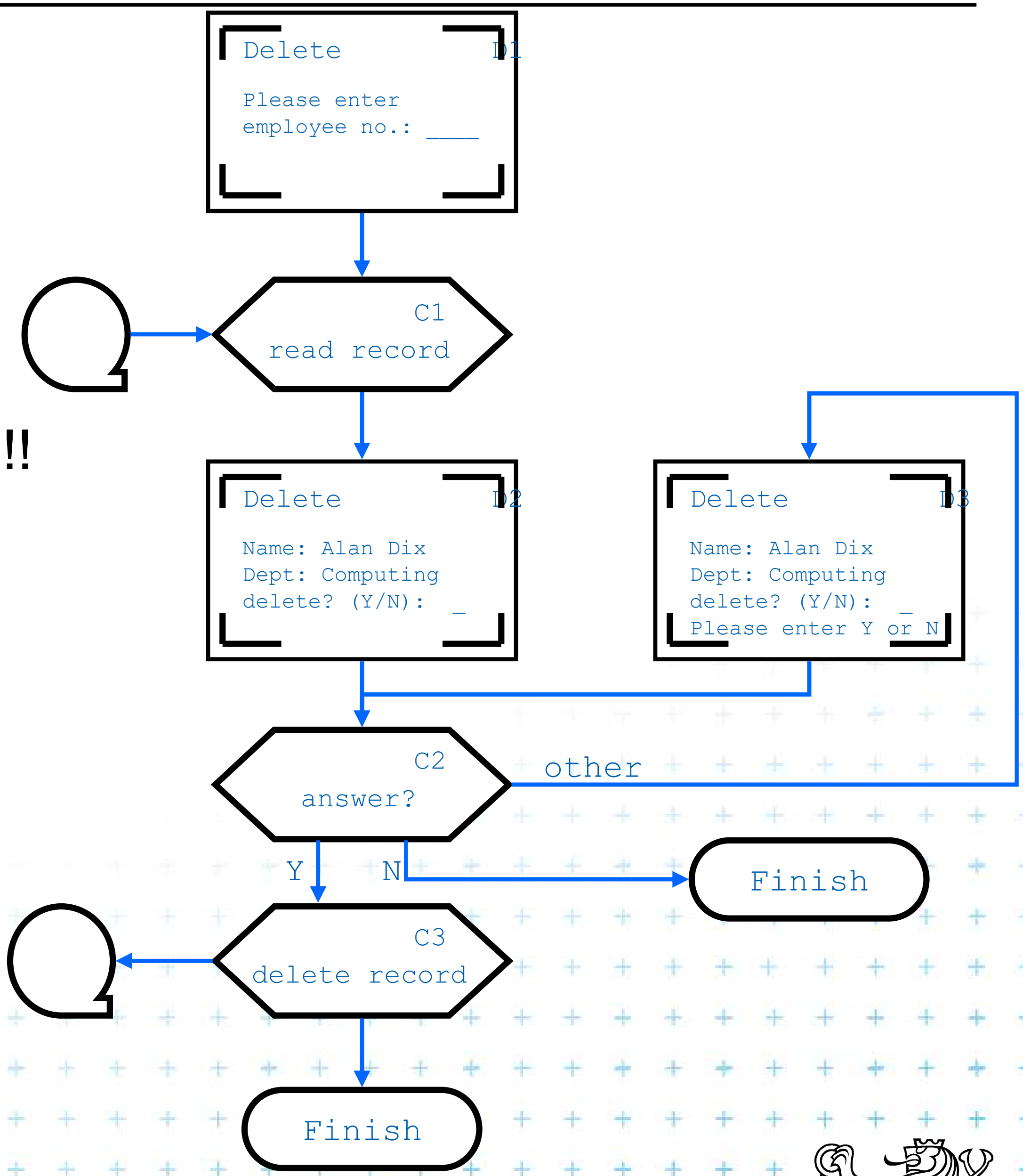
Weather forecast STN



Other models

Flowcharts

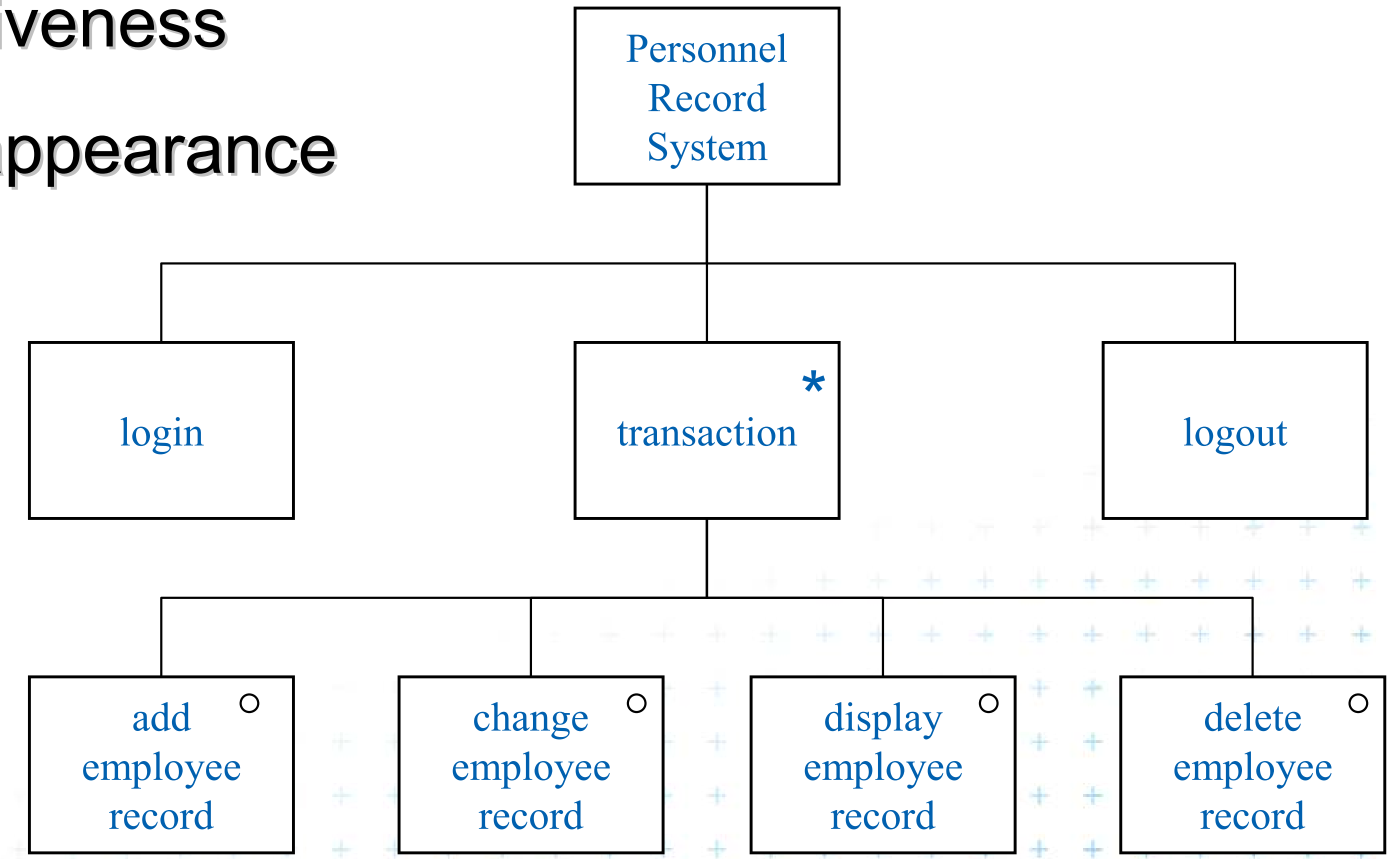
- Blocks
  - processes/events
  - stateless
- Dialog description
  - not the algorithm description!!!



JSD diagrams

For tree structures

- Poor expressiveness
- Good visual appearance



Dialog description - Summary

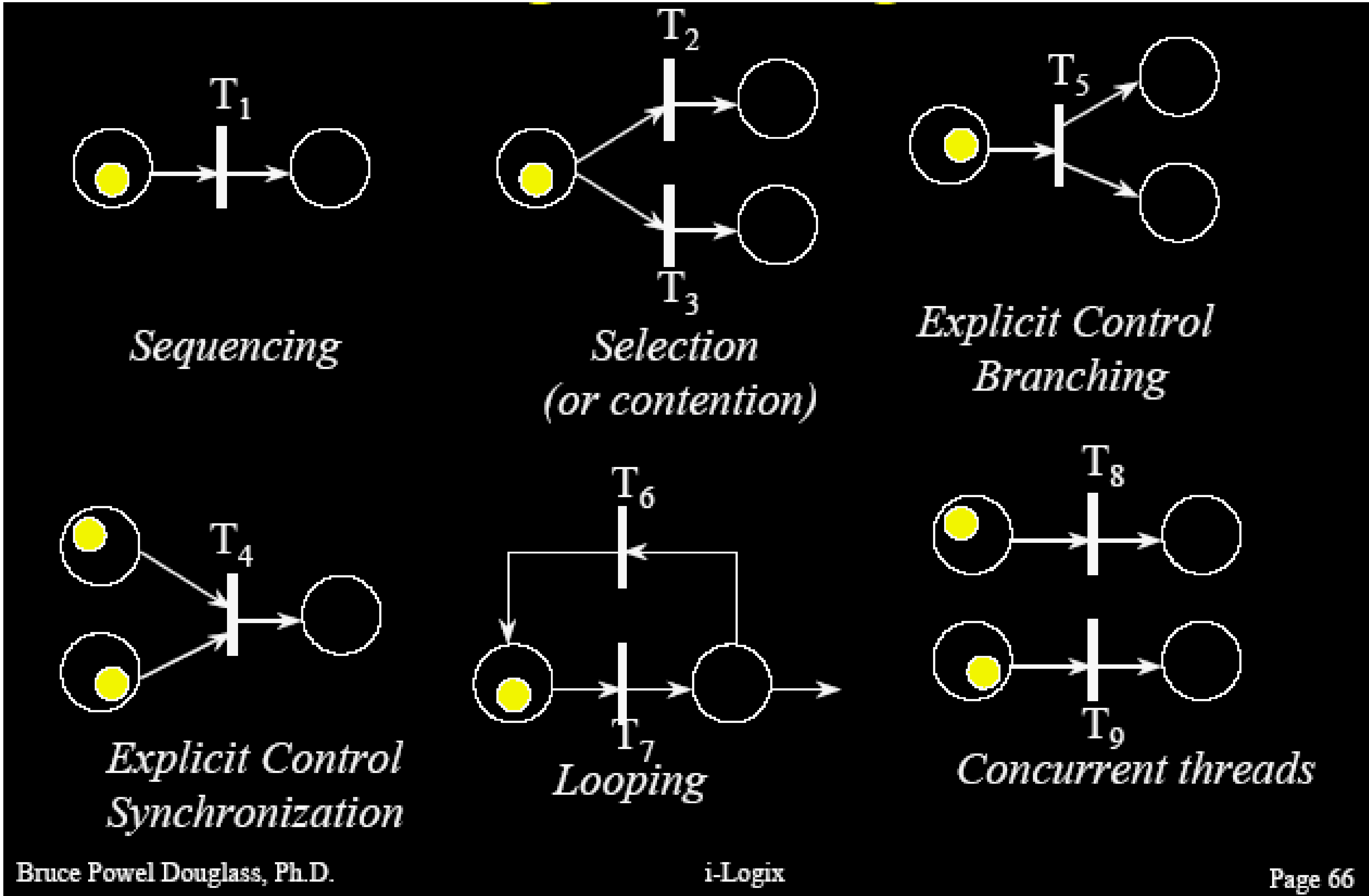
- Diagrammatic
  - STN, JSD, Flow charts
- Textual
  - grammars, production rules, event algebras
- Issues
  - Based on events and transitions between states
  - Powerful description and easy to "read" and interpret
  - model vs. description
  - Sequential vs. parallel

Dialog model: Petri nets (PN)

What are Petri nets?

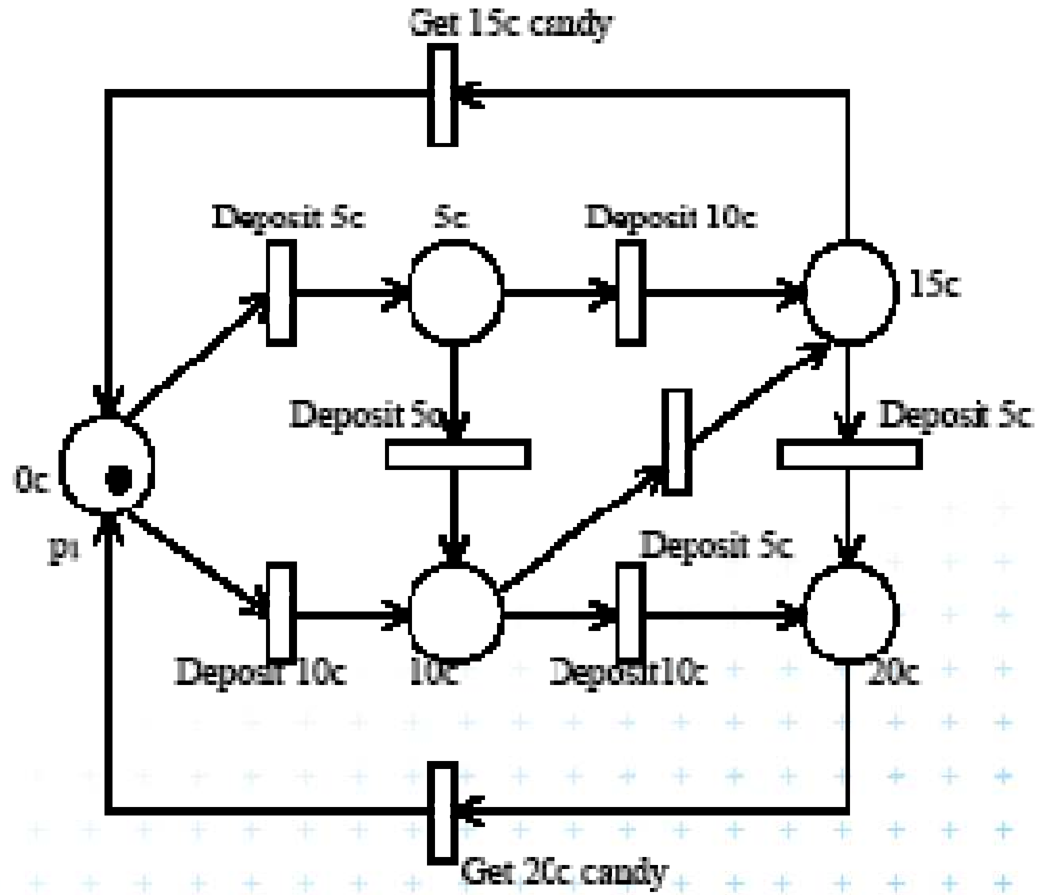
- Similar to FSA
- Transition between states is made by “token shift”
- Event can trigger the transition only in the case when tokens are on all inputs
- Result of a transition: tokens are removed from inputs and they are placed on outputs (synchronization)
- PN are applied mostly in HW applications (synchronization)

PN: Standard programming constructs

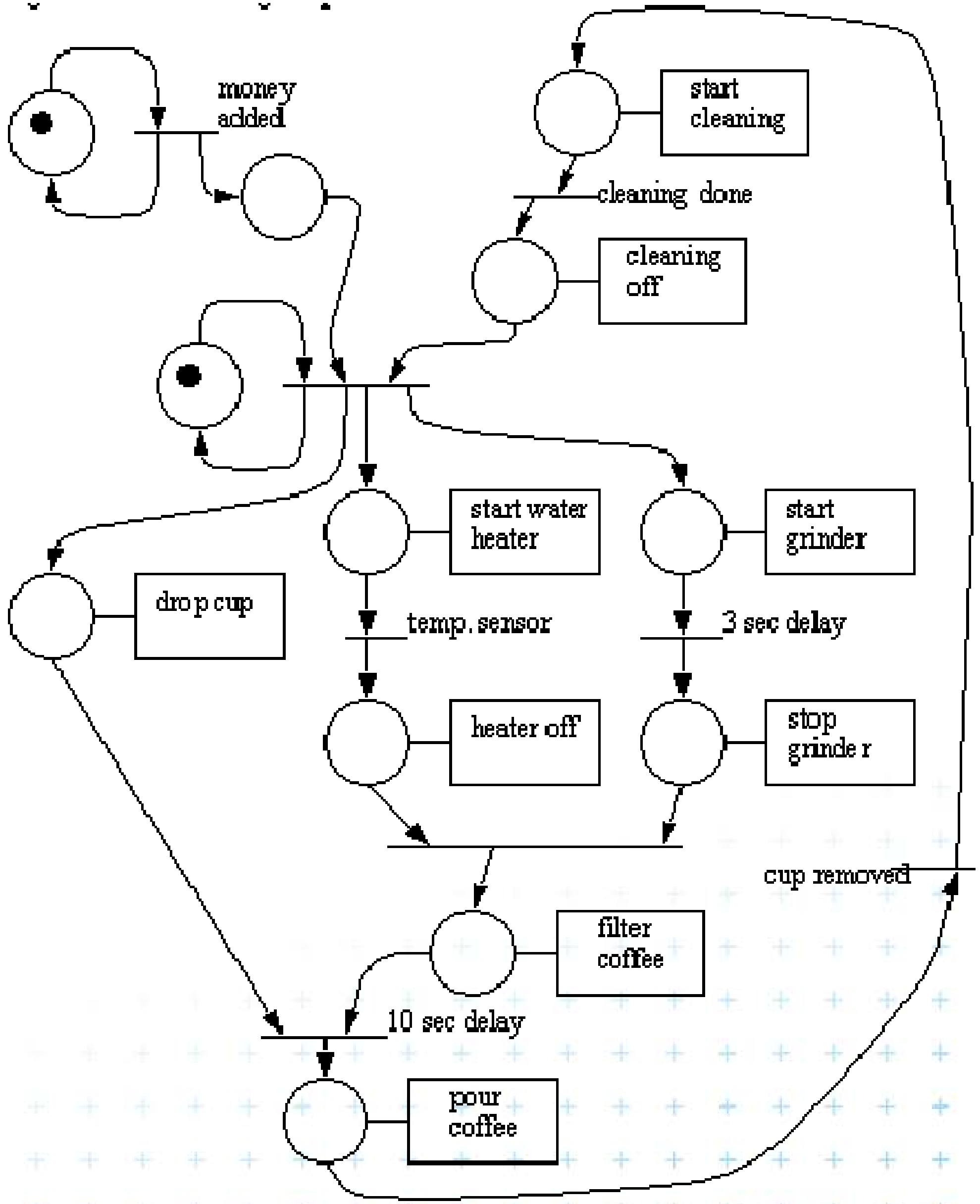


PN example: vending machine

- finite-state machine
- accepts either nickels or dimes
- sells 15c or 20c candy bars
- vending machine can hold up to 20c
- it does not return coins



PN example: coffee maker



STN and VISUAL PROGRAMMING

Slides from MIKE SCOTT

What is Visual Programming?

- What is a computer program?
  - Do you use many programs?
  - Have you ever written a program?
- What skills are necessary for programming?

Text Based Programming

```
private String[] getWordList()
{    Scanner s;
     String temp;
     JFileChooser chooser = new JFileChooser();
     TreeSet<String> words = new TreeSet<String>();

     //open the file chooser and get a file
     int retval = chooser.showOpenDialog(null);
     chooser.grabFocus();

     if (retval == JFileChooser.APPROVE_OPTION)
     {    File source = chooser.getSelectedFile();
          try
          {    s = new Scanner( source );
               while( s.hasNext() )
               {    temp = s.next();
                    if( temp != null && temp.length() == WORD_SIZE )
                    {    words.add( temp );
                    }
               }

               s.close();
          }
          catch(IOException e)
          {    System.out.println("An error occured while trying " +
                   "to read from the file: " + e);
          }
     }

     return words.toArray(new String[words.size()]);
}
```


Eww!!

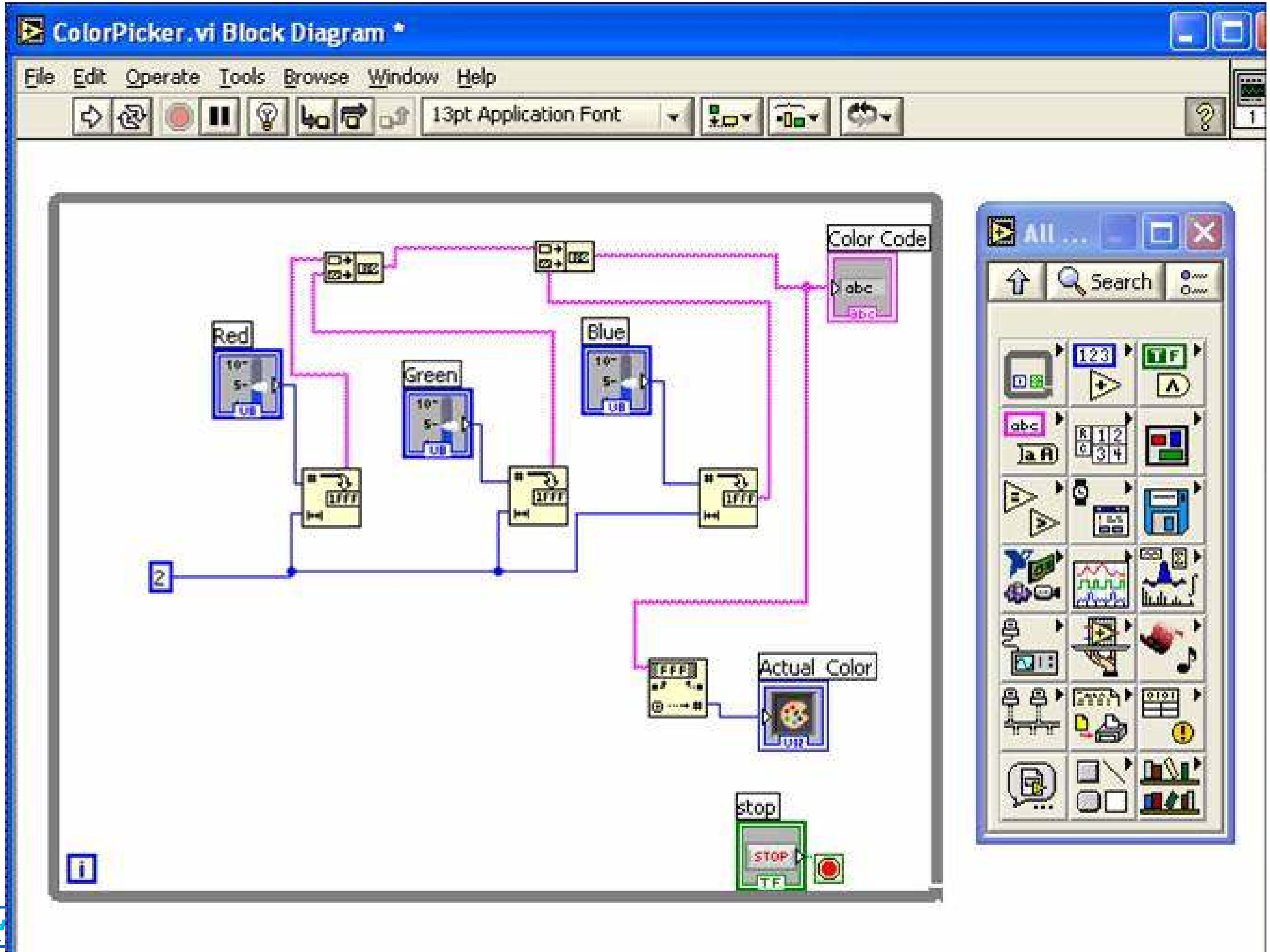
Visual Programming

- Use of a graphical interface to create the program instead of text
  - easier with simple programs
  - less likely to make mistakes
  - Visual Basic, Visual C#, Visual C++
    - visual tool for creating graphical user interface (GUI), but lots of text based programming still required
- IEEE has an annual conference on Visual Languages/Human Centric Computing (VL/HCC)

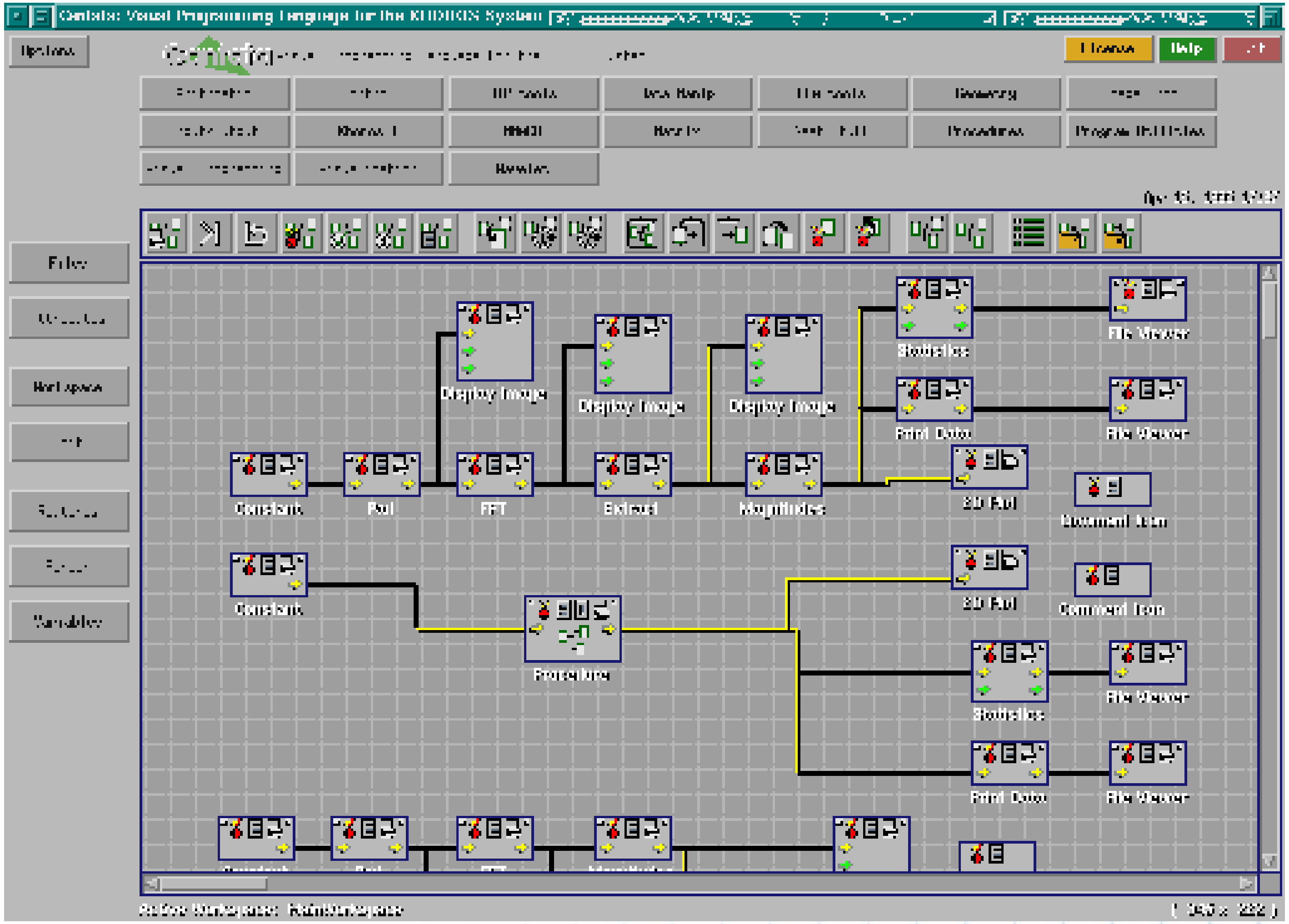
Visual Programming

- Lots of true Visual Languages
  - VIPR, Self, Pygmalion, Prograph and many, many more
  - not a lot of commercial successes
- Exceptions: LabVIEW + some more

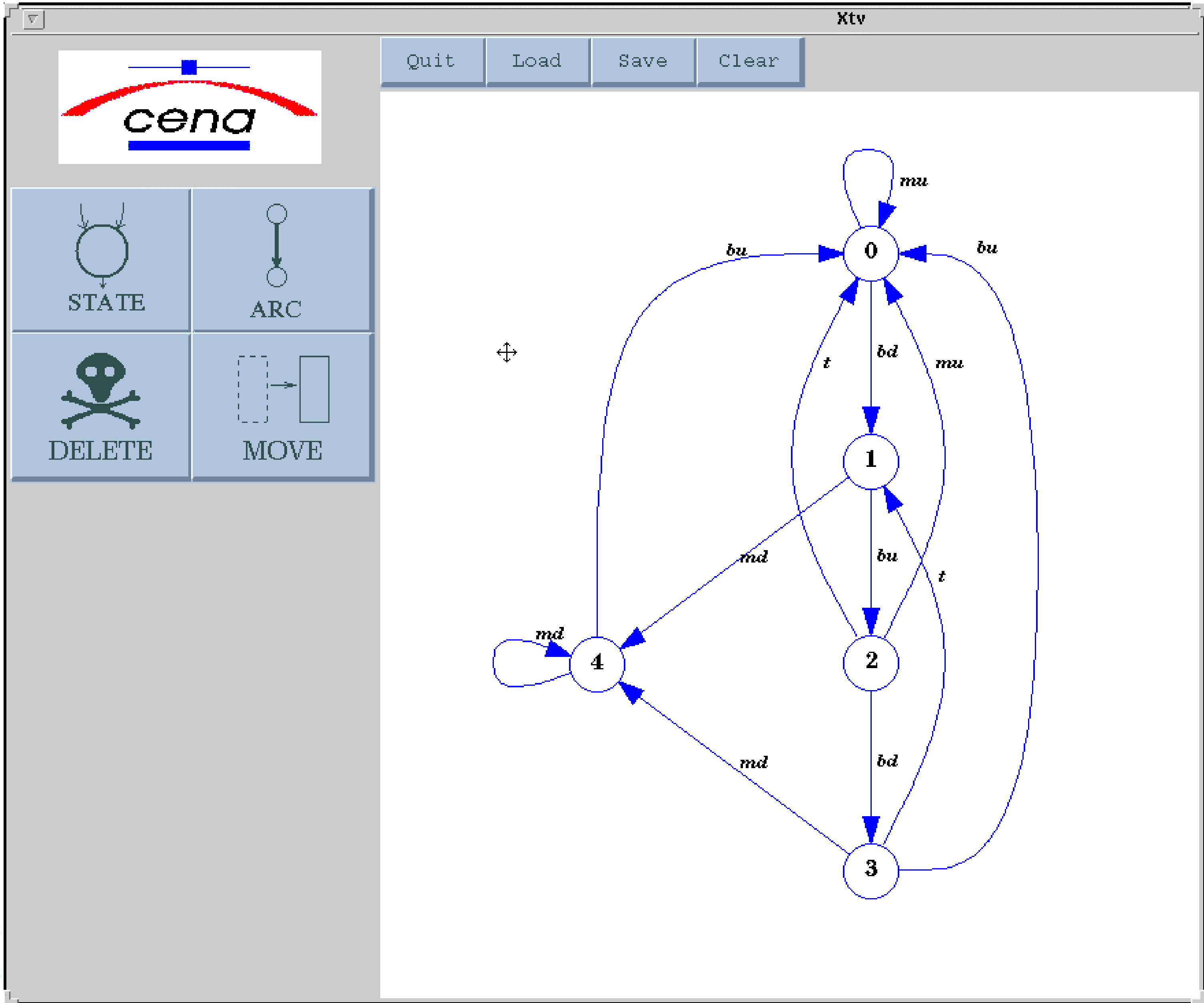
LabVIEW Program



KHOROS - Visual Programming

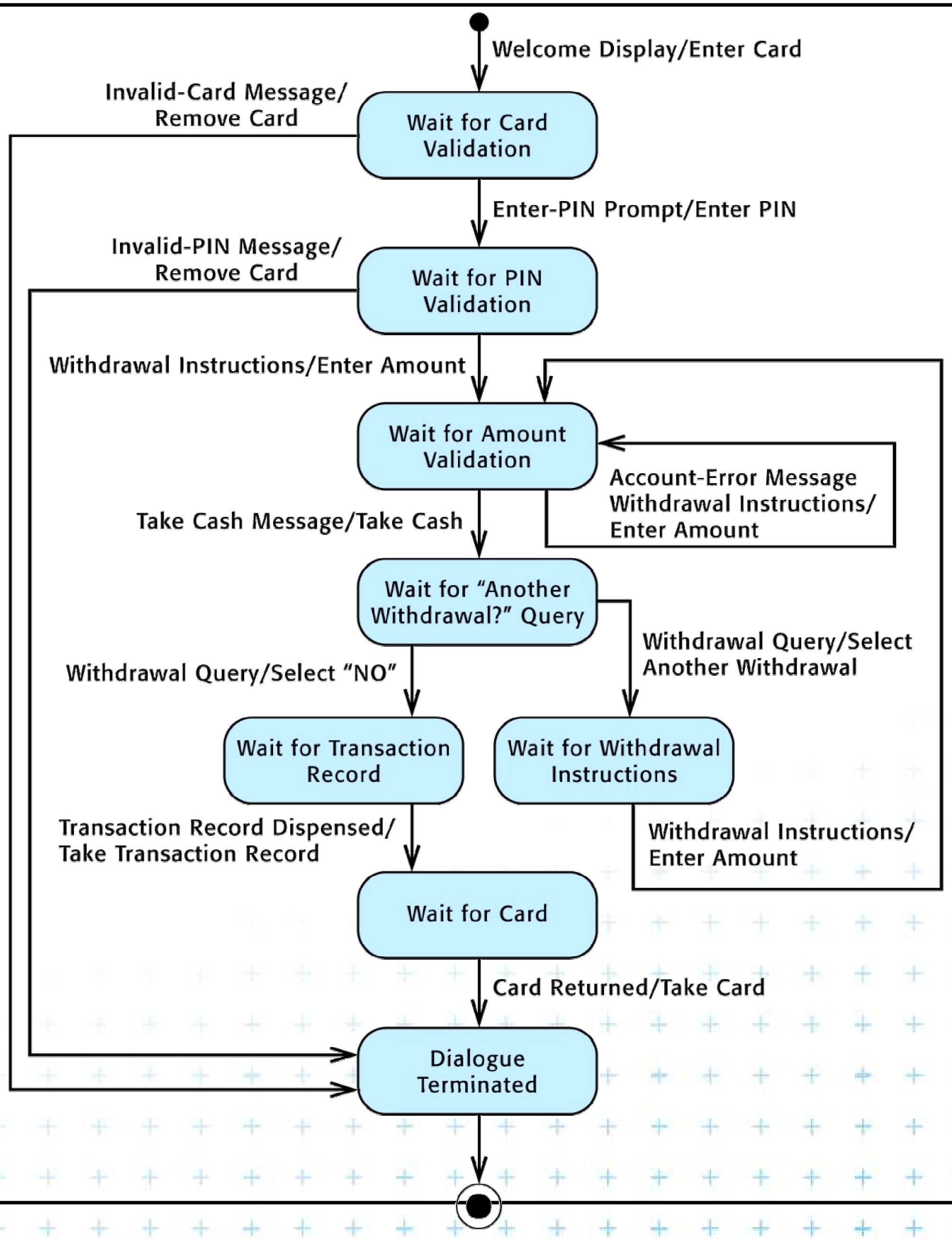


System for STN creation

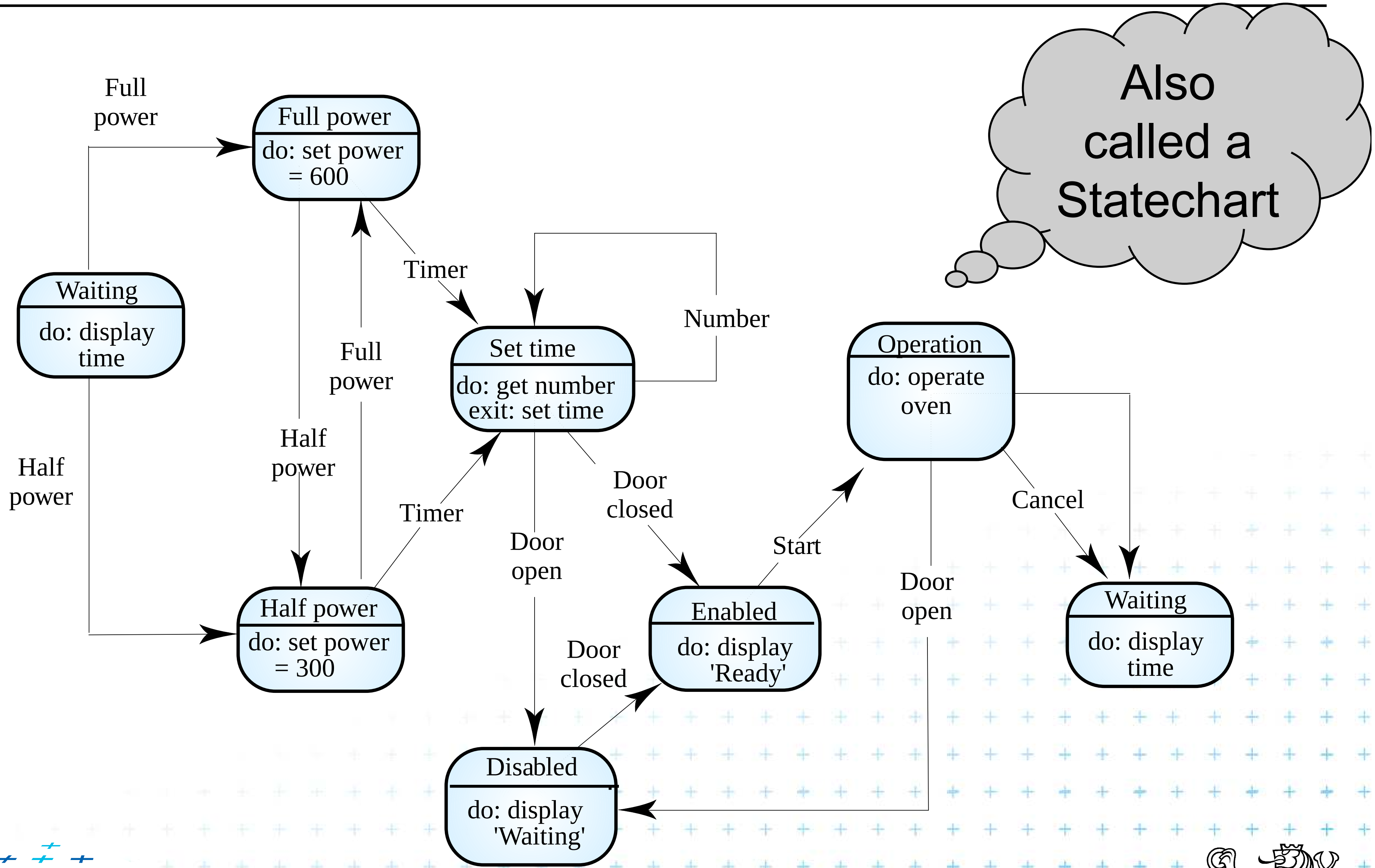


Homework – analyze the following examples

An Example of a State Transition Diagram



Microwave oven model



- Slides with pictures in this lecture were taken mostly from M. Rautenberg (TuE - The Netherlands)

Thank for your attention