Computer algebra in interface design research

Harold Thimbleby
h.thimbleby@ucl.ac.uk

Jeremy Gow
j.gow@ucl.ac.uk

UCL Interaction Centre (UCLIC)
Remax House, 31-32 Alfred Place
London WC1E 7DP, UK

ABSTRACT
Tools to design, analyse and evaluate user interfaces can be used in user interface design research and in interface modelling research. This demonstration shows two working systems: one in Mathematica, that is mathematically sophisticated, and one as a 'conventional' rapid application development environment, where the mathematics is hidden, and which could form the basis of a professional design tool — but also based rigorously on the same algebraic formalism.

Keywords
User interface design, computer algebra, matrix algebra, Mathematica, *MAUI*.

INTRODUCTION
Finite state machines (FSMs) machines can in principle represent any finite discrete process, including concurrent systems such as graphical user interfaces. FSMs support the definition and analysis of many standard user interface concepts, such as reachability and shortest paths, which can be naturally expressed in them. Various techniques, notably statecharts and process algebras, have been proposed, and successfully used, to manage the models.

FSMs can be partitioned into matrices, based on the transition matrix, such that each labelled transition has its own matrix. Operations on the state are now represented as matrix multiplications. This gives us an algebra of user actions and states.

The main advantages of this algebraic approach are that:

1) Properties of the user interface are now readily expressed as algebraic theorems, and their scope and validity is easily calculated using elementary matrix operations. The scope and value of doing this is a subject of our research.
2) A matrix representation lends itself to modelling and simulation. We have built a tool *MAUI* to do this [1].
3) Matrices can be compressed, and there are good grounds to think that this may suggest ways to improve user interfaces
4) Matrix algebra is a standard mathematical concept; no new notation or ideas need to be introduced. Matrix calculation and algebraic methods are well supported by numerous tools, including computer algebra tools, such as Mathematica [3].

DEMONSTRATION
The demonstrations show our research working in two ways: Mathematica [3] simulations (here, of a calculator) and our tool *MAUI* (here simulating a Sanyo CD player). The simulated devices are familiar and need little further explanation here; they show how the approach can support research (e.g., into modelling and formal issues in HCI), and are suggestive that the approach can scale up to many other sorts of interactive device: the simulations are accurate, and our approach has not taken any unnecessary 'short cuts' in its favour.

Example 1: Mathematica & a Casio calculator
Our first example is developed in Mathematica, a widely used computer algebra system. Mathematica represents an enormous resource in mathematics: e.g. it allows a researcher to write papers and mathematics together, with Mathematica evaluating and doing routine calculations. Mathematica can also be programmed and hence extended.

In our case, a small amount of programming supports our matrix approach. (In practice a Mathematica user would include a package, and this sets up Mathematica to work as described here.)

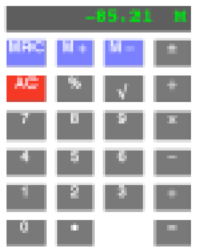

Figure 1. Interactive Mathematica device simulation

A fairly complete specification of a Casio HS-8V handheld calculator was written in Mathematica (it does not handle numerical errors as the Casio does, and it does not have an auto power-off). The style of specification is executable, from which Mathematica generates a fully working simulation (which can record user events for later analysis). A picture of such a simulation is shown in Figure 1. The Figure is *not* very exciting — it is pretty similar to the

Casio itself — but that is the point: the simulation is realistic and functionally accurate.

The specification of the calculator in Mathematica is essentially unrestricted, and the simulation can be tested on users, *etc*. The package then defines utility routines that convert such arbitrary specifications into matrix form (if possible — the conversion process will highlight modes and other design issues that may have been overlooked).

Example 2: MAUI and a Sanyo CD player

Whilst Mathematica is undeniably powerful and flexible, it is only suitable for research and exploratory development. For more practical use a different approach is necessary. In our second example, then, we show how the matrix algebra approach can be accessed via a fairly conventional GUI user interface, much like a rapid application development environment.

The *MAUI* system [1] was built to *automatically* support the kind of algebraic analysis that one might carry out in Mathematica, but in the context of a practical design/modelling tool. A model of a user interface can be built up in *MAUI* via a series of editing commands, then simulated and analysed (using matrix algebra techniques). *MAUI* makes a useful subset of the computer algebra techniques accessible without the need to explicitly program it (or Mathematica). Of course, we loose a vast amount of mathematical sophistication: *MAUI* restricts the model to being a (possibly non-deterministic) FSM.

MAUI was used to build a model of the play/pause/rest modes of a Sanyo portable CD player (see Figure 2), consisting of 29 states and 4 user actions. As well as checking the model against an algebraic specification of actions and states, the *MAUI* was used to generate suggestions for algebraic properties that are true or *nearly always true.* Designs can often be improved by making properties universally true — they make the user experience simpler and more consistent. This ability to generate `nearly always true' theorems is unique in a design tool that could in principle be used by professionals (particularly in safety critical domains).

EXAMPLE RESULTS

The two examples in this paper are very different, but both raised some interesting design issues, which are easily represented as algebraic theorems. To illustrate:

— A user cannot easily store the calculator's displayed number in memory unless the memory is already zero. It is non-trivial to zero memory when the display is non-zero.

— Most of the time the CD player's Mode button cycles through seven play modes. In order to help the user two identical states could be merged, making it true all of the time. *MAUI* identifies the almost true theorem.

Matrix algebra has the potential to be used to explore a much wider range of usability issues than we can illustrate here. Determining the scope and applicability of such methods is the subject of further research.

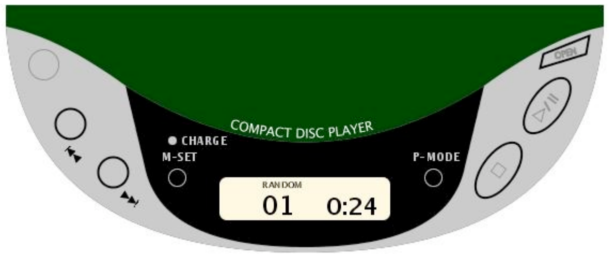

Figure 2. Interactive SVG simulation, generated from MAUI XML.

CONCLUSIONS

We have shown that matrix algebra is a rich source of research and development ideas in user interface design. Matrices are well known, efficient and easily implemented. They can be manipulated in standard programs (e.g., Mathematica) or in special purpose programs can be developed (e.g., *MAUI*). Both our approaches are platform independent, though Mathematica is an expensive tool. *MAUI* in contrast is written in Java and is free.

Mathematica is enormously flexible, and there are surprisingly few limitations on an algebraic approach in the hands of a skilled user. On the other hand, *MAUI* shows that very practical — but rigorous — development can be done is a standard GUI environment, and that this route is perfectly adequate for some stages of conventional interactive device design. Using XML, *MAUI* integrates well with other open source tools (including Mathematica).

It would be nice to see this research embedded in some ways into conventional design tools, though obviously the requirements of design are different from the requirements of research. Current design tools (Flash being an example) emphasise generality rather than precision, so retrospectively introducing formality would be pointless. In safety critical areas, however, the benefits are significant, and we can look forward to

ACKNOWLEDGMENTS

Harold Thimbleby is a Royal Society Wolfson Research Merit Award Holder. Jeremy Gow is funded on the award.

REFERENCES

1. Gow, J. & Thimbleby, H. (2003) *MAUI: Matrix Analysis of User Interfaces*, project web page: http://www.uclic.ucl.ac.uk/usr/jgow/maui
2. Thimbleby, H., Cairns, P. & Jones, M. (2001) "Usability analysis with Markov Models," *ACM Transactions on Computer Human Interaction*, **8**(2):99-132.
3. Wolfram, S. (2003) *The Mathematica Book*, Wolfram Media.