Empirical Modelling as a conceptual framework for computing

The overall scope of the module will be sketched and motivated with reference to the distinction between a conventional program and an EM model, as epitomised by the original BBC 'JUGS' program and the EM jugs model `jugsBeynon1988` (see slides 1 and 2).

The classical program

```
                -----------------------
INPUT  ->  |       PROGRAM       | -> OUTPUT
                -----------------------

                 "Program as abstraction"
```

- Identification of a program with a functional relationship
- Computable function, as associated with a Turing machine
- Logical characterisation via recursive function theory

Computing activity seen to be founded on the mathematical theory of computation. Well-suited to addressing the earliest and most conceptually clean forms of computer use.

Computing practice

Several different purported views of programs are represented in practice:

- Program as **artefact**:
  - Viewed as an artefact, a program has physical characteristics and state that can be observed, manipulated and informally interpreted. It typically also has an "intention" - it is associated with some external referent from which it derives its meaning.
  - Example: consider the EM Jugs model, a geographical information system, a library database, a spreadsheet, a *Julia set* visualization.

- Program as **animation**:
  - Viewed as an animation, a program has a behaviour in which all the intermediate states are displayed and meaningful.
  - Example: consider the JUGS animation, dynamic systems simulation, as in airflow or weather system, a vehicle cruise control or sailboat simulation,

- Program as **agent**:
  - Viewed as an agent, a program is being interpreted with reference to its contribution to behaviour at a very high level of abstraction, as if (e.g.) it exhibited autonomous behaviour and could be pro-active. This requires some presumption of rich context for operation and interpretation in which it apparently makes sense to regard a program as "intelligent", or "trustworthy".
  - Example: consider the mechanisms behind a button interface, an 'intelligent' software agent, an automated player/character in a computer game, wrappers / drivers for sensors and actuators.

A tension between theory and practice

Each of the above views of a 'program' drawn from *practice* is legitimate in certain contexts. None of these views of a program is easily reconciled with the notion of program-as-abstraction that epitomises computer science *theory.*

What to do about this?

Where practical programming is concerned, try to be faithful to the spirit of program-as-abstraction in two ways, by:

- developing declarative programming paradigms
- providing mathematical semantics for procedural programs

FP not addressing the artefact aspect - how is input-output mediated? Must be a change of state for user to appreciate that computation has occurred, but declarative paradigm does not give much support for state. Similarly problem in respect of animation: dataflow, Lucid - stream data types, artificiality and inflexibility of such technical devices.

FP and logic programming incorporating extra-logical mechanisms: lazy evaluation, the cut ...

The notion of intentionality

It is widely (if tacitly) recognised that computation is in one way or another a symbolic or representational or information-based or semantical - that is, as philosophers would say, an *intentional* - phenomenon. Somehow or other, though in ways that we do not yet understand, the states of a computer can model or simulate or represent or stand for or carry information about or signify other states in the world (or at least can be taken by some people to do so).
- *Brian Cantwell-Smith, The Foundations of Computing*.

For a declarative programming paradigm that respects the spirit of logic the relationship between form and content must resemble that in a formal language - symbols are ascribed interpretations that are context-independent prior to execution.

Mode of dealing with program content / intention then very limited: as Brian Cantwell-Smith observes - supplying a mathematical semantics for programs does not assist in making the link between form and content (cf. Slide 3).

Two ways of dealing with the intentional aspects

Separate form from content: cf. the formalist view as expressed by Bornat in *"Is 'Computer Science' science?"*:
"Programming is ... a branch of formal logical proof, which is the only completely meaningless activity in human history."
"... programming is all made up, quite unreal, so we can't be physically scientific about programs or even their executions."

... BUT intentional aspects are necessarily bound up with the process of constructing and debugging programs, hence there is interest in ways of relating program structure and internal program state to that of its referent in program design and execution. Hence, in programming practice ...

Aim to integrate form with content:

Consider e.g.

- Data structures and algorithms - exploit physical metaphors to make the relationship between what is being represented and how this is represented in the machine easier to understand.
- Object and process abstractions - deliberately organising data and procedures in ways that reflect the external organisations they are designed to reference.
- Design artefacts - creating class diagrams, message sequence charts, state charts etc to specify interactions between software components.
- Software management strategies, such as XP, designed to link development with domain analysis and identification of content

Integrating form and content is problematic precisely because the fundamental concept of program is ill-suited to treating programs as artefacts, as animations and as agents.

More about the program context

The practical interpretations of programs are legitimised (when they are!) only with reference to a broader context.

What other elements are associated with a program?

```
-----------------------------------------------
!            ----------------------            !
! INPUT  ->  |       PROGRAM      | -> OUTPUT  !        \
!            ----------------------            !  -----  (>
!                                              !        /
!            "Program as abstraction"          !
-----------------------------------------------
```

- There is an implicit human interpreter - indeed in the broad study of programming there are several candidate interpreters: users, developers, analysts etc
- There are some 'agents' interfacing with the program - for instance, there may be an agent like the user of a calculator, or a spreadsheet, or a database, or there may be a internal processes to generate the input and/or respond to the output, or sensors and actuators attached to input and output.
- To justify the term 'program', it must be possible to identify the context as relatively stable, reconstructable and objective: cf. the way in which an algorithm relates to a class of possible input-output pairs (as in: *what is the product of two given integers? does a given graph have a three-colouring?* etc).

Intention and interpretation in the program context

In practical use, programs have an intentional aspect - they are 'about' something. When programs are interpreted as artefacts, animations and agents, the way in which they have to be interpreted is critically important, and far outside the scope of conventional "program semantics". Relevant issues concern:

- (for any program) the status of behaviour as a *construal*: a human construct in the mind associated with a configuration in the world that embraces the stability of the interactive context, the predictability of machine response, and the protocol for interaction and interpretation.
- (for program-as-artefact) the way in which relationships between variables are perceived to be respected in open interaction - cf. how the functional operation of a spreadsheet program is to be interpreted - and uninterpreted - in spreadsheet creation and use.
- (for program-as-animation) how intermediate program states are made visible and interpretable in animation.
- (for program-as-agent) how agents both contribute to state and are deemed responsible for state change, how timely and reliable etc is the response of the program.

Over and above the intended use and interpretation of a program, there is also the possibility for opportunistic use (e.g. if I invoke an application simply in order to hide confidential text content in a terminal window).

Programs-as-artefacts in an archetypal setting

Of particular relevance for Empirical Modelling are programs-as-artefacts with a specific intention. For instance, a computer-based artefact might refer to a financial situation, a geographical feature, a piece of music or an engineering design prototype. Consider say, a computer-based artefact that serves as a map of a geographical feature.

Primarily concerned with a static interpretation - cf. spreadsheet state, database view, *as of now*, but at the same time vitally concerned with the correspondence between the map and its referent, and with supporting typical kinds of interaction with map and referent that don't actually change the referent.

Over and above this, the mode of construction is significant - consider the relevance of change and the importance of being able to synchronise change to referent with change to the artefact.

In summary, must support the possibility of many different kinds of agency e.g.

- interaction associated with travelling around and getting different viewpoints,
- effects of erosion, subsidence, changes to river course, catastrophe (earthquake / volcano / tsunami), building development,
- impact of political change, further exploration, reclassification etc,
- significance of a technological advance in map technology,
- relevance of greater efficiency in realisation of map,
- need to accomodate an enhanced conception of the role of the map etc ...

... but also process of computer-based map construction, possible subversion of the representation through backdoor access to code etc

---

Programming guided by intention - hence modelling

Programming as modelling:

- object-oriented modelling and programming
- relational databases
- software development with XP, process-oriented [Warboys etc], Harel's play-in scenarios to code
- agent-based software engineering

Relevant issues and critiques:

- Kaindl, difficulties in the transition from OO analysis to design
- the constructionist agenda
- Brooks, No Silver Bullet
- Harel, From Play-in scenarios to code
- Ridley and Papadimitriou on the foundations of databases
- Loomes, spurious reification in the software development process
- Lind's reflections in *Issues in agent-oriented software engineering*

Empirical Modelling as a conceptual framework for computing

Empirical Modelling involves creating (computer-based) artefacts called **construals** rather than programs in the conventional sense. (The framework for model-making as construal is set out on slide 4.)

Subject to technological constraints etc, EM construals can mimic programs; they can also play a role in developing programs.

EM construals are better suited than computer programs for the roles of artefacts, animations and agents, especially having regard to the significance of change, reinterpretation and development in their realisation.

Term 'construal' and the initial orientation are borrowed from David Gooding, whose research is concerned with "aspects of scientific work largely neglected by modern, especially analytical, philosophy. These are the agency of observers and the way their observation of nature is mediated by their interactions with each other, with their instrumentation and with the natural world." The most appropriate philosophical stance is to be found in William James's Radical Empiricism.

Three examples by way of posters to illustrate EM artefacts, animation and agency: a model of a planimeter, of Schubert's song Erlkoenig, and of the historic railway operation in the vicinity of the Clayton Tunnel. (These can be accessed online in the "posters" subdirectory of `kaleidoscopeBeynon2005` in the EM archive.)

How do construals and programs relate?

No simple answer - the relationship is very complex and subtle.

The relationship is similar to that between the personal, subjective, particular, provisional and tacit and the public, objective, general, assured and explicit.

What is amenable to formal symbolic representation and logical analysis belongs to the objective domain: construals of which we have personal experience of interaction are the means to embody our subjective understanding.

The analogy that William James uses for this ... (see slide 5)!

Better analogy? Believe that EM provides this - the study of modelling with definitive scripts etc opens the way to a comprehensive new metaphor for computing ...

EM is particularly concerned with the importance of grounding logic in experience (cf. slide 6) ... the relationship between a program and a construal is just an indication of how deep and difficult an issue this is in general.

About CS405

The module will address the ways in which EM supports the interactive development of construals that serve as artefacts supporting many different kinds of agency and modes of animation. EM principles and tools will be introduced with reference to:

- Creating EM artefacts through **modelling with definitive scripts**
- Developing a framework for realising agency and animation in which agent action is modelled by concurrent redefinition.

A selection of applications of EM relating to areas such as software development, educational technology, concurrent design and human computing will then be reviewed.

The assessment for the module will involve the submission of a paper and a model on a theme of your own choice to the third Warwick Empirical Modelling Bulletin: WEB-EM-3. For more background, see:
First Warwick Empirical Modelling Bulletin: WEB-EM-1
http://www.dcs.warwick.ac.uk/~wmb/web-em-1/

Empirical Modelling horizons

Questions into which EM potentially offers insight:

- how to tackle problems of requirements?
- how to support incremental software development?
- how to give computer support for constructionist learning?
- how to develop a conceptual framework in which to integrate manual and automatic activities?
- what are the foundations of agent-based systems and databases?
- to what extent can virtual experiments contribute to science?
- how to support a phenomenological perspective on computing activity?
- how to make sense of the relationship between logic and experience?

References

EA Ashcroft and WW Wadge, Lucid, a Nonprocedural Language with Iteration, Comm. ACM, 20 (7), pp. 519-526, July 1977.
Beck, K. Extreme Programming explained: embracing change.
Bornat, R. Is 'Computer Science' science?

Brooks, F.P. No Silver Bullet.
Cantwell-Smith, B., The foundations of computing.
Cantwell-Smith, B., Two lessons of logic.
Gooding, D., Experiment and the making of meaning.
Harel, D. From play-in scenarios to code.
James, W. Essays in Radical Empiricism.
Kaindl, H., Difficulties in the transition from OO analysis to design.
Lind, Issues in agent-oriented software engineering.
Loomes. M. Fact and Artifact: Reification and Drift in the History and Growth of Interactive Software Systems.
Ridley, M.J. 'Database Systems or Database Theory - or "Why Don't You Teach Oracle"'.
Papadimitriou, C. Database Metatheory: asking the big queries.
B.Warboys, P. Kawalek, I. Robertson and M. Greenwood. Business Information Systems: a process approach. McGraw-Hill, 1999.

An extract from the 'JUGS' program, as programmed for the BBC microcomputer by Robin Bartlett on August 26th 1982

procedures

subroutines

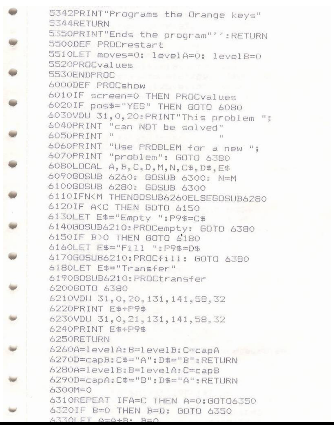
assignment

transfer of control

iteration

alternation

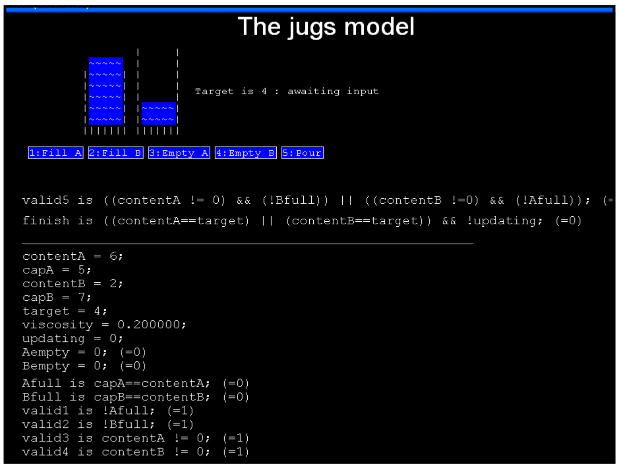
input-output statements



```
5342PRINT"Programs the Orange keys"
5344RETURN
5350PRINT"Ends the program"'':RETURN
5500DEF PROCrestart
5510LET moves=0: levelA=0: levelB=0
5520PROCvalues
5530ENDPROC
6000DEF PROCshow
6010IF screen=0 THEN PROCvalues
6020IF pos$="YES" THEN GOTO 6080
6030VDU 31,0,20:PRINT"This problem ";
6040PRINT "can NOT be solved"
6050PRINT "                    "
6060PRINT "Use PROBLEM for a new ";
6070PRINT "problem": GOTO 6380
6080LOCAL A,B,C,D,M,N,C$,D$,E$
6090GOSUB 6260: GOSUB 6300: N=M
6100GOSUB 6280: GOSUB 6300
6110IFN<M THENGOSUB6260ELSEGOSUB6280
6120IF A<C THEN GOTO 6150
6130LET E$="Empty ":P9$=C$
6140GOSUB6210:PROCempty: GOTO 6380
6150IF B>0 THEN GOTO 6180
6160LET E$="Fill ":P9$=D$
6170GOSUB6210:PROCfill: GOTO 6380
6180LET E$="Transfer"
6190GOSUB6210:PROCtransfer
6200GOTO 6380
6210VDU 31,0,20,131,141,58,32
6220PRINT E$+P9$
6230VDU 31,0,21,131,141,58,32
6240PRINT E$+P9$
6250RETURN
6260A=levelA:B=levelB:C=capA
6270D=capB:C$="A":D$="B":RETURN
6280A=levelB:B=levelA:C=capB
6290D=capA:C$="B":D$="A":RETURN
6300M=0
6310REPEAT IFA=C THEN A=0:GOTO6350
6320IF B=0 THEN B=D: GOTO 6350
6330LET A=A+B: B=0
```

Slide 1

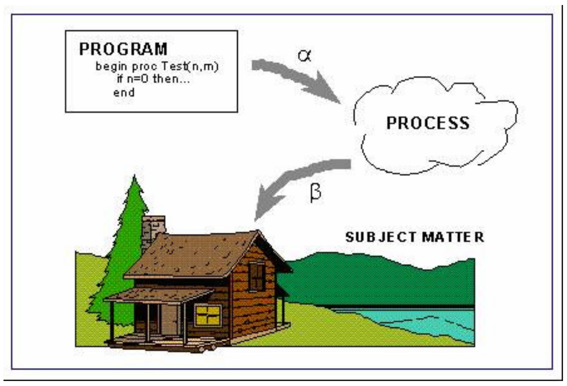
The jugs model



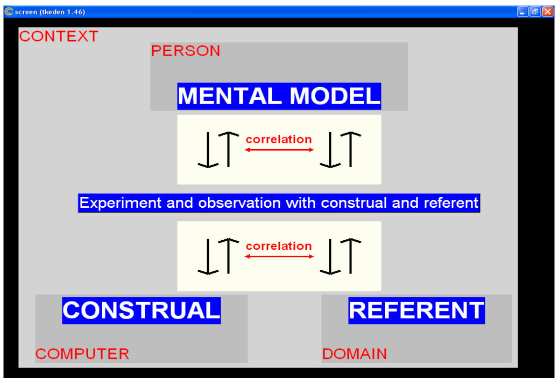
```
Target is 4 : awaiting input

1:Fill A  2:Fill B  3:Empty A  4:Empty B  5:Pour

valid5 is ((contentA != 0) && (!Bfull)) || ((contentB !=0) && (!Afull)); (=
finish is ((contentA==target) || (contentB==target)) && !updating; (=0)
________________________________________________________
contentA = 6;
capA = 5;
contentB = 2;
capB = 7;
target = 4;
viscosity = 0.200000;
updating = 0;
Aempty = 0; (=0)
Bempty = 0; (=0)
Afull is capA==contentA; (=0)
Bfull is capB==contentB; (=0)
valid1 is !Afull; (=1)
valid2 is !Bfull; (=1)
valid3 is contentA != 0; (=1)
valid4 is contentB != 0; (=1)
```

Slide 2

Cantwell-Smith - process semantics …



Slide 3



Model making as construing

Slide 4

On the face of it, if you should liken the universe of absolute idealism to an aquarium, a crystal globe in which goldfish are swimming, you would have to compare the empiricist universe to something more like one of those dried human heads with which the Dyaks of Borneo deck their lodges.

The skull forms a solid nucleus; but innumerable feathers, leaves, strings, beads, and loose appendages of every description float and dangle from it, and, save that they terminate in it, seem to have nothing to do with one other. Even so my experiences and yours float and dangle, terminating, it is true, in a nucleus of common perception, but for the most part out of sight and irrelevant and unimaginable to one another.

Essays in Radical Empiricism, William James (1912)

Slide 5

The common cormorant or shag
Lays eggs inside a paper bag.
The reason you will see, no doubt,
Is to keep the lightning out.

But what these unobservant birds
Have missed is this: that herds
Of wandering bears may come, with buns
And steal the bags to keep the crumbs.

Christopher Isherwood

Slide 6