Programming from an Empirical Modelling perspective

Programming / modelling in EDEN

The three primary concepts in EDEN are:

- definition
- function
- action

Informally

definition ~ spreadsheet definition
function ~ operator on values
action ~ triggered procedure

Definitions in eden

A formula variable v can be defined via

v is f(a,b,c);

EDEN maintains the values of definitive variables automatically and records all the dependency information in a definitive script.

Yellow text indicates eden keywords

Functions in eden

Functions can be defined via

```
func F
/* function to compute result = F(a,b,...,c) */
{
        para a, b, ..., c         /* pars for the function */
        auto result, x, y, …, z        /* local variables */
        <sequence of assignments and constructs>
        return result
}
```

Actions in eden

Actions can be defined via

```
proc P : r, s, …, t
/* proc triggered by variables r, s, …, t */
{
        auto x, y, …, z     /* local variables */
        <sequence of assignments and definitions>
}
```

Action P is triggered whenever one of its triggering variables r, s, … , t is updated / touched

Basic concepts of EDEN 1

Definitions are used to develop a definitive script to describe the current state: change of state is by adding a definition or redefining.

Functions are introduced to extend the range of operators used in definitions.

Actions are introduced to automate patterns of redefinition where this is appropriate.

The JUGS program ….

EM in the first instance models *state* …

… many varieties of state in programming

States relevant to programming ...

- state within the executing program
- external state: what is visible?
- state in respect of interaction
- state in program development
- state significant in the external world

Diverse representations are required:

- *state within the executing program*
  - Program variables, machine locations
- *external state: what is visible?*
  - Graphics / display techniques
- *state in respect of interaction*
  - Statechart, message sequence diagram

Diverse representations required …

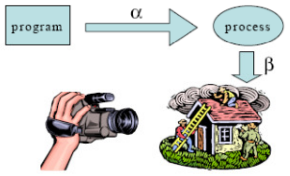
- *state in program development*

  UML diagrams, prototypes
- *state significant in the external world*

  apprehended by the human interpreter

cf. Brian Cantwell-Smith on semantics …

Semantic Relations (I)



The semantics of a traditional program

Semantics of a traditional program …

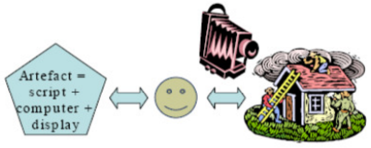
- Mathematical semantics is concerned with how the program manipulates variables
- “Real” semantics is concerned with how this activity connects with things in the world
- Traditional semantics associates abstract behaviours (“sequences of state-transitions”) with external concrete behaviours

⇒ Meaning is attached to *processes* not *states*

Reflections on traditional semantics

- Getting the program right vs getting the right program
- Cantwell-Smith: “the semantics of the semantics of the program”
- Problematic nature of matching processes rather than interactive states in experience

Semantic Relations (II)



The semantics of a definitive program

Semantics of a ‘definitive’ program …

- Counterpart of the mathematical semantics is how the script affords interactive experience
- “Real” semantics is concerned with how this interactive experience matches experience in the world (cf. the digit/cabinet construals)
- Empirically devising a suitable mechanism **and** associating the appropriate meaning

⇒ Meaning is attached to *state-as-experienced*

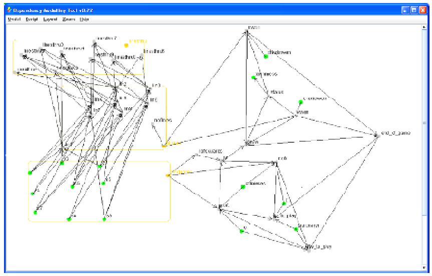
Reflections on “definitive” semantics

- Getting the mechanism right and making the right connection in experience two ways of interpreting one and same experience
- Blending of “the (real) semantics and/of the (mechanical) semantics of the script”
- Experimental method applies to checking: correlating interactive states in experience

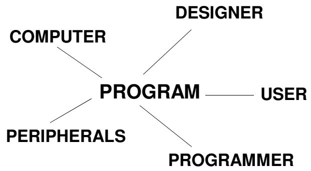
States within oxoJoy1994

Definitive scripts express …

- internal state – contents of squares
- visible state – appearance of the board
- interaction state: whose turn is it?
- state of development
- state of mind of the player: which square?



Dependencies in the oxoJoy1994 model



DESIGNER

COMPUTER

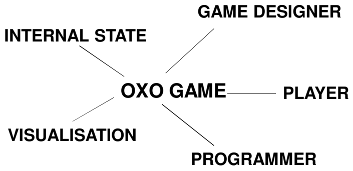
PROGRAM — USER

PERIPHERALS

PROGRAMMER

Diverse relations / representations in a traditional program

… compare this with the OXO laboratory



GAME DESIGNER

INTERNAL STATE

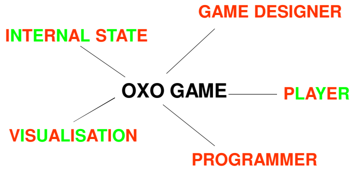
OXO GAME — PLAYER

VISUALISATION

PROGRAMMER

… all relations mediated by definitions

… Behaviour as programmed state change



GAME DESIGNER

INTERNAL STATE

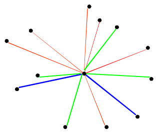
OXO GAME — PLAYER

VISUALISATION

PROGRAMMER

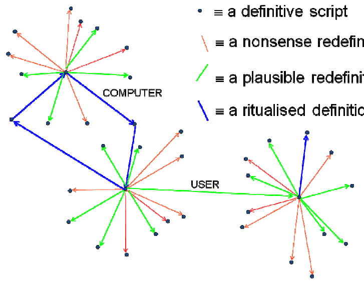
Static and dynamic elements of state

Definitive scripts as “fuzzy blobs”



- ≡ a definitive script
- ╲ ≡ a nonsense redefinition
- ╱ ≡ a plausible redefinition
- ╲ ≡ a ritualised definition

Plausible : *could* open the desk drawer
– note continuous spectrum of redefinitions
Ritualised : door *automatically* closes after being opened
Nonsense : opening the drawer makes the room smaller



- ≡ a definitive script
- ╲ ≡ a nonsense redefinition
- ╱ ≡ a plausible redefinition
- ╲ ≡ a ritualised definition

3 ingredients in construal development:

- engineering the states within which the agency of the user and the computer operate;
- crafting the behaviours which these agents then play out;
- projecting meanings on to the agent actions

"Vertical", "horizontal" and "orthogonal" dimensions of state ...

... playing a key role in programming

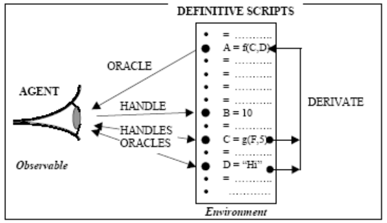


Figure 2-18: Definitive script as observer’s model of state (‘one-agent’ modelling)

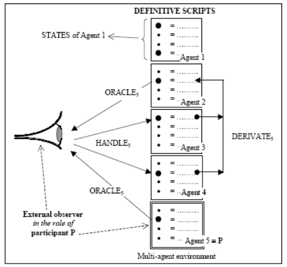


Figure 2-19: Definitive script as observer’s model of state (‘multi-agent’ modelling)

Modelling with definitive scripts:

… a holistic view of state that integrates and conflates all the different perspectives

in contrast to

Programming-in-the-wild:

… an eclectic model of state in which many different strategies for representation and interpretation are jumbled up together

Classical programming …1

Behaviour is derived from a pre-specified conception of function and purpose …

… based on interactions whose outcomes are reliable and for which the mode of interpretation is determined in advance

…motivates declarative approaches

Classical programming …2

… motivates declarative approaches:

`output=F(input)`

… problematic to deal with a dynamic input, as in playing a game

… hence add "lazy evaluation" to model as

`stream_of_output=F(stream_of_input)`

Significance of interpretation …

Miranda *can* be viewed as a definitive notation over an underlying algebra of functions and constructors

BUT this interpretation emphasises

program design as a state-based activity

NOT

declarative techniques for *program specification*

Illustrative example

… a version of 3D OXO written in the functional programming language Miranda

… to be compared with oxoJoy1994 which was in some respects 'derived' from it

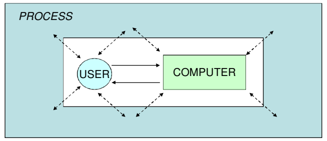
Two experimental systems!

A definitive Miranda ("admira"): definitive notation with general functional programs and types as operators & data structures

The Kent Recursive Calculator (KRC): developing functional programs by framing definitive scripts

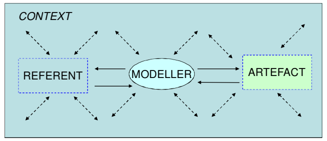
Programming from two perspectives

- a program is conceived with reference to how its behaviour participates in a wider process with functional objectives: states emerge as the side-effects of behaviours
- a computer artefact is developed so as to reflect the agency within an environment: the artefact and environment evolve until (possibly) program-like processes emerge



Conventional programs as embedded in *processes* of interaction with the world

Programs are understood in relation to processes in their surrounding environment



Artefacts and their referents as sculpted out of open interaction with the world

States of the referent and the artefact are connected through experience of interacting with the referent and the artefact

… but this presents some philosophical challenges …

An EM perspective on programming …
… some problematic issues

In focusing on current state-as-experienced, we have some problems to resolve:

- Behaviour raises questions about agency: what is the status of a “computer” action?
- How do we deal with state-as-experienced in semantic terms?
- How do we make science of activities in which human interpretation is so critical?