

## Definitive notations and modelling with definitive scripts (MWDS)

1

## Background and History

A *definitive notation* = a simple formal language in which to express definitions

A set of definitions is called a *definitive script*

Definitive notations differ according to **types** of the variables that appear on the LHS of definitions and **operators** that can be used in formulae on the RHS. These are termed the *underlying algebra* for the notation.

2

## The definitive notation concept

- Todd relational algebra query language ISBL
- Brian & Geoff Wyvill's interactive graphics languages
- spreadsheets
- style definition in word processors

The term "definitive notation" first introduced by Beynon

"Modelling with Definitive Scripts" is fundamental to EM  
[Rungtanaubol's PhD Thesis: **A treatise on MWDS**]

3

## Related developments

spreadsheets with visualisation mechanisms

spreadsheet-style environments for end-user programming (e.g. AgentSheets)

generalised spreadsheet principles in application-builders (e.g. ACE)

"object-linked embedding" in Windows

4

## What does definitive mean?

definition has a technical meaning in this module  
definitive means "definition-based"

"definitive" means  
**more** than informal use of a programming technique.

Definitive notations are  
a means to *represent state* by definitive scripts  
and *how* scripts are interpreted is highly significant.

5

## 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  
rather than  
declarative techniques for *program specification*.

[cf. 'admira' application and contrast with KRC]

6

## Definitive notations

The tkeden interpreter uses many definitive notations

eden: scalars, strings, lists

DoNaLD: for 2-d line drawing

SCOUT: displays, windows, screen locations, attributes

EDDI: relational tables and operators

ARCA: edge-coloured digraphs in n-space

7

## DoNaLD: a definitive notation for line-drawing

Donald = a definitive notation for 2-d line-drawing

underlying algebra has 6 primary data types:  
**integer, real, boolean, point, line, and shape**

A **shape** = a set of points and lines

A **point** is represented by a pair of scalar values {x,y}.

8

## Defining shapes in DoNaLD

Two kinds of shape variable in DoNaLD:  
these are declared as **shape** and **openshape**

An **openshape** variable S is defined componentwise  
as a collection of points, lines and subshapes

Other mode of definition of shape in DoNaLD is  
shape RSQ

RSQ=rotate(SQ)

- illustrated in definition of vehicle in VCCS model.

9

## Reference and Moding

A definition relates a reference to a value

*reference = value*

A definitive variable differs from

- traditional procedural variable  
meaningful only during the execution of a program
- declarative variable  
statically defined, independent of program execution.

*Definitive variables aim to support references*

*similar to observables as we use them in everyday life.*

10

## Reference and Identity

Reference is associated with a concept of identity

Cf mathematical concept :

double point on a self-intersecting curve

Script can have *distinct* variables with *same* values:

a = b

b = 3

c = 2\*a-b

.....

each with a different "identity" and significance

11

## Modes of definition

*mode of definition*

= the way in which reference and value are associated

form of a definitive notation not determined by the  
underlying algebra alone

possible modes of definition also important

..... there are many ways to define a complex structure

12

## Defining a complex structure

Examples of list definitions in EDEN

- define a list in its entirety:  
list1 is reverse(list2)
- give a list of recipes to define a list component-wise:  
list3 is [l1,l2,l3]; l2 is 2\*l3; l1 is 2
- use a recipe combining the two modes of definition:  
list4 is [list1, list3, [l5,list5]]; l5 is 7; list5 is [l5].

13

## Issues in mode of definition

*Can list components be treated as independent variables?*

Should we be able to rewrite the previous definition

list4 is [list1, list3, [l5,list5]]; ...

as list4[1] is list1; list4[2] is list3; list4[3] is [l5,list5];

Should the definition 'list4[1] is list4[2];' be deemed cyclic?

Many different modes of definition => potential inconsistency.

For instance:

list1 is reverse(list2); list1[1] is 3;

involves two independent definitions of list1[1]

14

## Different modes of definition

Different modes of definition represented in tkeden:

- in shape and openshape in DoNaLD
- in EDEN, components of a list can't be defined via  
l[1] is ...
- in ARCA the mode of definition of a variable is itself specified in a definitive notation over an underlying algebra of modes
- In EDDI, can't express dependency within tuples

15

## Agents and semantics

Archetypal use of MWDS: human-computer interaction  
*"single-agent modelling"*

Variables in a definitive script represent

- the values that the user can observe
  - the parameters that the user can manipulate
  - the way that these are linked indivisibly in change
- ⇒ definitive script can model physical experiments

[cf the role of spreadsheets in describing and predicting]

16

## Environment not document

A script = an **environment** rather than a document.

In a document:

meaning of a symbol has to be represented in a stateless fashion  
the **reader** animates it by studying the contexts in which it occurs

In a definitive script:

explore significance of symbols via experiment and observation

17

## Concurrent Systems Modelling

In EM for concurrent systems modelling:

generalise MWDS for the user-computer interface  
to model the relationship between all interacting agents

each agent-system interface is treated as a domain for experiment

*"multi-agent modelling"*

18

## Objects vs observations

A definitive script  
represents the atomic transformations of a geometric symbol

DoNaLD room can be transformed through redefinition in ways that correspond 'exactly' to the observed patterns of change associated with opening a door, or moving a table.

Thesis:  
set of atomic transformations of a symbol captures its semantics  
of Klein's view of a geometry

The digit eight vs the floor-plan of a filing cabinet: a geometric pun

19

## Is the DoNaLD room an object in the OOP sense?

Can view each room transformation as a method for the object  
BUT definitive script is an object specification only if  
`set_of_transformations_performed_on_room` **circumscribed**

Circumscription creates objects  
BUT  
definitive modelling merely records observed transformations

Comprehending an object = knowing everything we can do with it  
BUT  
definitive script doesn't circumscribe transformations we can apply

20

## Object model vs. account of observation

An account of observation is the more primitive concept:  
entails fewer preconceptions about what might be observed

**"Definitive scripts neutral wrt agent's views & privileges"**

definitive script differs from an object:  
can express different agent views and privileges to transform

What architect can do to the room layout (e.g. relocate the door)  
vs what the room user can do (e.g. open/shut the door).

⇒ significance of script *relative* to view of possible transformations

21

## Variable values, observations and state

Definitive variables

- correspond to observables of phenomena external to the computer system
- have an identity and a value that can change according to the circumstances of observation.

The term *state* refers to what we understand by  
*a set of observations made 'simultaneously'*

22

## Interpreting the current state

The current state =  
*what I deem* to be simultaneous observations

The concept of state is  
relative to the observer ("observing agent")  
relative to focus of attention and mode of observation

A definitive script can represent several states at once

23

## Broad objective in MWDS

Use as a basis for universal agent-oriented modelling ...

Possible applications:

use definitive scripts as primitive device to represent a whole range of abstractions in PL design and development.

perception of state in everyday observation - concurrent perceptions of state of Miranda script + evaluated function

uses of reference that arise in mathematics: exposition of proof has never been [can never be] formalised

24

## Definitive notations: Scout

Design of Scout is described by SY in Intro to Scout  
Jugs interface as a case study in modelling with Scout  
See roomviewerYung1991 for demonstration of features  
Cf use of windows to represent jigsaw pieces  
~wmb/public/projects/games/jigsaw  
Discussion in Rungrattanaubol's PhD: script as artefact

25

## Definitive notations: EDDI

EDDI based on Todd's ISBL

definition = relational database view  
relational operators: +, -, \*, %, :

cf timetable application in pure EDEN: complex list ops

Uses Chris Brown's agent/observation-oriented parser  
(see implementation of SQLEDDI [sqleddiBeynon2001])

26

## Definitive notations: ARCA

ARCA was designed by Fahranak & Beynon early 80s  
Re-interpreting observation of line drawing (cf Donald)  
Exposed the issues of moding in mid-80s  
First implementations by Kevin Murray c.1985  
Most recent implementation by Stuart Bird early 90s

27

## symcubeWong2001

Model illustrates combined use of ARCA and SASAMI

Note the many possible interpretations of nodes in S4:

- points in 2-space and 3-space
- permutations of {1,2,3,4}
- transformations of cube
- matrices (see the text output window)

Matrix algebra manipulation implicit in model: re-use?

28