

Modelling with Definitive Scripts

About Definitive Scripts

Definitive scripts

Use scripts of definitions to represent state
Use redefinition to specify change of state

Scripts make use of definitive notations:

- DoNaLD - line drawing
- SCOUT - window layout
- ARCA - combinatorial graphs

Each notation is oriented towards a different metaphor

About Definitive Scripts

Definitive notations

Definitive notations are simple languages within which it is possible to formulate definitions for variables ("observables") of a particular type.

A definitive notation is defined by

- an underlying set of data types and operators
- a syntax for defining observables of these types.

Review/illustrate key features of DoNaLD and SCOUT

About Definitive Scripts

DoNaLD data types

Donald is a definitive notation for 2-d line-drawing
Its 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 $\{x,y\}$.
Points can be treated as position vectors: they can be added ($p+q$) and multiplied by a scalar factor ($p*k$)
A **line** $[p,q]$ is a line segment joining points p and q

About Definitive Scripts

DoNaLD operators

The DoNaLD operators include:

arithmetic operators:

$+$ $*$ div $\text{float}()$ $\text{trunc}()$ $\text{if ... then ... else ...}$

basic geometric operators:

$\cdot 1$ $\cdot 2$ $\cdot x$ $\cdot y$ $\{, \}$ $[,]$ $+$ $*$
 $\text{dist}()$ $\text{intersects}()$ $\text{intersect}()$
 $\text{translate}()$ $\text{rot}()$ $\text{scale}()$
 $\text{label}()$ $\text{circle}()$ $\text{ellipse}()$

A DoNaLD file should begin with a "%donald"

About Definitive Scripts

DoNaLD syntax – points and lines

declaring (NB) and defining points and lines

point o , p , q , m

line l

$l = [p,q]$

$m = (p+q) \text{ div } 2$

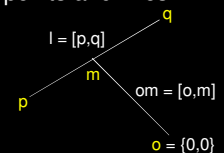
line om

new declarations can be introduced at any stage

$o = \{0,0\}$

$om = [o,m]$

.....



About Definitive Scripts

DoNaLD syntax – shapes

```

openshape S
within S {
  int m # this is equivalent to declaring int S/m outside S
  point p, q
  openshape T
  p = {m, 2*m}
  within T {
    point p, q # this point has the identifier S/T/p
    p, q = ~/q, ~/p
    # a multiple definition: p = ~/q and q=~/p
    # ~/... refers to the enclosing context for T
    # viz. S, so that ~/p refers to the variable S/p
    ....
  }
  ...
}

```

About Definitive Scripts

DoNaLD extras

Can define shapes in another way also: e.g.
shape rotsquare = rotate(SQ,...)
 where SQ is defined to be a square

The “**within** X { ...” context is reflected in the input window in EDEN

A syntax error in a ‘**within**’ context resets to the root context ...

... *there are NO SEMI-COLONS (;) in DoNaLD !!!*

About Definitive Scripts

SCOUT types

SCOUT is a definitive notation for screen layout

Its primary data type is the **window**

Other types include: **display** (collection of windows, ordered according top to bottom);
integer, **point** and **string**.

Windows are generally used to display text or DoNaLD pictures.

About Definitive Scripts

SCOUT screen definition

Overall concept

a SCOUT script defines the current computer screen state
screen is a special variable of type *display*
 the display is made up out of windows

Simplest definition of **screen** has the form
screen = < win1 / win2 / win3 / win4 / win5 / >
 where ordering of windows determines how they overlay

Alternatively can define **screen** as union of displays
screen = disp1 & disp2 & disp3 & disp4 &

About Definitive Scripts

SCOUT window definitions

A SCOUT window definition takes the form

```

window X = {
  fieldname1: ...
  fieldname2: ...
  ...
}

```

where the choice of *fieldnames* depends on the nature of the window content.

About Definitive Scripts

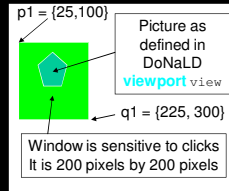
Defining a window to hold a DoNaLD picture

DoNaLD Window		
field name	type	description
type	content	Must be the value DONALD
box	box	The region in which the DoNaLD picture is shown
border	integer	Set the border width of the bounding box
pict	string	The name of the DoNaLD picture
xmin	point	
ymin	point	Show the portion of the DoNaLD picture
xmax	point	bounded by the points (xmin, ymin) and (xmax, ymax)
ymax	point	

About Definitive Scripts

A simple SCOUT DONALD-window

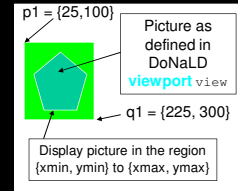
```
point p1 = {25, 100};
point q1 = {225, 300};
window don1 = {
  box: [p1, q1],
  pict: "view",
  type: DONALD,
  border: 1
  bgcolor: "green"
  sensitive: ON
};
# locations of points are in pixels from top left of screen
# coordinates of DONALD picture {0,0} to {1000, 1000}
```



About Definitive Scripts

Another SCOUT DONALD-window

```
window don2 = {
  box: [p1, q1],
  pict: "view",
  type: DONALD,
  xmin: zoomPos.1 - zoomSize/2,
  ymin: zoomPos.2 - zoomSize/2,
  xmax: zoomPos.1 + zoomSize/2,
  ymax: zoomPos.2 + zoomSize/2,
  border: 1
  sensitive: ON
}
```



About Definitive Scripts

Defining a window to hold text

Text Window		
field name	type	description
type	content	Must be the value TEXT
string	string	The string to be displayed
frame	frame	The region in which the string is shown
border	integer	Width of the border of the boxes of the frame
alignment	just	NOADJ, LEFT, RIGHT, EXPAND and CENTRE are the possible values to denote no alignment, left justification, right justification, left and right justification and centre of the text inside each box in the frame
bgcolour	string	Colour name for the background colour of the text
fgcolour	string	Colour name for the (foreground) colour of the text

About Definitive Scripts

A simple SCOUT TEXT-window

```
window doorButton = {
  frame: ([doorButtonPos, 1, strlen(doorMenu)]),
  string: doorMenu,
  border: 1
  sensitive: ON
};
string doorMenu = if _door_open then "Close Door" else "Open Door" endif;
```

About Definitive Scripts

SCOUT extras

When aspects of the screen are undefined by the SCOUT script, it will not be drawn / redrawn

Sensitive SCOUT windows generate definitions of associated mouseButton variables: they supply information about the mouse state and location & can be used to trigger EDEN actions

Mouse clicks show up in the command history

About Definitive Scripts

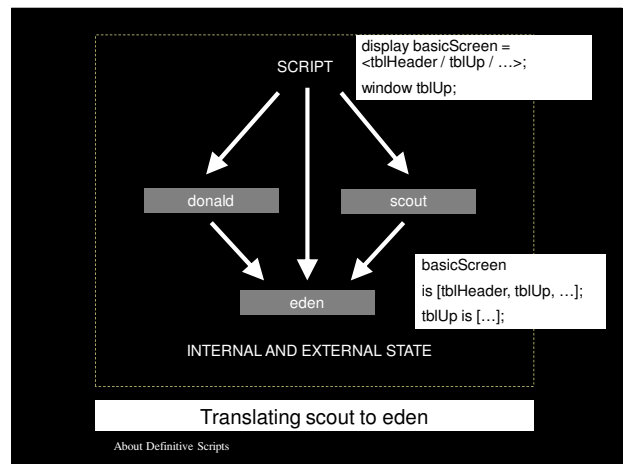
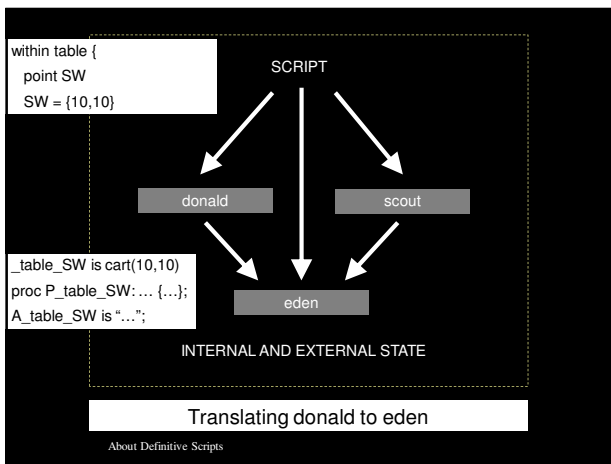
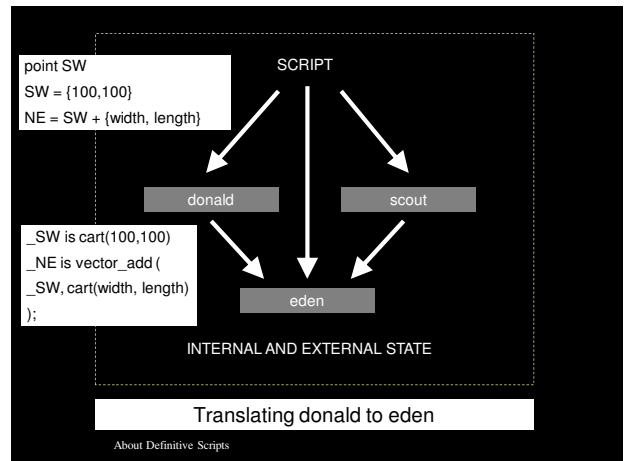
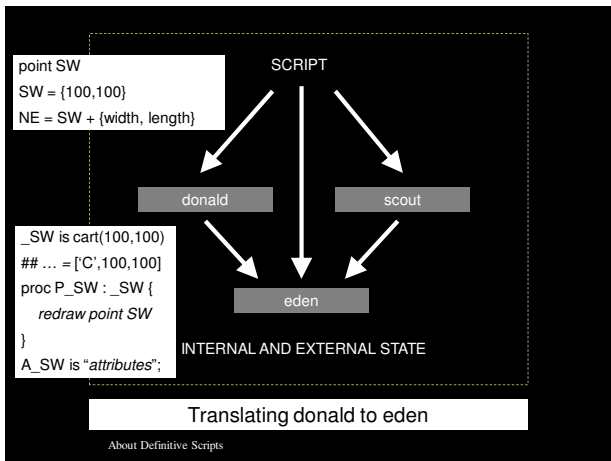
SCOUT & DoNaLD extras

By default, a DoNaLD picture is displayed in a system generated SCOUT window, and has coordinates between {0,0} and {1000,1000}

SCOUT observables can be accessed in EDEN by the same names

A DoNaLD observable X/t can be accessed in EDEN and SCOUT by `_X_t` etc.

About Definitive Scripts



Examples of definitive notations

Notation

eden
donald
scout

arca
sasami
eddi

Basis for underlying algebra

scalars, recursive lists, strings
points, lines, shapes
windows, displays
(window = template + content)
diagrams, vertices, incidences
polygonal meshes, renderings
relational database tables and views

Each notation is adapted to the metaphorical representation of different kinds of observable