D. Definitive Programming

programming as modelling: vars give info re objects modelling, + also uninterpretable
Jordan's agent-oriented analysis 1-3 is LSD, but not operational model at this point

What is definitive programming?

a. Definitive Programming "for concurrent systems simulation"
b. Definitive Programming "for distributed programming"
c. Definitive Programming "for sequential programming"

... neither a nor b is really programming in some sense (theme of this course)
i.e. ultimately may get useful circumscribed automatic behaviours, but typically need super-user intervention etc

The more interesting question concerns c. Is there a programming paradigm that serves a similar function to procedural / functional programming (of defining algorithmic solutions to traditional tasks)?

procedure calling strategies	ways of invoking transient agents
for-loops	observing several agents simultaneously active in similar contect (with differing index variable)
assignment	recording effects of agent action, commitment
recursion	distributing server viewpoint to clients, new data
data structures	regimes for interaction with observables

Definitive scripts as states, and procedures / assignments as actions / transitions

Updating of script is uninterpreted computation (conceptually discounted)
DP as "organising the uninterpretable computation into coherent chunks"

Examples

a. OXO b. Replicator c. Heapsort

Client-server use / virtual agent abstractions assist the solution of such problems
NB issue is not to achieve concurrent execution, but to distribute observation
(cf info hiding, local variables of procedures)

Identification of disadvantages excellent

a. efficiency (though trade-off with flexibility)

envisage translation techniques (Trident etc, OXO Pascal)
DAM and parallisation (Leiserson, MIT)
knowledge representation for procedural program specification
model -> program, change program requirement -> refine model

b. interfacing with conventional programs

certainly problematic at present
(cf screen update, evaluation of inconsistent formulae)

c. lack of data structures

encapsulation difficult area connected with strategies for data management
cf Concurrent Engineering

ARCA moding issues, CADNO

A would-be case-study in Definitive Programming

The files in the REPLICATOR directory in $PUBLIC/projects/misc illustrates the concept and some of the issues of definitive programming. They represent "work in progress", rather than a completed program.

The objective of the scripts in this directory is to construct an environment in which it is possible to decompose a piece of text displayed in a window into a list of segments by using mouse interactions. (The broad idea is that a piece of text has a conceptual structure that the observer would like to incorporate in a model without having to explicitly construct in internal representation in a laborious fashion.

The functionality of the model can be illustrated by first invoking tkeden with the files inwin.e, then including ident.e and assigning the variable w1str to teststr, then including the files mvwin.e and segwin.e. At each stage of this process, different elements of the functionality can be demonstrated:

After inwin.e has been included, it is possible to key text into the (initially empty) text window on the screen (w1), after clicking the left-hand mouse button on the window.

After ident.e has been included, and w1str assigned, it is possible to use the left mouse button to select segments of text from a line of text in the window w1. This segment will then be displayed in a second window w2. The partition of the input text into substrings is recorded in a variable called partitionstr, which is a list of three strings.

After mvwin.e has been included, windows can be moved around the screen by using the middle mouse button.

After segwin.e has been included, the segment selected from the text in w1 is highlighted in another window (segwin) that is displayed on top of the text.

At each stage, the environment that has been constructed is a simple example of how a definitive script can represent a programmer's concept of "current state of the model" with respect to functionality (cf. functional programming). The problematic aspects concern the interrelationship between these states.

Issues to be explored:

1. What happens when w1str is assigned to the empty string after a selection in Stage 2?

2. What would be needed to develop the process recursively, so that selection can be applied to the first or third segments of partitionstr?

3. What is the significance of the singular behaviours that can arise when (for instance) we attempt to assign w1str to partitionstr[1] in order to explore the recursive decomposition process? When eden fails to execute a procedure successfully, what does this mean conceptually? For instance does it signify a bug in the specification (programmer's conceptual error), or a failure due to limitations of the procedural paradigm when handling incomplete or inconsistent information? (For instance, the screen doesn't update unless every relevant variable is defined.)

4. What happens if, at the final stage, w1str is set-up as a string with a single line of text, and the entire text is selected? Can you give an easy fix for this?