QUTOR: A Computer Program for Presenting Tutorials

RONALD G DUGGLEBY

Department of Biochemistry
University of Queensland
St. Lucia, Queensland 4067
Australia

Introduction

There is considerable interest in developing computer-based instructional systems and several such programs have been described in recent issues of *Biochemical Education*. In most cases, these programs are designed to teach one or a few specific topics such as acid/base dissociation, enzyme kinetics, and so on. Typical of this genre is the program described by Chaplin[1] and the three programs discussed by Spencer.[2] While these programs may be excellent for the particular application that their authors had in mind, they suffer from the disadvantage that they do not clearly separate *process* from *content*. It follows that construction of a tutorial on a new topic (say, haemoglobinopathies) requires that the program itself must be substantially modified or even entirely rewritten.

The purpose of this report is to describe an extremely simple and completely general program for presenting tutorial material. The tutorial material itself and information controlling the logic flow is contained in a series of text files. This organisation allows for flow paths of essentially unlimited complexity.

This concept of a general program which interprets a series of text files is far from unique and other educational programs have been described[3,4] which adopt this principle. What makes the present program unusual is its great simplicity. It has been given the name QUTOR, a corruption of 'Queensland University Tutor'.

Organisation

In order to describe the program, it is first necessary to understand the organisation of the text which is to be presented to the student. It is assumed that a mass storage device, usually some sort of disk, is available and that information will be presented on a visual display unit.

Text files The text, which consists of background information, questions, hints, rewards, reprimands, compliments, threats, summaries, and so on, is divided into a series of pages. Usually, the information displayed on the computer screen will correspond to one page although it is possible to build up the image on the screen by combining two or more pages. Each page is stored in a file named *QUTOR.xxx* where *xxx* is an *identifier* consisting of a unique set of up to three characters. The contents of a typical file, taken from a tutorial on the control of carbohydrate metabolism, is shown in Fig 1. Immediately following the text in each file is a control character (?, + or $) in the first column. The $ control character indicates termination and all flow paths must eventually lead to a file which contains this terminator. The + control character indicates that the text from another file is to be appended to that which is currently on the screen. On the line following the + is the identifier (up to three characters) of the file which is to be appended.

```
Release of epinephrine (adrenaline) into the circulation
stimulates liver adenylate cyclase leading to a rise in
intracellular cyclic AMP.  This in turn:

1  Phosphorylates a protein kinase
2  Activates a protein kinase
3  Causes glucagon to be broken down
4  Stimulates glycogen synthesis
5  Activates a lipase

If none of these is correct, type N
If all five are correct, type A
If only one is correct, type its number (1-5)
If you would like some help, type H
?
12345AHN
923
924
925
926
927
921
934
922
```

Figure 1 A typical text file for the QUTOR program. This example is taken from a tutorial on the control of carbohydrate metabolism. The student sees only the information preceding the line containing a single question mark. The remainder of the file is control information which is interpreted by QUTOR. The ? indicates that a keyboard response is expected and the following line lists all valid replies. The remaining codes are file identifiers which match on a one to one basis with the valid replies. Thus, if the student's reply is H (the seventh of the valid replies) the seventh identifier (934) is located and the text in file QUTOR.934 would be presented to the student. In this example, QUTOR.934 contains a short summary of the effect of cyclic AMP since the student's reply indicated that help was required. In a similar way, if the student reply was 5, the text from the file QUTOR.927 would be presented which would praise the student's knowledge of adipose tissue metabolism then point out that the current question is concerned with the effect of cyclic AMP in liver.

The ? control character indicates that a keyboard response is expected from the student. The response must be a single character and the next line of the file is a list of valid responses. In the example shown in Fig 1, this list is *12345AHN* indicating that any of these eight possible responses is acceptable. Following the list of valid responses are a series of file identifiers, each of which corresponds to one of the responses. In the present example, there are eight identifiers (one per line); these can be all different, all the same or some combination. Frequently, the text will consist of some background information which is to be held on the screen long enough for it to be read by the student. In this situation it is desirable that *any* keyboard response be considered valid and this is indicated by replacing the list of valid responses with the character &. A single file identifier would be present on the following line of the file.

Figure 2 shows a very simple example of the organisation of a series of pages and the flow paths connecting

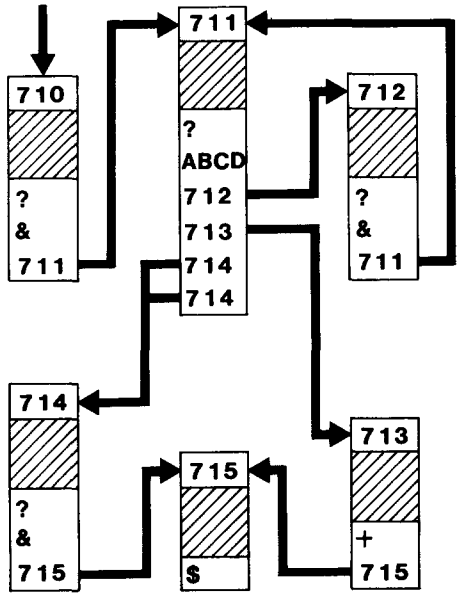

Figure 2 A simple example of the organisation and logic flow of QUTOR text files. Each large box indicates a page of text associated with the identifier shown at the top of that box. The identifier can be up to three alphanumeric characters although I routinely use three digit identifiers. The cross-hatched area represents the text which would be presented to the student and is detailed below. The lower section of each box contains the control codes. The entry point is at page 710 which presents some preliminary information and invites the student to key in any response whereupon a question is posed from page 711. In this case there are four possible responses; A is completely wrong and B is the correct answer, while C and D are partially correct. If response A is received, the student is shown page 712 which explains why that answer is wrong; subsequently, the question is repeated. Response B leads to page 713 which contains an appropriate congratulatory message to which is appended page 715, a summary. In this example, the flow path ends here. Response C or D each cause page 714 to be displayed which indicates that the student's response is only partially correct with some supporting arguments. Thereafter the student is presented with the summary on page 715.

them. Obviously, any real tutorial would consist of many more pages and might have considerably greater complexity in the logic flow. This particular example is a slightly adapted section from a tutorial on immunoglobulins which consists of 55 pages and up to eight valid responses.

Program The program, which is written in BASIC, is shown in Fig 3. It begins (lines 100–120) by performing some preliminary initialisation, setting the identifier to the string *MEN* so that the first page to be presented is in the file *QUTOR.MEN*. This will usually contain a menu of the tutorials which are available. Lines 130 and 140 open the current file. Line 150 checks whether a screen clearing operation is required and line 160 clears the screen. This latter operation is specific for a particular type of terminal

```
100 F$="QUTOR."
110 E$="MEN"
120 L$=""
130 I$=F$+E$
140 OPEN I$ FOR INPUT AS FILE #1
150 IF L$="+" THEN 170
160 PRINT CHR$(27)+"H"+CHR$(27)+"J"
170 LINPUT #1,S$
180 L$=SEG$(S$,1,1)
190 L=POS("?+$",L$,1)
200 ON L+1 GOTO 210,230,310,350
210 PRINT S$
220 GOTO 170
230 LINPUT #1,S$
240 LINPUT R$
250 IF S$="&" THEN 310
260 IF R$="" THEN R$="\"
270 P=POS(S$,R$,1)
280 IF P<>0 THEN 320
290 PRINT "Invalid reply ... try again";
300 GOTO 240
310 P=1
320 FOR L=1 TO P
330 LINPUT #1,E$
340 NEXT L
350 CLOSE #1
360 IF L$<>"$" THEN 130
370 END
```

Figure 3 The QUTOR program

(a VT52) and may need modification to suit local hardware. For many systems, line 160 would be simply the *CLS* statement. Lines 170–220 read the text file and copy it to the screen until one of the control characters is detected in column 1 (lines 180–200). It should be noted that the *POS* function in line 190 is equivalent to *INSTR*(1,"?+$",L$) which other dialects of BASIC may use.

If the ? character is found, the list of valid responses is read (line 230) and no further action is taken until a keyboard response is received (line 240). If any response is to be considered valid (line 250), the program jumps immediately to reading the identifier (lines 310–340); otherwise the response is checked against the list of valid responses (lines 270 and 280). Receipt of a null response (ie simply pressing the RETURN key) is translated to a back-slash (\) in line 260; the back-slash may or may not be in the list of valid responses. An invalid response is flagged with a message (line 290) while a valid response is used to locate the appropriate identifier from the file (lines 320–340). The current file is then closed (line 350) and the program returns to the point where it opens a file. The file which is opened is that which is specified by the identifier which has just been read.

Detection of the + control character circumvents most of this logic; the file identifier is read, the current file is closed, the new file is opened and this is copied to the screen without clearing it first. Detection of the $ control character leads to termination of the program (lines 350–370).

Depending on local hardware and software, some modification of this program may be necessary. Screen clearing and the *POS* function (lines 190 and 270) have already been mentioned in this context and the *OPEN* statement (line 140) may also need to be adapted. Other possible changes, with their common variants, are *LINPUT (LINE INPUT)* in lines 170, 230, 240 and 330; and *SEG$ (MID$ or LEFT$)* in line 180.

Discussion

It will be evident from Fig 3 that the program is extremely short, almost embarrassingly so. One should not confuse brevity with triviality for the program is capable of presenting tutorials of substantial complexity. The small size of the program, coupled with the fact that no arrays and only a handful of variables are used, makes the memory requirements within the range of the simplest (and cheapest) of home computers.

The program described in this report performs extremely well although there are many additional features which users may wish to include. Indeed the version which is currently used in this department has been altered considerably to accommodate the strengths and weaknesses of local hardware (IBM Personal Computers). For example, each page of text is not stored as a separate file as I have described here; rather several pages are grouped together in one large file which is searched for the appropriate page. While this increases access time somewhat, it allows for a much more efficient usage of disk space. The few seconds delay that this introduces is insignificant in relation to the several minutes that a student may need to assimilate information from the screen.

The program currently in use has a number of additional features, some of which are listed below.

(1) Text files can contain coded graphics information which are interpreted by the program to produce diagrams, biochemical pathways, structures, and so on.

(2) There is a 'go back' facility which allows the student to return to an earlier page to check on something which may have been missed.

(3) The student may stop part way through, then resume the tutorial at a later time. This is an important point since it gives the student the feeling of being in control, rather than just a slave to the computer.

Some features are still under development. For example it is eventually planned to permit students to reply with words, phrases or sentences rather than single character coded replies. As a first step in this direction, the version which is currently in use allows for numerical replies consisting of several digits, with or without a decimal point. Other areas of possible development are to incorporate simulations and animation.

The implementation of many of these features, especially the graphics, depends on the particular hardware on which the program is run. It is for this reason that only the simplest version of the program is described here. No doubt this will be adapted to local circumstances. While the program has been developed with a view to teaching biochemistry, it could equally well be used in most disciplines or even as the basis of, for example, a television repair manual.

Acknowledgements

I would like to thank Dr Leigh Ward and Dr John de Jersey for their comments on this report, and Professor Brian Shanley for stimulating my interest in computer-assisted learning.

References

[1] Chaplin, M F (1983) *Biochemical Education* **11**, 2

[2] Spencer, T (1983) *Biochemical Education* **11**, 27

[3] Hliwa, W R, Holden, D M and Klick, R L (1982) *Laboratory Medicine* **13**, 246

[4] Raj, P P, Kricka, L J and Clewett, A J (1982) *Medical Education* **16**, 332

Occam's Razor and Theories of Allostery

GEOFFREY D SMITH

Department of Biochemistry
The Australian National University
GPO Box 4, Canberra, ACT 2601, Australia

Summary

The concept of an allosteric site, first invoked by Monod and coworkers, has been widely used to explain the kinetics of regulatory enzymes. However, the number of enzymes demonstrated to possess distinct allosteric sites is small and it is shown that the simple two-state model can explain most types of kinetic behaviour found in regulatory enzymes without any need for such sites. Occam's razor is invoked to suggest that any *a priori* assumption about the mechanism by which an inhibitor or activator exerts its effect should assume that the enzyme possesses only an active site at which substrates and effector molecules each bind.

Introduction

The catalytic activities of many enzymes are modulated by effector molecules such as end products of metabolic pathways. Theories to explain such modulation have proliferated and, in general, experimental evidence for them has lagged far behind, leading Whitehead[1] at one stage to lament that "If a branch of science goes, according to the nomenclature of Stent (1968) through Romantic, Dogmatic and Academic phases this development has in the case of 'allosterism' been compressed within less than ten years". Since that time still relatively few enzymes have been studied by physical methods to a sufficient extent to enable their regulatory mechanism to be defined unequivocally.

Many of our concepts of regulatory behaviour grew out of the work of Monod and coworkers[2,3] in the early 60s, their major hypothesis being that regulatory enzymes possessed an 'allosteric site', distinct from the catalytic site, at which regulator molecules bind. They introduced the concepts of a *homotropic* effect, which represents the sigmoidal kinetics observed with substrate(s) alone, and a *heterotropic* effect which results from the presence of an inhibitor or activator binding at the allosteric site. Thus, there are two aspects of their model: (1) that an enzyme exists in two conformational forms with different catalytic activities and that substrate(s) binds preferentially to one of them, and (2) each form has an allosteric site to which regulatory molecules bind differentially. I wish to examine each aspect of this theory and to show that in many cases