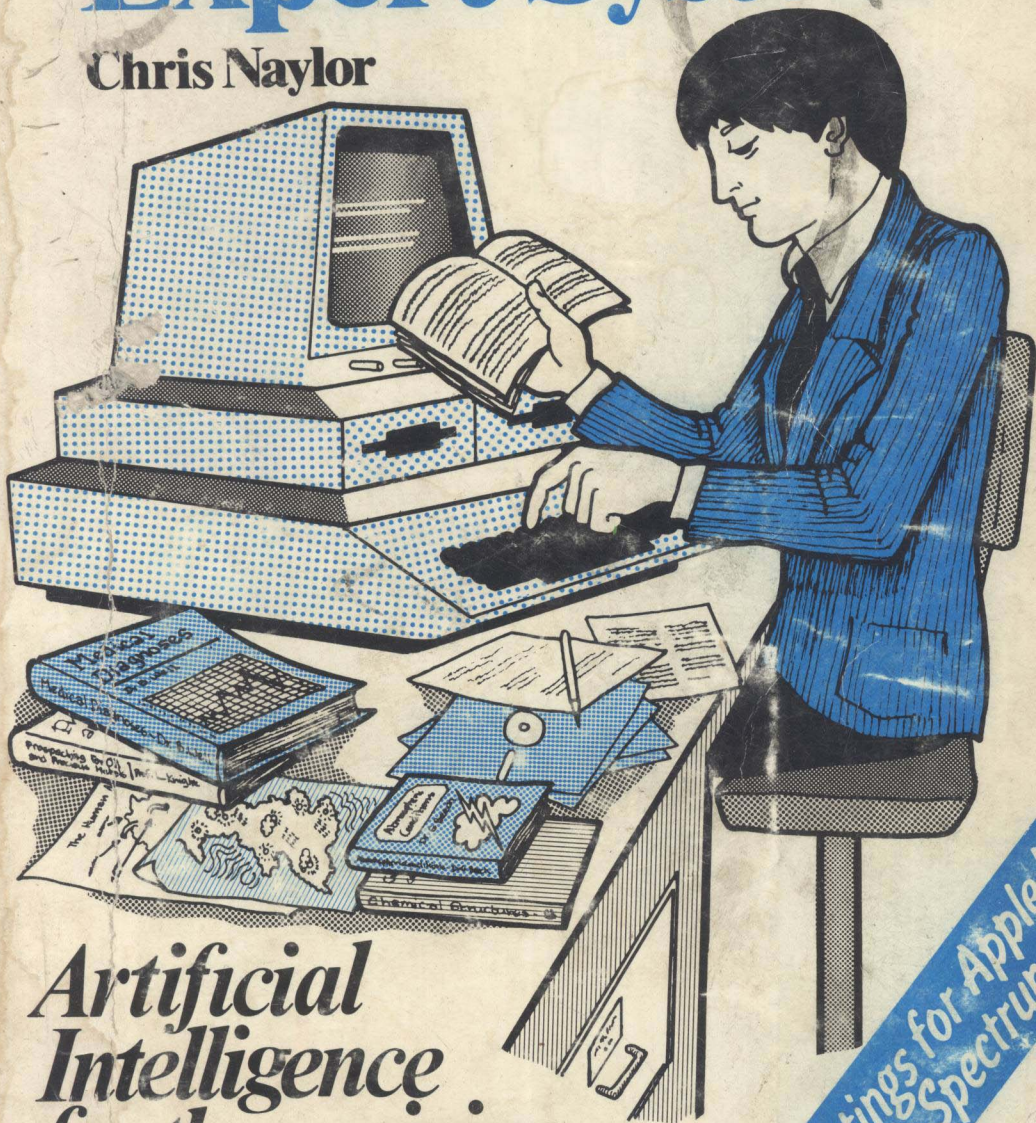


Build your own Expert System

Chris Naylor




*Artificial
Intelligence
for the aspiring
microcomputer*

*With listings for Apple
and Sinclair Spectrum*

Build Your Own Expert System

by

Chris Naylor

 Sigma Technical Press

Copyright ©1983 by C.M. Naylor

All Rights Reserved.

No part of this book may be reproduced or transmitted by any means without the prior permission of the publisher. The only exceptions are for the purposes of review, or as provided for by the Copyright (Photocopying) Act or in order to enter the programs herein onto a computer for the sole use of the purchaser of this book.

ISBN 0 905104 41 2

Published by:

SIGMA TECHNICAL PRESS,
5 Alton Road,
Wilmslow,
Cheshire,
U.K.

Distributors:

Europe Africa:
JOHN WILEY & SONS LIMITED,
Baffins Lane, Chichester,
West Sussex, England.

Typeset by:

Designed Publications, 75, Wilmslow Road,
Handforth, Wilmslow,
Cheshire. SK9 3EN

Printed in Great Britain by J. W. Arrowsmith Ltd., Bristol BS3 2NT

PREFACE

A potential reader is standing in a bookshop with a copy of a possible purchase in his or her hand. Having looked at the front cover, and looked at the back cover, he or she now reads the Preface to determine whether or not the book should be purchased. The Preface, says the Theory, clinches the sale.

So much for theory. The real reason you should buy this book is because, for a book on computers, it is relatively cheap. It also contains working examples in BASIC for both the Apple II and the Sinclair Spectrum so you get some 'free' programs thrown in for your money.

It tells you a moderate amount about Expert Systems, but, frankly, to disclose exactly what it does tell about Expert Systems would rather annihilate the reason for buying it. After all, you could just stand there, in the bookshop, reading the preface and hang onto your money. But it will (just as a sort of an appetiser) enable you to build your own medical diagnosis program. Or a program for working out why your car won't start in the morning. Or it will enable you to build a learn-by-example Expert System which can be taught expertise in a wide range of areas.

It will also teach you a fair amount about statistics and inferencing systems but, despite that, you should still shell out the necessary and buy a copy.

I know money is tight these days and you could usefully spend it on something else, like drink, which would give *you* greater pleasure but I have to make a living too you know and the cost of typewriter ribbons alone was pretty enormous when it came to bashing this lot out.

Well, after all that, maybe you've had the heart to fork out on a copy and you're actually planning to read the thing now. What you do is start at the beginning and carry on until you get to the Technical Overview. Then think of something you'd like to try out — such as a learning system or a bit of problem diagnosis — and dig out the relevant parts using the contents pages, the index, and the technical overview to tie it together.

Alternatively, you could have your computer switched on as you read the book and key in the examples as you go along, that way seeing how they work as an aid to understanding. If you do this, it could take you ages to get to the end, of course.

The big point to note though is that this book is not arranged like a normal text book. The chapters are not isolated entities. One way or another you do have to churn right through from front to back to get the ideas in the proper sequence. A few people have commented that it reads more like a novel than a text book in the way it's arranged - which is a fair point, but it's probably as well to warn you that it's like that.

Prior to publication this book was read by Graham Beech, Phil Bradley, Bill Hudspith and Phil Manchester (in alphabetical order).

They all chipped in with comments of one sort or another and they each have caused some improvements to be made to the final version. Which was nice of them.

If anyone, having read the book, has any comments which might lead to useful future alterations, then drop the publisher a line and let him know.

Chris Naylor, 1983

CONTENTS

1. Why 'Expert Systems'?	1
1.1 What do You Want an Expert System for?	3
1.2 What do Other People Want an Expert System for?	4
1.3 What is an Expert System?	5
1.4 What do You Want Your Expert System to do?	9
1.5 Some Untrue Things about Expert Systems	11
 2. A Statistical Scheme	 13
2.1 Setting up a Matrix	13
2.2 Probabilities	20
2.3 More Probabilities	23
2.4 More Variables	26
2.5 Bayes' Theorem	33
 3. Avoiding Probabilities	 35
3.1 How to Make the Computer do the Hard Work	36
3.2 The Learning System	37
3.3 Other Types of Data	43
3.4 The Judgement Rule	45
3.5 Building a Rule	49
3.6 Prior Probabilities	53
3.7 Expanding your Options	54
3.8 Can it Make a Mistake?	58
3.9 Summing Up: The Program so far	61
 4. Improving your Expert	 65
4.1 Parallel and Sequential Decisions	65
4.2 Adding some Commonsense	72
4.3 A Trial Run of our New Expert	78

5. The Making of A Real-World Expert	89
5.1 The Weather Again	89
5.2 A Chi-Squared Program	98
5.3 Exercising your Expert	99
5.4 Direct Estimation	105
 6. Running for Real	 110
6.1 Using your Expert	110
6.2 Reserved Judgement	115
6.3 The Problem of Distance	117
6.4 Understanding your Problem	124
 7. An Expert on Everything in the Entire Known World	 125
7.1 Nodes	125
7.2 The Variables so Far	131
7.3 Going through the Nodes	137
7.4 Tailor-made Nodes	142
7.5 Specific Code	145
7.6 Saving your Expert	146
7.7 The Multi-Node Code	148
7.8 Some Examples	157
 8. How can you Use your Expert	 164
8.1 Choosing a Problem	164
8.2 Analysing the Problem	166
 9. Large - Scale Expert Systems	 170
9.1 MYCIN - Medical Diagnoses	170
9.2 PUFF - Breathing Disorders	178
9.3 DENDRAL - Chemical Structures	185
9.4 PROSPECTOR - Searching for minerals	190
9.5 Some Other Examples	196

10. A Rule-Based BASIC Expert	201
10.1 A System that Works Backwards	201
10.2 The BASIC Program	205
10.3 A Medical Knowledge Base	214
11. The Tower of Babel	221
12. Summary and Technical Overview	229
12.1 Events	229
12.2 Probabilities	229
12.2.1 Bayes' Theorem	230
12.2.2 Prior and Posterior Probabilities	231
12.2.3 Odds	232
12.2.4 Approximations	232
12.2.4 Combinations	233
12.2.5 Descriptive Statistics	233
12.2.6 Normal Distribution	234
12.2.7 Discrete and Continuous Variables	235
12.3 Surfaces	235
12.4 Discrimination	236
12.5 The Learning Algorithm	237
12.6 Parallel and Sequential Procedures	237
12.7 Maximum and Minimum Values	238
12.8 Processing Strategies	238
12.8.1 Goal Driven Strategies	238
12.8.2 Data Driven Strategies	239
12.8.3 Selecting the Next Variable	239
12.9 Intermediate Conclusions	240
12.9.1 Explanatory Systems	241
12.10 Linear Interpolation of Responses	241
12.11 Data Formats	242
13. Select Readings	244
Index	247

CHAPTER 1

Why 'Expert Systems'?

"An expert system is regarded as the embodiment within a computer of a knowledge-based component from an expert skill in such a form that the system can offer INTELLIGENT ADVICE or take an INTELLIGENT DECISION about a processing function. A desirable additional characteristic, which many would consider fundamental, is the capability of the system, on demand, to JUSTIFY ITS OWN LINE OF REASONING in a manner directly intelligible to the enquirer. The style adopted to attain these characteristics is RULE-BASED PROGRAMMING".

A formal definition of expert systems approved by the British Computer Society's committee of the specialist group on expert systems.

Once upon a time, a long time ago when the Earth was still new and the Sun had a big smile on it's face when it got up each morning, there was no such thing as Expert Systems. Yet, suddenly, everyone seems now to be talking about the things. Why is this?

Well, the answer is that Scientists Have Determined that there is an increasing quantity of Government Money around for computer applications and, further, they have also determined that there is scant chance of accessing this pile of money unless there is a goodish chunk of expensive and arcane research activity to be carried out on the application of computers. So, to this end, they invented Expert Systems which, because nobody in Governmental circles knows what they are, are liable to attract Government Money if only as an aid to identification.

Less mercenary scientists (whose names currently escape one) are not interested in Government Money *per se* and will, in fact, point to the notable absence of Government Money to date in this field. These altruists will simply state that they want to make computers more accessible to more people. They want to make computers *think* like people. They want computers to *replace* people. They want computers to be *user-friendly*. And, in the final analysis, they probably also want computers to take on the job of allocating Government Money.

All of this is fine - but the real problem lies in finding out enough about Expert Systems so that one can even begin to attract Government Money on one's own behalf. Beyond this point in the book the subject of Money will receive little attention but, hopefully, the book will enable you to get the hang of Expert Systems so that you can, at least, build your own - Government Funded or not.

The first part of the book relies on your total ignorance to build an Expert System - it is a learn-by-example system which can pick up skills in a wide range of subject areas. It *acquires* expertise.

Later, some knowledge of a specific area is assumed and we build an Expert System which is able to use this knowledge intelligently to offer *advice* after the fashion of a human expert.

A point which really must be made is this: that this book, as an aid to understanding and as a sop to sloth, gives all of its examples in BASIC. But it does not contain an overview of, or instructions on, how to program in BASIC. That is the one piece of expertise which the reader has to bring to the task for him or herself.

The main examples are given in BASIC with the major programs written in dialects for both the Apple II and the Sinclair Spectrum so they should be readily modifiable onto most micros. Very short sections of code are written just in Apple BASIC (i.e. Applesoft.)

The final chapter in the book summarises the information given in previous pages so that, when you get that far, you'll be able to see exactly what it was that you were reading about earlier.

1.1 What do you want an expert system for?

There are two major faults possessed by most existing expert systems and these two faults are: that you, personally, don't understand how they work, and that you, personally, haven't got one.

These faults can, in extreme cases, be quite serious.

You slink around avoiding other people's eyes. You avoid conversation with others. You hide behind COBOL manuals. You listen greedily to other's talk of 'knowledge bases', 'artificial intelligence' and 'real-world representations'. You are afraid to come clean and ask what it's all about for fear of social rebuff.

You become an outcast and a despised person in your own eyes.

All of which can become a bit irksome after the first five minutes or so. Particularly when you feel that the subject matter can't really be all that difficult as evidenced by the fact that those who do profess to understand it can't, surely, be any cleverer than you yourself are.

What is needed, to put matters right, is for you to have your Very Own Expert System. Tailor-made to A Little Known Design it will enable you to lean confidently on any bar counter and pontificate on the subject of expert systems. Instead of hiding your head and keeping your own counsel you will be able to raise a slight sneer in the direction of those who have expert systems but didn't build them themselves. You will be able to laugh condescendingly at those who don't really understand how their own expert system works. And at those who actually don't have an expert system of any kind (one doesn't even begin to know how such people manage to get by) you will be able to raise a deeply quizzical eyebrow.

For you (the tall, confident one standing near the centre of the bar and holding forth to a crowd of admirers, many of whom are young and nubile) are about to Build Your Own Expert System.

No special knowledge is required although it would be handy if you had access to a computer; otherwise much of what is to follow may come to seem, somehow, well, Academic.

1.2 What do other people want an expert system for?

There are two main uses for expert systems and these correspond to the two concepts of manifest and latent functions in sociology.

The manifest function of an expert system is to provide, on a computer, human expertise. For instance, they can diagnose illness, deduce chemical structures, suggest sites for digging up precious metals or carry out a host of similar tasks. They are user-friendly to some degree - embodying human knowledge in a form vaguely similar to the form in which a human expert might hold knowledge. They often have some ability to explain their actions and opinions in much the same way that a human expert might. And, like a human expert, they might even be able to teach their expertise to someone.

The other function of expert systems - hinted at earlier - is their latent function which is, one suspects, to baffle the ignorant with arcane explanations of how they were built. Typically, they use large computers, of the sort you have not got, and employ exotic-ish languages, such as LISP and PROLOG, which you have not got either. This tends to have the advantage of somewhat sewing up the market for supplying expert systems because, if demand can be stimulated by a description of an expert system's manifest functions, this demand - surely? - can't be met simply by anyone possessed of a micro and a BASIC interpreter. Which doubtless affects the price.

The way out of this problem is to provide a micro-orientated guide to building expert systems. Not a complete guide which gives the last word on the subject necessarily - but enough to break down some of the mystique and get the average person started.

Up to now the real problem in understanding expert systems has been simply that there was no simple place to go for an introduction to the subject. Unless you were prepared to put some pretty hard thought into the matter, all of the currently available writings on the subject just seemed to emphasise the difficulty of ever understanding any aspect of the subject - which tends to inspire one to give up and leave it to the experts! It's all rather reminiscent of trying to climb a mountain (well, hill, maybe) on which there is just no first foothold to be found. If you could only get started you'd feel a lot more optimistic about what might be done and you'd even be able to work out a lot of the subsequent steps for yourself without the aid of a book. And, after all, that's what the human experts on expert systems are doing: working it out for themselves.

The trick lies in getting yourself into the same way of thinking as the current leaders in the field. To realise that there is something concrete and accessible to think about with real footholds to hang onto. To think that there are some interesting problems here and that these problems, with a bit of thought, are perfectly soluble and that the solutions as they appear are perfectly amenable to being explained in plain language. That the subject of expert systems is not in fact a mystic art but, like all computer subjects, is something as practical and down-to-earth as carpentry.

1.3 What is an expert system?

Before you climb into your overalls and rush to your workbench it's as well to just pause for a moment and ask: what, actually, is an expert system? For, if you think about it, you'll realise that the answer to this question might well have a profound effect on the finished object. Actually, the answer won't affect the object much - as we shall see - but, certainly, you'd think it should.

It all started many years ago, back in the empty vastnesses of time when computers were about as powerful as a pocket calculator and transistors were spoken of in awed tones. In between being struck speechless by the power and complexity of the monsters with which they worked, computer scientists found words to express their deepest desires.

"Wouldn't it be nice," they used to say to each other, "if we could get this thing to do something other than payroll calculations."

"Yes," they used to agree, "we've got literally hundreds of words of memory and it now works faster than the chief accountant yet, for all that, it is like working with an idiot."

"Certainly he is an idiot," another would concede, "but our computer is little better than him."

And they would drink beer long into the night and attempt to hatch schemes whereby the full power of their Frankensteinian monster could be unleashed.

Well, they are still sitting there, drinking beer and plotting against the chief accountant and the computer is still doing the payroll. The progress has not been dramatic in the direction of getting the computer to do something more, but the motivation is still the same. For, like Frankenstein, they wanted to dream up a way of breathing a little life into

their subject. They didn't just want a glorified adding machine, they wanted an actual thinking machine.

That, of course, is how the whole field of artificial intelligence started up, and the field of expert systems was just a part of it.

People noticed that the chief accountant, say, wasn't just a glorified adding machine (despite the rumours) but that he was something of an expert in his own field. He could cook the books, for instance, in a way quite beyond the abilities of any computer.

And it wasn't just accountants that could beat computers. In every field there were human experts who had special skills and knowledge which made them both indispensable and expensive. Every time a group of computer scientists talked to someone who was an expert in some field they would listen to him for, maybe, a couple of hours. Then they would start to long for the day when he could be replaced by a computer so that they could switch him off, and then forget to pay him.

The dream was an alluring one. So alluring that, in time, it took sufficient hold for people to think that there must be some way of realising that dream. The problem was: how?

It's easy to see that, if one did solve the problem, one could call the result an 'expert system'. It would have the expertise previously found in human experts yet it could be switched off at night. But, the problem as to how this was to be done not only proved hard to solve - it has hardly been solved yet.

Ask anyone how a human expert works and, apart from comments like 'slowly', you'll find that nobody knows. Or, at least, they don't know to the extent that you could write a computer program to do it. Any explanation will be fuddled with words like 'judgement' and 'experience' which simply don't occur in the world of formal languages.

But the basic ideas are simple enough.

People, it's said, are general purpose thinking machines. Give them any problem and a bit of experience and they can use their judgement to think out a satisfactory solution to the problem. All that people consist of is a collection of brain cells, wired together somehow, and all that computers consist of is a collection of memory cells, also wired together somehow. So, write a program which will solve problems (in general) and you have a system which will replace people. With the added advantages of not breaking down, forgetting, going wrong, wanting to be paid, and so on. The problem is that this general purpose problem solver has been pretty elusive.

Less elusive have been specific problem solvers. For example, you might have a calculation to perform. Well, you could easily write a program to do that for you. In fact, if you've used a computer you already have done that. Give your machine an arithmetic expression and it can work out the answer for you. That sounds pretty obvious but in evaluating an arithmetic expression the computer has done everything which a human mathematician would have done. It hasn't just added a few numbers together, it's taken a string of symbols and manipulated them to put them into a sensible form. After that, any arithmetic would have been really trivial.

And if you should think that this is beside the point, consider a problem which your current computer can't immediately handle. Take the problem of integrating a mathematical expression. Integrating expressions can be notoriously difficult to do with some functions - so you might reasonably think that it would be a suitable topic for an expert system. But now suppose that you had written such a system. What would it actually look like?

It would look, probably, just like your current computer with its BASIC interpreter. The only difference would be that it had a few more expressions in the language. In fact, if you look around at some pocket calculators, you'll find that they've got 'integrate' keys which will carry out numerical integration for you on any function. So there's really nothing remarkable about that anymore.

And this serves to illustrate a point: once we know how to do something and can write a program for it, then it all ceases to appear at all remarkable. Only a short while ago, if you didn't happen to be a maths graduate yourself, you'd have had to call one in to perform integration. In other words, you'd have sent for an expert. With the calculators and programs available today that isn't necessary because the task has ceased (virtually) to be one which needs an expert. And, as a consequence, we'd hardly call the 'integrate' key on a pocket calculator an expert system.

To take a more everyday example, there once were people who were payroll experts. In fact, some of the most agile minds in the British Empire could have been found in the Army Pay Corps. Adept at finding loopholes, special cases, hitherto unknown allowances, these human experts (some would say 'superhuman') held sway for years. It was simply the payroll program which sent pay clerks the way of the dinosaur. The payroll program can easily be seen as an example of an expert system - it embodies the total sum of human expertise on the subject of payrolls. But nobody thinks of it like that anymore.

It's simply another case of a problem seeming to be trivial once someone has solved it.

And that's really the case with expert systems today. Previous advances are dismissed as scarcely being advances at all for the very simple reason that, like the early computer scientists, we're all sitting around wishing that we could get the computer to do something 'special'. Something more than it's doing already. But, if it did, we'd still be sitting around making noises of discontent because we'd still be looking for that something extra which, as yet, we have not got.

Each advance brings with it another computer program - no more and no less than that. But there just doesn't seem to be that one, final, Big Advance which brings the ultimate program. The one that finally breathes life into the monster.

And, just to make matters worse, if there were such a system, the next thing that would happen is that there would appear a new breed of experts. Human experts, they would be expert in the workings of the ultimate systems. They would know more about it than did anyone else - and they would be well-paid for their expertise. And, if you don't immediately believe that, ask yourself how it was that computer programmers ever appeared on the scene.

Suppose, for instance, that you devise an expert system which is able to carry out medical diagnosis. This has, of course, been done already. What do you think would happen next? Well, learned papers would appear in academic journals describing the system and saying what it could do. And others, noting what it couldn't do would set about building a better expert system. This, of course, has already happened. It is past history. You might sum it up by saying that: a computer program was written which was then criticised and improved by human experts who then wrote a better computer program. And so on. ...

Inevitably then, any expert system is only going to be a temporary palliative. Something to stave off the pangs of deprivation for a while. The trick is to find a good palliative rather than a bad one. Something that makes you think 'That's clever!' - even if you don't always continue to think it's clever. A computer program, in short, which will do something that you hadn't realised could be done by a computer at all. A computer program which does something which you'd have thought really needed the services of a human expert to achieve.

But still, in the end, a computer program.

1.4 What do you want your expert system to do?

Having defined an expert system as a computer program, no more, no less, we could just sit around content in the knowledge that we knew what an expert system was and we already had one. This is the easy way out.

The hard way out is to take the functional approach and ask: what do you want your expert system to do?

This is a dangerous question to ask because the answers can involve you in having to do some work, something which is generally considered distasteful. But that's computers for you. Generally speaking, people don't define computer programs except by virtue of what they do. So, if expert systems are computer programs then, what do they do?

At this point the onus goes over, temporarily, to you. Because the question is not "What does one, in general, want an expert system to do?" It's "What do you, in particular *you*, want your very own expert system to do?" After all, it's you who's going to build it. You might as well have some say in the matter if you can.

Well that question is, of course, easy. Settling into your armchair you switch on your computer, close your eyes and dream. The room swims before your eyes, a warm glow passes through your very being. A relaxed, confident smile plays on your lips as you idly key in the question: HOW CAN I BECOME A MILLIONAIRE? and a few simple, well-chosen phrases appear on the screen by way of answer.

"Of course!" you exclaim. "Yes, certainly. Yes. That would obviously make me a millionaire. If only I'd built my own expert system sooner."

And you make a note of the answer and proceed to interrogate that machine on the previously-vexed question of how you can stop your hair going grey, followed by a session on how to stop dandelions growing in your lawn.

It all works like a dream and, in fact, it is a dream.

Which is a bit of a pity, really. But Life's like that and no amount of programming is really going to help that much.

The problem is that some things are impossible to program, expert system or no expert system, and if you want to program an impossible thing then you will encounter difficulties of implementation.

Returning from dreams of great things to the real world, in which you're