SOCIAL SCIENCES: A THIRD LEVEL COURSE
COGNITIVE PSYCHOLOGY

D309

TMA 03 COGNITIVE MODELLING PROJECT

Prepared for the course team by: **Paul Mulholland**
Stuart Watt

Second Handover Draft, July 1, 1999

Contents

Project guide......1
1 Why do cognitive modelling?......3
2 Introducing Hank......8
3 Representing information in Hank......11
4 Filling in the gaps: using variables......17
5 Creating fact cards......21
6 Introducing instruction cards......26
7 Modelling schema theory in Hank......40
8 Extending the picnic schema......51
9 Current issues in cognitive modelling......56
References......61
Acknowledgements......62
Index of concepts......63
Answers to SAQs......64
Fact cards......*back cover*

Project guide

The purpose of this cognitive modelling project is to give you first-hand experience of some of the basic tools of cognitive modelling, and to show you how they can help shed light on problems of interest to cognitive psychologists. Although cognitive modelling borrows many techniques from artificial intelligence — and therefore draws heavily on concepts used in computer science — great care has been taken in this project to maintain a cognitive psychology perspective towards these concepts. Their relevance to cognitive processing is always emphasized, and no prior knowledge of computers is assumed. The project does not involve any mathematics at all — virtually the only numbers you will encounter are the page numbers in this text.

Objectives

By the time you have finished this booklet, you should be able to:

- define these terms and describe their relevance to models of cognitive processing:

  cognitive model, symbol, symbol structure, database, question, question processor, process, value, variable, specific value, default value
- build useful and valid cognitive models using all of these concepts
- explain the aspects of cognitive processing which are embodied in your models
- appreciate the importance of knowledge representation.

You should also have grasped these issues, which are fundamental to cognitive modelling:

- the interrelation of process and representation
- the role of facts and instructions
- the role of permanent and temporary stores
- how cognitive modelling is related to cognitive psychology.

How to approach this project

Cognitive modelling can only be learned by building cognitive models, so the material in this project will give you experience of building real models — and then working through them to see them work. All the modelling activities in this project are designed as paper and pencil exercises. However, you will have a chance at Summer School to try your models out by running them on a computer.

Suggested study timetable

In order to help you organize your time during the project, the table below shows a recommended study sequence, assuming you allocate ten evenings of study to carrying out this TMA. The table is intended to give you an indication of the amount of work involved in the various sections, although you, of course, may want to organize your workload on a different timetable. Notice that TMA 03 is divided into five distinct parts. These are described in the accompanying *Cognitive Modelling Assignment Booklet*.

Evening	***Activity***	***SAQs***	***TMA 03 part***
1	Project guide, Sections 1 and 2		
2	Sections 3 and 4	1 to 3	TMA 03-A and B
3	Section 5	4	TMA 03-C
4/5	Section 6	5	TMA 03-D
6/7	Sections 7 and 8	6 and 7	TMA 03-E
8	Miller article and section 9		" "
9/10	Finish writing report		" "

1 Why do cognitive modelling?

In the course of this project, you're going to be looking at a new, rather different, way of doing psychology, using **models**. So far, you've been working with theories, formulating hypotheses and then testing them, for the most part using experimental evidence. This is why the 'techniques' boxes in the course books usually have 'rationale', 'method', and 'results' sections. Modelling is a rather different approach to psychology, but, as we'll claim in this introduction, it is not a replacement for the experimental method; instead, it can be a useful addition to the cognitive psychologist's tool box.

But what is a cognitive model anyway?

First of all, you will find plenty of **cognitive models** in this course. They're practically everywhere! In language, there is Schank's model of scripts and actions (see *Language Understanding*, Part I, section 2.4 and Part II, section 4.1). In memory, there is Anderson's ACT* model of memory (see *Memory*, Part IIB). Collins and Quillian's model of conceptual categories is a good example (*Perception and Representation*, Part I, section 2.4). And in problem solving, Newell and Simon's model of means-ends analysis (*Problem Solving*, Part I, section 5.4) is another typical cognitive model. Finally, the 'neural network' approach of parallel distributed processing (or PDP), which you'll meet later in the course, is another common approach to using cognitive models. Above and beyond these, though, there are many more in other areas of psychology; in vision, analogy, emotions, and even consciousness. Some cognitive models are (like the examples we've just mentioned) so precise that they can even be turned into computer programs, and run as if they are typical experimental participants.

These examples will give you a bit of a hint that a model is very like a theory, but it is often a bit stronger, because models do more than predict what happens because of a given set of cognitive processes. They explain *how* and *why* those cognitive processes actually cause the behaviours that we observe. This is central to cognitive modelling — and central to cognitive psychology. We are not behaviourists, we emphasize the processes that are actually going on inside people. Models help us to emphasize these processes by giving us a precise way of describing them, which we can use to talk about them scientifically. We can study hidden mental processes inside people; by building models of these processes, we can show them to other psychologists, discuss them and test them, and check to see if they could explain human cognition in a useful way.

How do I build a cognitive model?

When you're putting together a cognitive model, the first thing you need is a theory. This isn't just a theory of what happens in some area of cognitive psychology, it is also an explanation of how it happens. It is this explanation of the **processes**, the mechanisms that underlie that bit of cognition, that makes the theory into something more like a model.

Let's look at Collins and Quillian's model of conceptual categories, described in *Perception and Representation*, Part I, section 2.4, as an example. For Collins and Quillian, there are predictions, which say that some statements take longer to verify than others, but there are also explanations that say that this happens because there is a search process through the levels of the concept hierarchy.

The process of building a model, then, is one of starting with a theory, and gradually refining it, by making its description more and more precise, until you have represented the theory as a clear and unambiguous model. You can use all sorts of evidence when you're refining your model: experimental evidence, other theories, and even introspection.[1] The problem of refining your model, then, is one of marshalling all this evidence, and putting it in a form that other people can look at and understand. This is the problem of representation.

Why is representation important?

Put very simply, **representations** are the link between what is going on inside someone's head — or a computer — and what is going on in the world outside. A representation in a model is an object (a word, for example) that is 'about' something in the world. Let's take the word 'blue', for example. This word is a **symbol** because it stands for, or represents, the colour we all know as blue. I can say 'the sky is blue' and we all know what that means, but there is nothing that is actually coloured blue in this sentence.

One of the most important points to remember, when building a cognitive model, is that all representations are not the same. For example, to us, the sentence 'le ciel est bleu' means the same as 'the sky is blue', only it is in French rather than English. Both sentences mean the same thing, but if we want to communicate this information to people who speak English, it matters which sentence we choose. Representations are just the same — if you want people to understand your model, you need to choose representations which will make sense to them. In this case, assuming we wanted to communicate to an English speaker, we would choose 'the sky is blue'. Similarly, if we wanted to show our model to psychologists, we would use words that make sense in psychology.

Actually, modelling *is* representation! All the difficulty in the modelling process is simply finding a good representation for the different bits of the model. But there are a few tricks that can help. The principle of 'cognitive economy' (*Perception and Representation*, Part I, section 1.1) is one example. Generally, it is a good idea to try to develop representations that don't involve saying things more than once. Another useful principle is consistency, being sure that you represent the same kind of things in the same way, and different things in different ways.

How do I evaluate a cognitive model?

So far, we've avoided one significant problem. How do you decide whether a model is an accurate description of what goes on inside people. This is not easy. Although there are a few rather informal ways of assessing a model, at the end of the day the only measure that really counts is: how well does it compare to real human behaviour?

Informally, a good way of 'eyeballing' a model is to look for the same kinds of things we've just mentioned as common in representations, economy and consistency. Is the model a lot more complicated than it needs to be to get the expected behaviour? Is it all a bit of a mess, with different things going on all over the shop? These problems don't necessarily make the model less valid, but they will make it a lot harder to explain to other people. Because

[1] Introspection might seem a bit surprising, but the problem with using introspection scientifically is that it is hard to make your introspective thoughts into something which other people can look at and discuss, all knowing that you're talking about the same things. Models are precise enough to make this possible.

models, like experiments, are the kind of thing you need to be able to write up and show to other people, clarity, economy, and consistency are useful and important. They also make it much easier to build the model in the first place!

There is really only one way to evaluate a cognitive model. Does the model make an accurate prediction of human cognitive behaviour? Typically, to make this kind of evaluation, you might 'run' the model (probably using a computer) as if it was a typical participant, and then you would compare the model's behaviour with that of human experimental participants. Do the two seem to show the same kinds of trends? — bearing in mind that literally comparing computers and people, say with a stopwatch, isn't really fair! If not, then, this also seems to throw doubt on the model, or, perhaps more importantly, on the theory it represents.

So why doesn't everybody do cognitive modelling?

Cognitive modelling is not perfect; it has both advantages and disadvantages. One of the most important advantages is that cognitive modelling can help you make sure that your theory is complete. Before you can build a computer model of a psychological theory, you need to be able to describe how every part of your theory works in a fair amount of detail. If you can't, if there are gaps in your understanding, this is itself pretty good evidence that your theory is incomplete. But even if you can build a complete model, it has to cross another hurdle. If it doesn't turn out to predict the behaviour of real people properly, that is also evidence that your theory may not be fully correct. Although this sounds rather negative, just like an experiment that doesn't work as you expect it to, if a model doesn't work there will be reasons why, and, therefore, useful lessons to be learned. But overall, if there is a good match between people's behaviour and the model's prediction, then there is some evidence that the model may be a good description of the psychological mechanisms underlying that behaviour.

This focus on behaviour is sometimes a problem — if we take behaviour alone as a measure of how good these models are, we have reduced cognitive modelling to a kind of behaviourism. This is really a myth: a cognitive model doesn't just look at what behaviour the theory generates, it looks at how the underlying cognitive structures generate that behaviour. In this sense, a cognitive model is very unlike a human experimental participant; it's very hard to look inside people's brains and find out what's going on. Rather like a working model of a car engine, a cognitive model makes it very easy to see what's happening inside, to make cognition go the way it does.

And what's modelling got to do with computers?

We often use **computers** to help us look at these behavioural parts of cognitive models; they allow you to take psychological models from your imagination, and to make them come alive. It is perfectly possible to build cognitive models without using a computer at all, and still to explore their behaviour when compared to people. But with a computer, you can even turn your models into 'typical participants', and try experiments on them, looking at the results, and comparing the results to make sure they're the same as you get with real people.

What computers do, and do supremely well, is to follow a set of **instructions** completely precisely and extremely quickly. If we turn our cognitive models into a set of instructions that a computer can follow, then we can get the computer to make our models come alive, without having to do it all ourselves. This has two advantages:

- Most cognitive models are very big, far too big and complicated for anybody to understand them in detail. When this happens, computers can take care of all the donkey-work and let us psychologists get on with the interesting stuff, looking at the cognition and behaviour that the model represents.

- It's very easy to assume that the model is both complete and correct, because we designed it that way. Other people, though, might see gaps or mistakes in our model. Computers are extremely unforgiving, and because they do only and exactly what they are told, they are very useful tools for checking models for gaps and mistakes.

In this project, we're not going to use a computer to test out our cognitive models — although you will get a chance to do that at the summer residential school. Instead, we're going to check our models by temporarily pretending to be computers, following a few simple rules to the letter, to make sure that our models work the way we intend them to.

But what about the 'computer metaphor'?

Some people take cognitive modelling a lot further, and link it to **artificial intelligence** (see the *Problem Solving* book, Part I, section 6.2). Artificial intelligence is not bound by modelling how people solve problems, it is really only interested in building computer programs that can solve problems fast and well. Having said that, artificial intelligence often uses ideas from cognitive psychology, because, basically, people are very good at solving problems fast and well.

But in the early days of cognitive psychology, in the 1950s and 1960s, artificial intelligence and cognitive psychology were linked a lot more closely, and many ideas flowed between the two disciplines. As a result, some ideas originally thought up in artificial intelligence have been very influential in cognitive psychology. Of these, one of the most important is the **computer metaphor**: quite simply, the idea that what minds do and what computers do is in some ways similar, so there is a correspondence between cognition and computation, and that people — like computers — are information processors.

The computer metaphor is nothing new, in that it simply succeeds a long line of models that have served as a kind of 'dominant metaphor' for psychological research. Before the computer came along, we used to think of the mind as like a clockwork system, then like a hydraulic system, then like a telephone system, and only then like a computer system (see *Memory*, Part III, section 1). Indeed, today there are *two* metaphors, because along with the computer metaphor there is the connectionist metaphor of parallel distributed processing, which suggests that we can think of the mind as like a lot of interconnected small and simple cells. As a metaphor, the computer just happens to be quite useful, but it is not necessarily a central part of theories in cognitive psychology.

A second, and perhaps more interesting connection between psychology and computers is that computers were actually modelled after people in the first place. In the 1940s and 1950s, electronic computers were designed to do some of the repetitive mathematical tasks which used to be done by people — people called 'computers'. Today, we have simply dropped the 'electronic'.

Despite the fact that computers were modelled after people, they have made surprisingly important contributions to some psychological theories already. For example, all psychologists today have no hesitation talking about 'memory' of one kind or another. But during the time of the behaviourists, you couldn't use the word 'memory' and get away with it (Turkle, 1988). Computers changed all that. Miller (1956) could talk about memory because he had borrowed the term from electronic computers — the engineers who had built these computers had no such qualms about using mentalistic terms like these. So it was computers that indirectly made the term ‘memory’ legitimate again for psychology.

Summary of section 1

- Cognitive modelling is another tool to help cognitive psychologists.
- Many existing theories in cognitive psychology have their roots in cognitive models.
- Modelling and experimentation are complementary, not alternatives.
- Cognitive modelling doesn’t depend on computers, but they can be a big help when testing large models.
- Simple and consistent representations make it much easier to build a model.

2 Introducing Hank

For this project, we'll be using a modelling language called Hank, which is specially designed for representing and building cognitive models. In fact, to use Hank you won't even need a computer — Hank has been designed so that you can build your models and work through them on pieces of paper.

Hank, like any other cognitive modelling language, focuses on how information is represented and processed. In order to achieve this, Hank has two main components: a place where information can be stored and represented, called the **database**, and a **question processor** which works on and processes this information, using it to answer questions. The question processor has a temporary 'working out' area, called the **workspace**, and works by following a set of rules called the **house rules**. The purpose of this chapter is to introduce you to these components.

The database is a permanent, all-purpose, storage area. The contents of the database are stable; it can be likened to the notion of long-term memory in Part IIA of *Memory*. Generally speaking, its contents do not change much over time. The contents of the workspace, on the other hand, degrade over time. It can be likened to the notion of working memory, in Part IIA of *Memory*. The question processor uses the information stored in the database and the storage area of the workspace to answer incoming questions and supply an answer. To answer any question, the question processor follows a standard set of rules called the house rules.

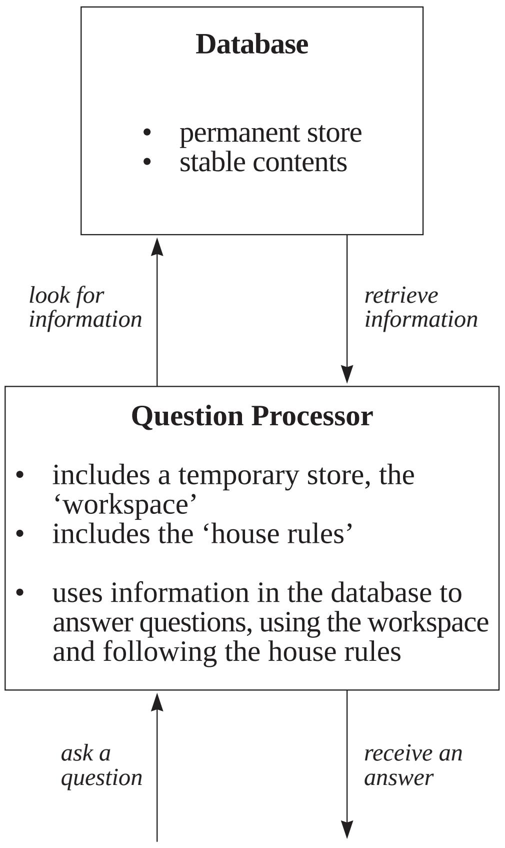

Database
• permanent store
• stable contents
look for information
retrieve information
Question Processor
• includes a temporary store, the 'workspace'
• includes the 'house rules'
• uses information in the database to answer questions, using the workspace and following the house rules
ask a question
receive an answer

Figure 1: An overview of the architecture of Hank

An overview of the architecture of Hank is shown in Figure 1. Let's consider briefly how the architecture works. Hank is spurred into action by sending a question to the question processor. The question processor then stores the question in the workspace. The question is kept in the workspace while an answer is sought from information stored in the database. This is important. It would be a waste of time starting a process to answer the question if the question was lost before the process had been completed. The question processor then looks in the database for information which can help answer the question. If appropriate information is found, then this information is copied to the workspace. The information and question are compared to see if it is sufficient to answer the question. If it is, then a reply is given to the original question. In many cases, the information that is brought back leads to more questions being asked. This will require further use of the database to retrieve more information. If the information needed to answer the question cannot be found in the database, then the question processor will say that the question cannot be answered. A more detailed account of the way the question processor works will unfold as you work through the project.

We will now look in more detail at the main components of Hank, starting with the database. The Hank database stores information in the form of 'cards'. There are two kinds of card within Hank: fact cards and instruction cards. Fact cards are used to store the basic pieces of information we know as facts. An example fact could be 'Paris is the capital city of France'. Fact cards store information that we do not have to work out. It is information that we simply 'know as a fact'. Below is an example of a fact card. This is shown here simply to give you an idea of what they look like. Fact cards will be explained in the next section.

Likes	
Does like	Is liked
John	Pavarotti
Susan	Bob Dylan
Jenny	Peter
Peter	Jenny

The other kind of cards in the database — instruction cards — store the definitions of processes. These can be thought of as being a bit like recipes. They provide a description of *how* some particular process should be carried out. For example, if someone asked you to multiply 64 by 3, you wouldn't know the answer as a fact, but you would be able to work it out. Instruction cards represent processes like the one you might use to do this working out.

A cognitive model's database contains a set of fact cards and instruction cards. These represent the knowledge and the processes that together make up the model. When you put a question to the question processor, it uses the information in the database to find — or to work out — an answer. This is what you might call 'running' the model.

The question processor is the other main part of the Hank architecture. You can think of the question processor as the 'worker' within the architecture. It is responsible for running any process, and for getting and using the information from the database. The question processor's behaviour is dictated by the house rules which give a complete account of how it works. Put simply, the house rules tell the question processor how to use the fact cards and the instruction cards to answer questions. A fuller understanding of the behaviour of the question processor, how it does what it does, will unfold as you work through the project.

Inside the question processor, as well as the house rules, there is the workspace. This is a temporary area where information can be stored while a process is being carried out. You can think of the workspace as like the 'workings out' that you might use while doing a difficult multiplication, or planning an essay. In Hank, the workspace is actually used to store any questions — and any answers — that are needed along the way while you're working out the answer to a main question. You'll see these intermediate questions in section 6.

So in this section we have considered the Hank architecture in general. This architecture applies both when you are using Hank at home for this TMA, and when using Hank at summer school. There is one big difference though. For your TMA you will be working with databases written only on paper, without the help of a computer. When running your model, *you* will take on the role of the question processor, following the house rules to find the answer. This will give you an understanding of the inner workings of a cognitive model, which you will find useful when you go to summer school, where a computer will take on the role of the question processor.

Summary of section 2

- The Hank architecture is made up of a database and a question processor.
- The database is the permanent information store.
- The information is stored using two kinds of cards: fact cards and instruction cards.
- Inside the question processor are the workspace and the house rules.
- The workspace is a temporary store for working out information used in answering a question.
- The house rules tell the question processor how to use the fact cards and instruction cards in the database to answer questions.

3 Representing information in Hank

In this section we will look at how information is represented in the Hank modelling language in the form of fact cards. We will then show how this information can be accessed using questions.

3.1 Structuring symbols

In section 2, we outlined the overall architecture of Hank, comprising:

- a database, and,
- a question processor.

We also said that the database is made up of two kinds of card: fact cards and instruction cards. We are now going to start to look at how information is organized or structured on these cards, starting with the fact cards. The guidelines as to how information is structured within a cognitive modelling language determine what is referred to as the **symbol structure**. Let’s consider a familiar example of a symbol structure: the telephone book.

Adams, R.	49 Low Street	52287
Briggs, B.	27 Bay Street	35635
Hawkins, S.	4 Green Street	67456
James, W.	65 Leaf Lane	73463
Lewis, K.	12 High Street	21341
Peters, F.	2 Flower Lane	85676

The telephone book is a good example of a symbol structure, containing three kinds of symbol: names, addresses, and telephone numbers. (Of course, each of these three kinds of symbol is itself made up of further symbols, letters and numbers, but we won’t worry about that now.) To make use of a telephone book, we need to understand how the symbols have been structured. First, we need to know what each column represents. The first column is a list of names, the second is a list of addresses and the third is a list of telephone numbers. Next, we need to know how the name, address and telephone number making up each row relate to each other. For example, the first row tells us that R. Adams lives at 49 Low Street and his or her telephone number is 52287. This style of symbol structure, where each column represents a different type of symbol, and each row indicates a relationship between a set of symbols, is very common. We see it not only in the telephone book, but also in bus and train timetables, shopping receipts, contents pages of books, and so on. As you will see, this familiar symbol structure underpins how we represent information in Hank fact cards.

3.2 Introducing fact cards

Fact cards are what we use in Hank to represent the things we know. Each column represents a particular type of information (e.g. a name or an address), and each row represents a connection between related pieces of information (e.g. the address and telephone number of a named person). Unlike the telephone book example above, fact cards have extra labels to indicate what the symbols mean. Each card is given a **card name**. This appears on the first row and has a dark grey background. The second row, shown in light grey, gives a label for each individual column. These are called **column labels**. The remaining rows each represent a related set of symbols. These are called **data rows**. Each fact card has one or more data rows. Fact card 1 has two data rows. In English the fact card could be read as 'triangles have 3 sides; rectangles have 4 sides'.

Fact card 1

Sides		
Name	Number	*Card name* / *Column labels*
Triangle	3	*Data rows*
Rectangle	4	

The format for fact cards is always the same (i.e. one row for the name of the card, one row for labelling the columns, and then any number of data rows). Fact card 2 has only one column, but the structure is exactly the same. A column label may seem a bit unnecessary here, but it helps to reinforce consistency between different fact cards. In English, fact card 2 would read as, 'triangles, rectangles, and pentagons are shapes'.

Fact card 2

Shape
Example
Triangle
Rectangle
Pentagon

A fact card can have any number of columns. Fact card 3 has three columns, and in English would read as, 'Peter gave an apple to Joan. Jenny gave a beer to Bill. Anna gave money to the RSPCA'.

Fact card 3

Who gave what to who		
From	To	Object
Peter	Joan	Apple
Jenny	Bill	Beer
Anna	RSPCA	Money

The piece of data found in each cell of the data rows is called a value. A **value** is a permanent piece of information, representing something we know about the world. From fact card 1, we *know* that a triangle has three sides. We do not need to work it out. We know it as a fact. Values can be contrasted with variables which are used to access stored values, and can have different values at different times. We will meet variables in section 4.

You might be wondering what kinds of things can be used as card names, column labels and values in data rows. The answer is — nearly anything. They can be any sequence of numbers, letters, spaces and punctuation marks. The only restriction is that they must not start and end with a question mark. At this stage, this may seem like an odd restriction, but things that start and end with a question mark are the variables you will meet later. So, all of the following could be used as card names, column labels or values:

John
5454.234
£1000
Mary had a little lamb!
(*^*^(@(£)@£@£*@£@&(^%&$!@£

The fact that you can use almost anything, should not encourage you to do so. When deciding what to use, there is another issue to consider which is far more important than what Hank will accept. The names, labels and values you use must be meaningful to you and your tutor.

SAQ 1
Represent the telephone book extract shown in section 3.1 in the form of a Hank fact card.

3.3 Asking questions

We can retrieve the information stored in fact cards by posing a **question** which relates to a row within one of our fact cards. For the rest of this chapter, we will be using the three fact cards from section 3.2. For easy reference, these fact cards are also shown on the back cover of this booklet. There is a noticeable physical difference between a fact card and a question. A question is drawn with dashed lines and has no shading. This is to prevent the question being mistaken for a fact card. Although questions have a different appearance, the symbol structure is very similar to the fact cards, except that they always have only three rows. The first row is the name. The second row contains the column labels. The third row is a data row. Questions only have one data row, as you can only ask one question at a time. Whenever we ask a question, Hank gives us a reply, which is either ‘Status = OK’ or ‘Status = Fail’. ‘Status = OK’ is Hank’s special terminology for ‘yes I found it’, and ‘Status = Fail’ means ‘no I did not find it’.

For example, if we asked the following question, which in English would read as ‘Does a triangle have three sides?’:

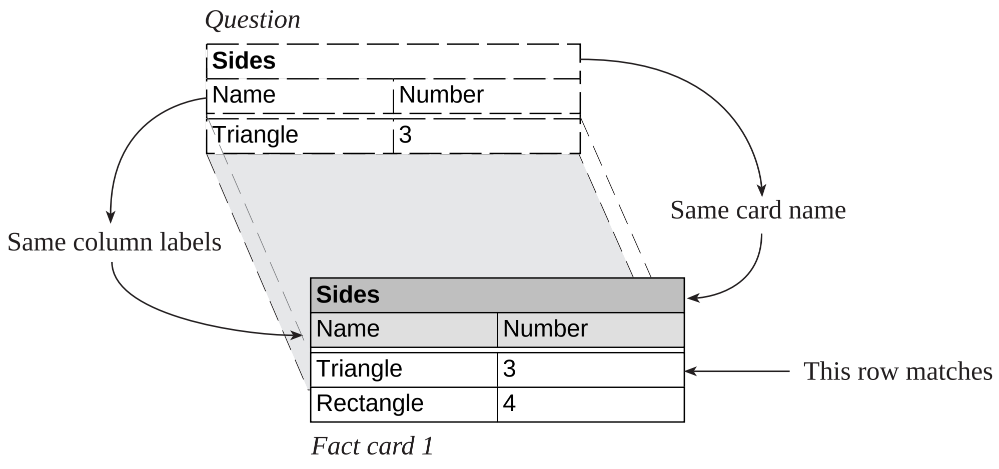
Sides	
Name	Number
Triangle	3

the reply would be:

```
Status = OK
```

The question **matches** against the Sides fact card, fact card 1 on the back cover. The process of matching a question to a fact card has three steps. The question has to match the card name, then the column labels, and finally a data row. In this case, the question name matches the card name of fact card 1. The column labels of the question match the column labels of the fact card. The data row of the question matches the first data row of the fact card.

The following diagram shows the matching process used in answering this question:



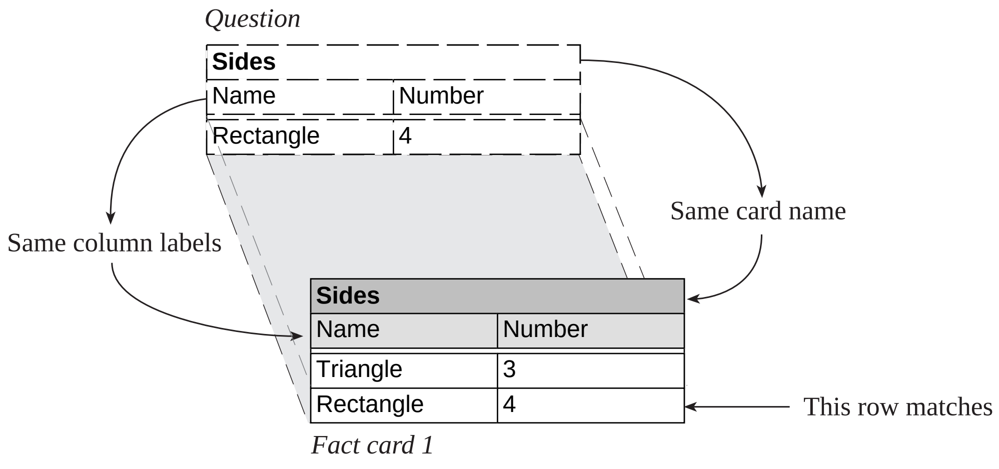
Similarly, we could ask whether a rectangle had four sides. We would draw the question as follows:

Sides	
Name	Number
Rectangle	4

Once again the reply would be:

```
Status = OK
```

The question has the same card name and column labels as one of the fact cards in the database. This time, the data row of the question matches the second data row of the fact card. When matching the data row of the question, the data rows in the fact card are tried in turn, from top to bottom. The matching process for this question is illustrated below.



In Hank, and in all cognitive modelling languages, there is nothing like common sense. So, for example, if we had used the word ‘Oblong’ rather than ‘Rectangle’, we humans might assume these to be the same, but Hank would not. In Hank, two things only match if they look the same. It is not enough for them to mean the same. Hank only matches symbols, it doesn’t know what those symbols mean. So, for example, ‘Rectangle’ and ‘Rectangle’ are the same, but ‘Rectangle’ and ‘Oblong’ are just as different to Hank as ‘Rectangle’ and ‘Aardvark’, or ‘Rectangle’ and ‘R#%*5%!$e’.

This principle is an important one. If you want to refer to the same thing in different places, you must always use the same name. It does not matter which names you choose, but you do need to be consistent. It is also a good idea to stick to names which make sense, so that other people — particularly and especially your tutor — can understand your model.

A question is answered by looking for a data row in a fact card which *matches* the question. For the answer to be found, not only should the card name and column labels be the same, but all the entries in that row should match the row in the question. So, for example, ‘Triangle’ will match ‘triangle’ (in Hank, upper and lower case letters are treated as being the same), and ‘3’ will match ‘3’. On the other hand, ‘3’ will *not* match ‘three’. In Hank, symbols are matched but there is no understanding that ‘3’ and ‘three’, for example, mean the same thing. So, if we asked (in English: ‘Is a triangle a shape?’):

Shape
Example
TRIANGLE

Compared against the Shape fact card, fact card 2 on the back cover, the reply would be:

```
Status = OK
```

as ‘TRIANGLE’ matches with ‘Triangle’ in the fact card.

On the other hand, if we asked (in English: ‘Is a square a shape?’):

Shape
Example
Square

this time, the answer would be:

```
Status = Fail
```

because although *we* know that a square is a shape, it is not included in fact card 2.
Also, if we asked:

Object
Example
Triangle

the answer would be:

```
Status = Fail
```

The name ‘Object’ cannot match with the name ‘Shape’.

SAQ 2
Draw the questions you would put to Hank for ‘Does a rectangle have 5 sides?’ and ‘Did Jenny give Bill a beer?’. What would Hank answer to each of these questions, taking into account fact cards 1 and 3.

The question processor, following some simple house rules, carries out the process of matching a question against a fact card. These house rules are summarized below to give a concise description of how Hank answers questions.

House rules for matching a question against a fact card
1 Ask the question Look to see if there is a match with a fact card in the database. This involves matching the name of the card, the column labels, and a data row. The data rows are tried in order from top to bottom.
2 Give the reply If a match was found then the reply (i.e. the Status) is OK, otherwise the reply is Fail.

The first step for the question processor is to try to match the question against a fact card. The second step determines whether the reply should be OK or Fail, depending on whether a match was found.

Summary of section 3

- Information is structured in the form of fact cards.
- Fact cards are made up of a card name, column labels and data rows.
- Fact cards contain values, which are permanent pieces of information.
- Information can be accessed using a question.
- Questions are answered by the question processor.

TMA 03-A
At this point complete TMA 03-A.

4 Filling in the gaps: using variables

By now, you have probably realized that the types of answer that can be given to a question are quite limited. So far, the only answers we can have are ‘OK’ and ‘Fail’. So, if we wanted to find out how many sides a rectangle had, we would have to ask questions starting with ‘Does a rectangle have one side?’, then two, and so on until we get the answer ‘OK’. Not only is this boring, it is unrealistic and psychologically implausible. Fact card 1, the Sides fact card, contains a row that says exactly how many sides a rectangle has. What we need is a way for the question processor to use a particular data row to find the answer, and reply with that answer.

We can do this by using **variables** in our questions. In Hank, a variable says that a particular cell in a data row can be filled in with any value during the work of the question processor. So, if you use a variable in a question, the question processor will initially look for a match as before. For the fact card to match it has to have the same name and the same column labels. The question processor then looks down the data rows of the fact card to find a match. If a matching data row is found then the variable takes on the value given in the matching fact card and the Status is OK. The question processor then reports the Status and the values taken on by any variables. If a match is not found then the status is reported as being Fail. If the Status is Fail, then the variables do not need to be reported as they did not take on any values. To make variables distinctive, they begin and end with a question mark.

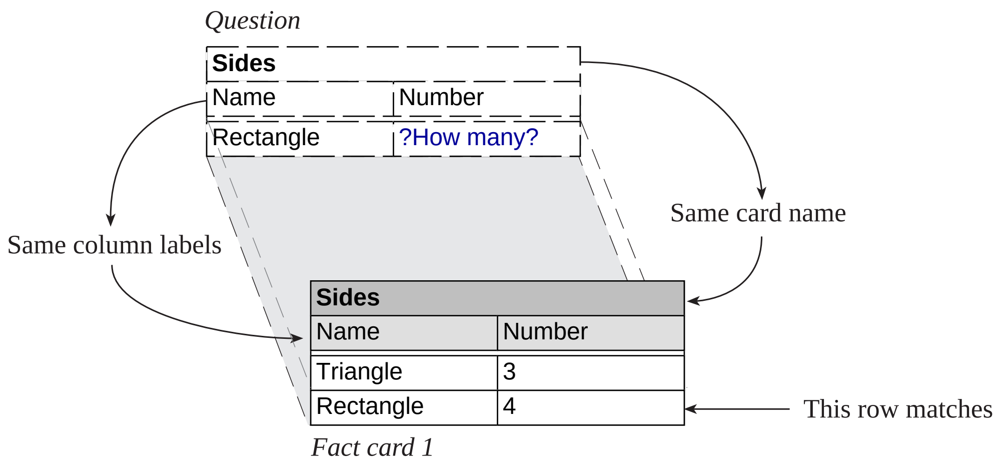
For example, given the Sides fact card, if we asked the question (in English: ‘How many sides does a rectangle have?’):

Sides	
Name	Number
Rectangle	?How many?

the answer would be:

Status = OK
?How many? = 4

The question matches the second data row of the Sides fact card. The Rectangle value in the question matches with the same word in the fact card. The variable ?How many? takes on the value 4. This matching process is shown below.



Alternatively, we could ask the question (in English: ‘What is the name of the shape with 4 sides?’):

Sides	
Name	Number
?What?	4

This time, the answer would be:

Status = OK
?What? = Rectangle

You can even use more than one variable in the same question. For example, we could ask the question (in English: ‘What did Jenny give, and to who?’):

Who gave what to who		
From	To	Object
Jenny	?Who?	?What?

In this case, from fact card 3, the answer would be:

Status = OK
?Who? = Bill
?What? = Beer

An important point to note about variables is that each variable can only contain one value. For example, if we asked the question (in English ‘Which person gave beer, and who did they give it to?’):

Who gave what to who		
From	To	Object
?Which?	?Who?	Beer

the answer would be:

Status = OK
?Which? = Jenny
?Who? = Bill

But if we asked the question:

Who gave what to who		
From	To	Object
?Who?	?Who?	Beer

the answer would be:

Status = Fail

This question would only work if someone was listed in fact card 3 as having given beer to themselves. This is because the same variable is appearing in both the From and To columns of the question. As a particular variable can only take on one value at a time, the question cannot match with any of the data rows in fact card 3, as all the data rows have different people in the From and To columns.

You may be wondering what happens if the question can match more than one data row in a fact card. For example, consider the following question (in English 'Tell me the name of a shape'):

Shape
Example
?Name?

This question could match any of the three data rows in fact card 2. In this situation, the question processor just picks up the first matching data row found, so the answer would be:

Status = OK
?Name? = Triangle

For some cognitive models it is necessary to get all the possible answers that match a question, though this is not required for any of the models you will build for this TMA. Hank has extra features that allow all matching data rows to be accessed. We will introduce you to these features if you carry out a cognitive modelling project at summer school.

An item in the data row of a question is taken to be a variable if it begins and ends with a question mark. All of the following would be treated as variables:

?Object?
?This is a variable?
?2468?
?£?

As always, the important point to remember is to use meaningful names for variables. It is also important to note that within a question only the bottom row can contain variables. You cannot use variables in the name or column label rows.

SAQ 3
Which of the following are legal questions? Note that you should work out which ones are legal, not which ones match against one of the fact cards.

(a)

Shape
?Column?
Rectangle

(b)

?What was it called?		
From	To	Object
Jenny	Bill	Beer

(c)

Shape
Name
Triangle

(d)

Who gave what to who		
From	To	Object
?Giver?	?Taker?	?Thing?

The processing of questions with variables is similar to that of questions only containing values. First, the question has to match against a fact card. This means that the card name, column labels, and a data row must match. Second, the question processor must keep track of the values that have been taken on by the variables. Third, the status is reported back, and if a match was found then the values of variables should also be reported. This is summarized in the extended version of the house rules below.

Extended house rules for matching a question against a fact card

1 Ask the question

Look to see if there is a match with a fact card in the database. This involves matching the name of the card, the column labels, and one of the data rows. The data rows are tried in order from top to bottom.

2 Match the variables

If a match was found and the question contained one or more variables, make a note of the values that they have taken on.

3 Give the reply

If a match was found, then the reply (i.e. the Status) is OK. The answer should also state any variables that have matched with values.

If a match was not found then the reply is Fail, and the variables need not be reported.

Summary of section 4

- Variables start and end with a question mark.
- They can be used to take on a value from a fact card.
- If a matching data row is found, the question processor reports the values of variables as well as the status.

TMA 03-B
At this point complete TMA 03-B.

5 Creating fact cards

When designing a cognitive model, you have to consider carefully the issue known as **knowledge representation**. A cognitive model contains a representation of some knowledge that we have in our heads. For example, if we built a cognitive model to simulate the way people solve the Towers of Hanoi problem (described in Part I of *Problem Solving*) we would need to represent knowledge of the rings and pegs, and the rules of how the rings can be moved around. Similarly, if we wanted to build a model of how people analyse syntactic structures (described in Part I of *Language Understanding*), we would have to work out a way of representing the syntactic structure of sentences. The theories described in psychological texts tell us *what* we need to model (e.g. rings, pegs, game rules, words, sentence structures) but they do not tell us precisely *how* we should model them in Hank. We have to work this out for ourselves. This section will raise important points to be considered when choosing a knowledge representation.

Basically, Hank represents facts by classifying objects into categories and by defining relationships between objects. In section 3 we saw the Shape fact card. This fact card classifies a set of objects (i.e. triangle, rectangle and pentagon) as being shapes. Fact cards comprising more than one column, not only classify objects, but also indicate relationships between them. For example, the Sides fact card expresses a relationship between some shapes and some numbers. Triangle is related to the number three as this is the number of sides it has. Rectangle is related to the number 4 in the same way. Hank can be used to represent any statement that indicates a classification of objects or a relationship between them. Consider carefully the statements shown below.

‘The number seven bus goes to Bletchley’

‘Pavarotti sings opera’

‘John likes Pavarotti’

‘The number seven bus departs from Wolverton’

‘Susan rather likes Bob Dylan’

‘Tammy’s hobbies are tennis and cycling’

‘Bob Dylan sings folk music’

We will now show how these statements can be represented in Hank, and try to draw out some of the issues that you need to think about when deciding how to construct fact cards. Let’s start by looking at the first statement:

‘The number seven bus goes to Bletchley’

This statement makes a relationship between the number seven bus and Bletchley. Bletchley is related to the number seven bus by being its destination. If we wanted to construct a fact card, we could give it a name such as ‘Bus destination’ and we would need two columns. One column would be for the bus number and the other column would be for the destination. We would then need a data row containing the number seven and Bletchley. Before drawing the card there is a final issue we need to consider:

Should we use ‘seven’ or ‘7’?

This is an issue of knowledge representation. The fact card could contain either of these symbols, so which one should we use? The answer is that whatever we put in the fact card has to match what we want to appear in answers to questions. If we use the number '7' then if we ask which bus goes to Bletchley we will get the reply '7'. If we use the word 'seven' in the fact card then the answer will contain the word 'seven'. Hank does not dictate which should be used — it is up to you as the modeller. Here we are going to use the number rather than the word, because buses usually use numbers rather than names. We can therefore draw the fact card like this:

Bus destination	
Number	Destination
7	Bletchley

Now let's move on to the second statement which is:

'Pavarotti sings opera'

When deciding how to turn this into a fact card, we once again have to resolve a question relating to how this knowledge should be represented:

Should my fact card represent opera singers or singers in general?

We have a choice here. We could make a fact card called something like 'Sings opera' and just have a single column to record Pavarotti and any other opera singers we knew. In this case the fact card would be used to list a set of people as being opera singers. Alternatively, we could make a fact card called something like 'Sings' and use two columns, one column for the person's name, and the other column to record the type of music they sing. In this case we are representing the information by thinking of singing as being the relationship between Pavarotti and opera. These two alternatives are shown below.

Sings opera
Example
Pavarotti

Sings	
Name	Type
Pavarotti	opera

Once again, the way you represent the information depends on the kinds of questions you wish to ask, and the range of information you wish to present in a single card. Now suppose we wanted to add that Bob Dylan sings folk music (as we will later on). If we choose the two-column version which relates names to the type of music they sing, then Bob Dylan and Pavarotti could both be placed in the same fact card. If we choose the more specific 'Sings opera' fact card, we would have to use a separate folk singers fact card for Bob Dylan. For this reason there is an argument that the more general 'Sings' fact card is preferable as it can be used to represent a larger amount of information, and means that we don't have to ask different kinds of question about different kinds of singer.

Now we can move on to the third of our statements:

'John likes Pavarotti'

Based on the previous discussion, this one seems quite straightforward. Here we can think of 'likes' as being the relationship between John and Pavarotti. In this relationship, John does the liking, and Pavarotti is one who is liked. This fact card is drawn below.

Likes	
Does like	Is liked
John	Pavarotti

Even in this seemingly simple example we have an issue of knowledge representation to consider:

What does ‘John’ represent?

John could mean a particular person you know called John, and indicate that this particular John likes Pavarotti. Alternatively, it could mean everyone called John. In this case the fact would be telling us that everyone called John likes Pavarotti. We could be more specific by writing ‘John Green’ rather than ‘John’ to indicate which John we mean. But, even if we did this, the issue would not go away. It could still be read to mean that everyone called John Green likes Pavarotti. The important point to realize is that, however carefully you design your fact cards, there is often some ambiguity as to what they actually mean. The only real solution is that you as the modeller should be clear about what you mean, and perhaps make a note next to your fact cards to try to say precisely what they mean. In this example, we will keep the fact card with just the word John, but be aware that someone else using our fact cards could be unclear about precisely what it means, and may need clarification.

We can now move on to the fourth statement, which states:

‘The number seven bus departs from Wolverton’

The information contained in this statement is related to the information in our first statement, which gave the destination of the number seven bus. We therefore have another knowledge representation issue to consider. Here, instead of making a new ‘Bus departure’ fact card, we could add an extra column to the ‘Bus destination’ fact card called something like ‘Departs from’ and rename the card ‘Bus information’. Once again there is no right answer to this question. Other modellers may prefer to have separate fact cards for recording departures and destinations. This would not be a wrong decision. It is up to the modeller, and depends on the kinds of question we wish to ask. In this case we will extend our existing fact card to make a ‘Bus information’ fact card, to keep all the bus information in a single fact card. Our new fact card is shown below.

Bus information		
Number	Departure	Destination
7	Wolverton	Bletchley

We can now move on to the next statement:

‘Susan rather likes Bob Dylan’

With this statement, we can see that ‘rather likes’ is the relationship between Susan and Bob Dylan. Here we have another knowledge representation issue to consider:

Are ‘likes’ and ‘rather likes’ the same thing?

We have already developed a ‘likes’ fact card for expressing the relationship between John and Pavarotti. If ‘likes’ and ‘rather likes’ are the same thing, then we could just add an extra data row to that fact card for Susan and Bob Dylan. The answer to this problem is again up to us as modellers. If we were interested in small variations in the extent to which one person may like another person, then we would probably construct a separate ‘Rather likes’ fact card. Let’s assume these small nuances are not of interest to us and simply add Susan and Bob Dylan to our existing ‘Likes’ fact card:

Likes	
Does like	Is liked
John	Pavarotti
Susan	Bob Dylan

We can now move on to the next statement, which is:

‘Tammy’s hobbies are tennis and cycling’

The issue we need to consider here is whether we should use one or two data rows to represent Tammy’s hobbies. The two possibilities are shown below.

People and their hobbies		
Name	Hobby 1	Hobby 2
Tammy	Tennis	Cycling

People and their hobbies	
Name	Hobby
Tammy	Tennis
Tammy	Cycling

With regard to this knowledge representation issue, we can be fairly certain that the fact card with two columns is preferable to the one with three columns. The fact card with three columns is not a very good representation for a number of reasons. First, this fact card can only be used to store information about people with two hobbies. We would need separate fact cards for people with different numbers of hobbies. Second, it would be more difficult to ask questions about hobbies. For example if we wanted to ask a question to find out if Tammy played tennis we would need to know which of the two hobby columns contained tennis. For these reasons, the two-column version is preferable.

We now have just one more statement to consider.

‘Bob Dylan sings folk music’

We decided earlier that we could add this as an extra data row to our existing ‘Sings’ fact card. Another small issue we have to consider is whether we should enter Bob Dylan’s type of music as ‘Folk’ or ‘Folk music’. In this situation it is probably preferable to enter ‘Folk’ rather than ‘Folk music’. If someone is singing something of a certain type, we can usually expect it to be a type of music, therefore the word ‘music’ seems unnecessary. Also, if we entered ‘Folk music’ rather than ‘Folk’ we would have to remember to add the word music each time we asked a question about folk, otherwise it would fail to match. For these reasons we will design our fact card as below.

Sings	
Name	Type
Pavarotti	Opera
Bob Dylan	Folk

This process of creating fact cards has illustrated the problem of knowledge representation. We often have quite a few decisions to make when designing our fact cards, and there is rarely a clear answer. When designing fact cards it is important to think carefully about what kind of information you wish to represent in that single card, and what kind of questions you will want to ask.

SAQ 4
Construct one or more fact cards to represent the following statements.

‘Ben’s car is an Escort’
‘Mary’s car is red’
‘Joan’s car is a blue Mondeo’

Summary of section 5

- Fact cards represent relationships between objects.
- Careful thought should be given to knowledge representation when constructing fact cards.
- Knowledge representation determines what each fact card contains and what kinds of questions can be asked.

TMA 03-C
At this point complete TMA 03-C.

6 Introducing instruction cards

So far, we have seen how we can represent factual knowledge — things that you either know or you don't know. As you have probably guessed by now, this is quite limiting. For example, if you think of problem solving, it is very unusual to know the complete solution to a problem (if you did it would not be much of a problem!). It is far more common to have to work out a solution, using one strategy or another. For this reason, Hank provides **instruction cards**. An instruction card simply represents a process that can be used to work something out. In this section we will first give an example of an instruction card and show how it works. We will then show how an instruction card can be used to link fact cards together, and how an instruction card can be used to ask several questions of the same fact card. Finally, we will give a few reasons explaining why instruction cards are so useful and so important.

6.1 Constructing an instruction card

We will construct our first instruction card using the Who fact card, fact card 3 on the back cover, shown again below.

Fact card 3

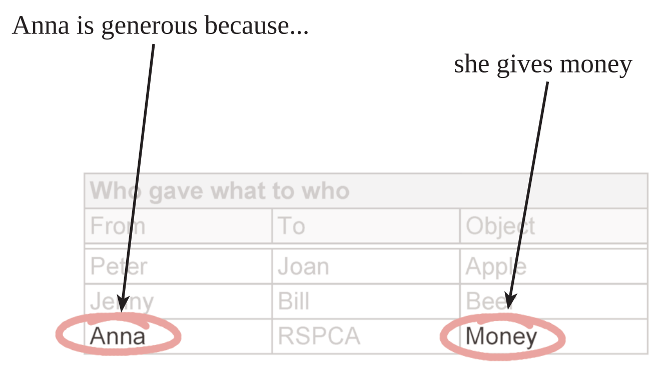
Who gave what to who		
From	To	Object
Peter	Joan	Apple
Jenny	Bill	Beer
Anna	RSPCA	Money

As we mentioned earlier, fact cards are used to represent facts about the world. Instruction cards can be used to derive extra information from the existing fact cards. We will start by looking at an example. Let's say we wanted to express the following statement in Hank.

'Someone is generous if they give money'

Information about who gives what is represented in fact card 3. The first column of the fact card refers to the giver, the second column to the recipient, and the third column to the object that is given. The statement above refers specifically to people who give money. Therefore when finding out who is generous we are only interested in people if the object that they gave is money; that is, 'Money' must appear in the Object column. The third data row of the fact card has 'Money' in the Object column. The person who is generous is the person named in the From column for that row. In this case, Anna is named as a person who gave money. The statement says nothing about to who or what the money is given, therefore we do not care about what is in the To column.

This can be represented graphically.



Who gave what to who		
From	To	Object
Peter	Joan	Apple
Jenny	Bill	Beer
Anna	RSPCA	Money

One approach to representing this statement in Hank would be find out who fits the definition of being generous (i.e. Anna) and then to make a new fact card to represent this information. The new fact card might look like this:

Generous
Name
Anna

If we now asked the question (in English ‘Who is generous?’):

Generous
Name
?Who?

we would get the answer:

```
Status = OK
?Who? = Anna
```

There is a problem with this approach. For example, if a new data row was added to the Who fact card, fact card 3, to express that Peter gave money to Susan, we would have to update the Generous fact card to include Peter. If this was not done, the information would be inconsistent. Even if we did always remember to update the Generous fact card, we still have a much greater problem. The Generous fact card does not provide a good representation of the information contained in the sentence: ‘Someone is generous if they give money’. This is the knowledge representation issue once again. If the sentence had been ‘Anna is generous’ then the fact card would be a good representation, but the sentence we are trying to represent does not explicitly mention Anna. Rather than telling us *who* is generous, the sentence describes the *process* of how we can find out whether someone is generous. Information that describes a process rather than a fact is represented using an instruction card in Hank.

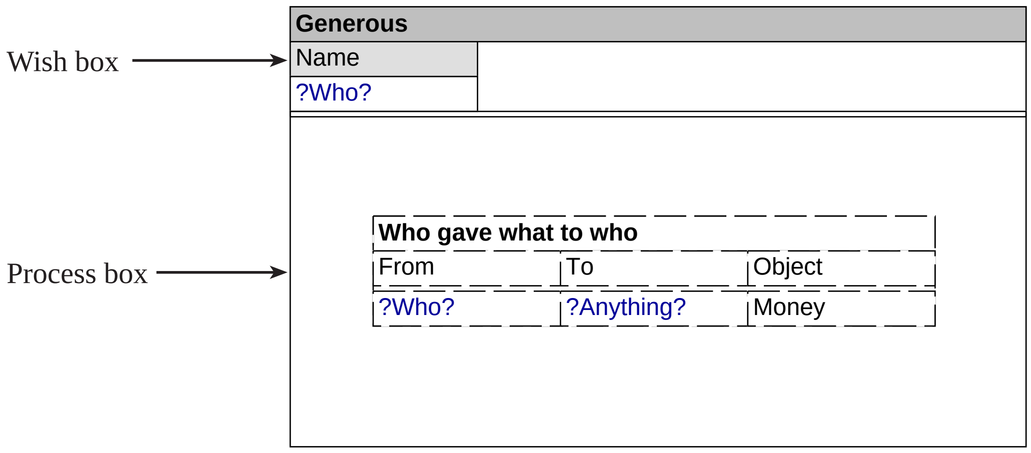
An instruction card is divided into two parts, a wish box and a process box. The **wish box** describes the kind of question we want the instruction card to be able to answer. The **process box** describes how Hank goes about answering it. For this example we want the instruction card to find out who is generous. The wish box will therefore be the same shape as the question we constructed above. This wish box, therefore, looks like this:

Generous
Name
?Who?

Now instead of constructing a fact card containing answers to this question, we are going to describe the process by which this question can be answered. The process itself is defined as asking one or more questions. The process of finding out whether someone is generous involves asking one question against fact card 3. The question in English is ‘Who gives money’, and is shown below.

Who gave what to who		
From	To	Object
?Who?	?Anything?	Money

The name that matches with the ?Who? variable will fit our definition of being a generous person. We can now put the wish box and the process box together to form an instruction card that works out whether someone is generous:



There are some important things to notice about this instruction card. First, notice that the same variable name (i.e. ?Who?) appears in the wish box and in the From slot in the question contained in the process box. This means that those two slots must contain the same thing. The person who is described as generous is the person who the money came from. A second thing to note is that the ?Anything? variable appears in the process box, but does not appear in the wish box. This means that although information about who receives the money is described in the fact card, it does not need to appear in the answer. We are only interested in who gives the money, not in the recipient. Even though we don’t care about what goes in the To slot we cannot just leave it blank, because then it would only match blanks in the fact card. It must have a variable name placed in it. In Hank, boxes should not be left empty. The final thing to notice is that the Object slot contains a value rather than a variable. This is because we are only interested in the data rows of fact card 3 that have Money in the Object slot.

Now we have the Generous instruction card added to the database we can ask the question (in English ‘Who is generous’):

Generous
Name
?Who?

and the reply will be:

Status = OK
?Who? = Anna

In answering this question, the question processor first matches the question against the wish box of the instruction card. This means that the ?Who? variable in the question matches against the ?Who? variable in the wish box. Now that it knows this is the right instruction card to use, the question processor then asks the question that is contained in the process box. This question matches against fact card 3. The ?Who? variable takes on the value Anna, and the ?Anything? variable takes on the value RSPCA. Once the question has been asked, the process is complete and the question that matched against the wish box can be answered.

You may be wondering what would happen if we asked the question with a different variable name from the one that appears in the instruction card. If we asked the question as follows, with the variable ?Person? rather than ?Who?:

Generous
Name
?Person?

the reply would be:

Status = OK
?Person? = Anna

The instruction card would work as before. This time the variable ?Person? in the question would match the variable ?Who? in the wish box. Essentially, we would be using two variable names for the same thing. ?Person? would be the variable name used in the question, and ?Who? would be the variable name used inside the instruction card, but they would be names for the same thing. Someone who can speak both French and English, will know that 'chat' is the French word for a cat. When in France you will use the word 'chat', and when in England you will use the word 'cat', but you will be referring to the same thing. Similarly, in the example above ?Person? is the variable name used in the question and ?Who? is the variable name used in the instruction card, but both variable names refer to the same thing.

If we asked the question (in English 'Is Peter generous?'):

Generous
Name
Peter

the reply would be:

Status = Fail

When answering this question, the question processor would start by matching the question against the wish box. In this case, rather than matching two variables together as above, the ?Who? variable takes on the value Peter. This means that Peter now replaces ?Who? wherever it appeared in the instruction card. The question processor now asks the question inside the process box, and ?Who? has been replaced with Peter. The question therefore becomes (in English 'Did Peter give money to anything?'):

Who gave what to who		
From	To	Object
Peter	?Anything?	Money

The answer to this question is Fail, as the question does not match any of the data rows in fact card 3. As the question in the process box gets the answer Fail, the instruction card as a whole also gets the answer Fail.

6.2 Using an instruction card to link fact cards

We will now show how an instruction card can be used to link fact cards together. In this example we will use the Likes and Sings fact cards constructed in section 5. The Likes fact card has had two more data rows added to indicate that Jenny likes Peter, and Peter likes Jenny. The fact cards are shown below, and also on the back cover.

Fact card 4

Likes	
Does like	Is liked
John	Pavarotti
Susan	Bob Dylan
Jenny	Peter
Peter	Jenny

Fact card 5

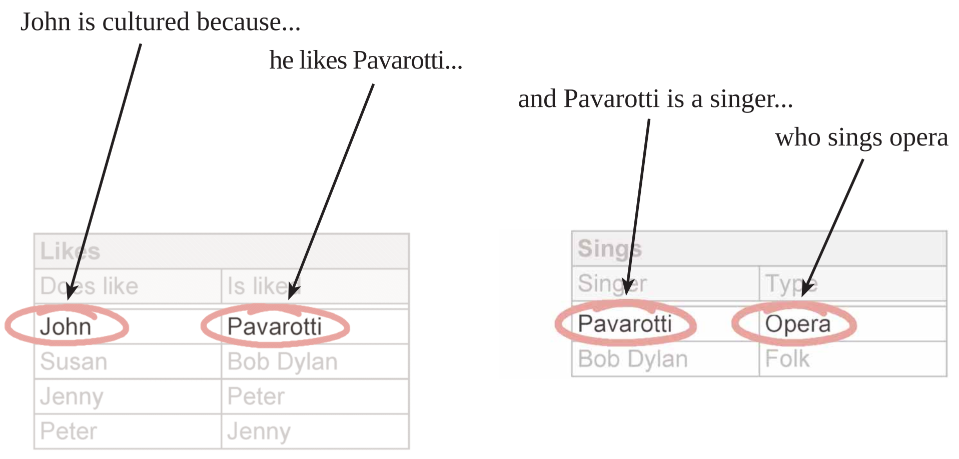
Sings	
Name	Type
Pavarotti	Opera
Bob Dylan	Folk

We will now try to represent the following statement in Hank:

‘A person is cultured if they like someone who sings opera’

Similar to the previous example, this statement does not tell us who is cultured, but it describes a process by which we can find out whether someone is cultured. Again, because this is a process, we need to represent this information using an instruction card rather than a fact card. The process of finding out whether someone is cultured has two parts. First, we have to find out who the person likes. This type of information is represented in fact card 4. Second, we have to find out whether the person they like sings opera. This type of information is represented in fact card 5. Let’s look at this in more detail.

From the Likes fact card we need to find a data row that expresses that a person (column 1) likes someone (column 2). That same someone must also be represented in a data row of the Sings fact card. The someone must appear as a singer (column 1) who sings the type of music known as opera (column 2). We can see from the Likes and Sings fact cards that John would fit our definition of being cultured. He is listed in the Likes fact card as liking Pavarotti, and Pavarotti is listed in the Sings fact card as someone who sings opera. We can show this graphically:



The picture above shows that in order to find out whether someone is cultured, we will need to ask two separate questions, one question against the Likes fact card and one question against the Sings fact card.

We can now begin to construct the instruction card, starting with the wish box. The wish box describes the kind of question we wish the instruction card to answer. In this case we want the instruction card to find out who is cultured. We can therefore write the wish box as follows:

Cultured
Name
?Who?

The next step is to decide what goes in the process box. This time the process box needs to contain two questions rather than just one. The first question we need to ask is who the person likes. This question can be written as follows:

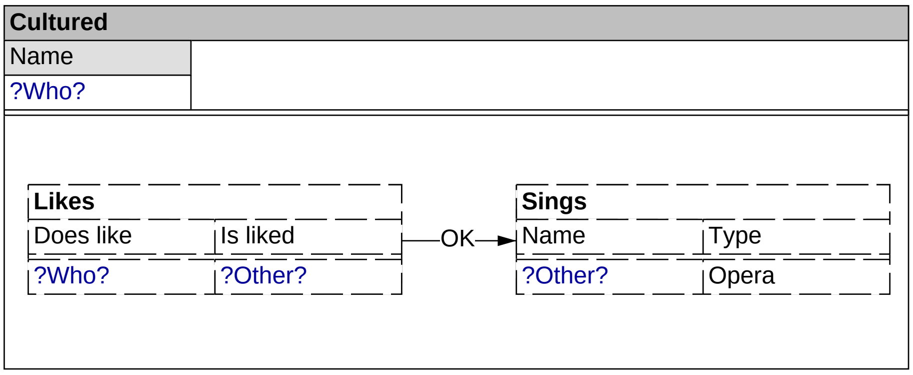
Likes	
Does like	Is liked
?Who?	?Other?

Note that the ?Who? variable, as well as appearing in the wish box, also appears in the first column of the Likes question. This is because the person defined as cultured is the same person who does the liking, therefore the same variable name appears in both positions. The second question has to determine whether the person who is liked also sings opera. This question can be written as follows:

Sings	
Name	Type
?Other?	Opera

Note that the Name column in this second question contains the ?Other? variable, the same variable that is used in the Is liked column of the Likes question. This is because the person who is liked is the person who must be the opera singer. The Type column contains the value Opera rather than a variable because we are only interested in opera singers.

Now we have worked out what goes in the wish box and the process box, we can construct the instruction card, which is shown below. The only extra feature which you have not met before is the **OK arrow** that joins the two questions together. The OK arrow means that if a successful match can be found for the Likes question (i.e. if the Status is OK for that question) then follow the OK arrow to the Sings question.



Now that we have this instruction card, we can ask the following question (in English 'Is John cultured?'):

Cultured
Name
John

and we would get the reply:

Status = OK

In dealing with this question, the question processor will first match the question against the wish box of the instruction card. This has the effect of the variable ?Who? taking on the value John. This means that John now replaces ?Who? wherever it appeared in the instruction card. The question processor now moves on to try the first question in the process box. As John has matched with ?Who? the question will be (in English 'Who does John like?'):

Likes	
Does like	Is liked
John	?Other?

This question matches the first data row of fact card 4. Matching the data row means that the variable ?Other? now takes on the value Pavarotti. As the question matched against the fact card we can follow the OK arrow to the next question. The variable ?Other? has taken on the value Pavarotti, so the next question will be (in English 'Does Pavarotti sing opera?'):

Sings	
Name	Type
Pavarotti	Opera

This question matches the first data row of the Sings fact card, fact card 5. As the matching process was successful the Status of OK can be given as the reply. The whole instruction card, then, has connected the facts to come to the conclusion that John is indeed cultured.

On the other hand, if we asked the question (in English 'Is Susan cultured?'):

Cultured
Name
Susan

the question processor would match the question against the wish box of the instruction card, so the variable ?Who? in the instruction card would take on the value Susan. This will apply to wherever ?Who? appears in the instruction card. The question processor now moves on to the first question in the process box. As the ?Who? variable has taken on the value Susan, the question will be (in English ‘Who does Susan like?’):

Likes	
Does like	Is liked
Susan	?Other?

This question matches the second data row of fact card 4, so the variable ?Other? takes on the value Bob Dylan. As a match was found for this question, the question processor moves along the OK arrow to the next question. As the variable ?Other? has taken on the value Bob Dylan, the question will be (in English ‘Does Bob Dylan sing opera?’):

Sings	
Name	Type
Bob Dylan	Opera

As this does not match any of the data rows in fact card 5, the reply is Fail. Because a part of the process fails, the whole instruction card fails, and the reply for the whole question is:

Status = Fail

Similarly, if we asked the question (in English ‘Is Clive cultured?’):

Cultured
Name
Clive

the question processor will match the question against the wish box, and the ?Who? variable will take on the value Clive. The first question in the process box will therefore be (in English ‘Who does Clive like?’):

Likes	
Does like	Is liked
Clive	?Other?

As this does not match any of the data rows in fact card 4, the answer is Fail. Because this step of the process fails, there is no need to follow the OK arrow, and the reply for the whole question is:

Status = Fail

SAQ 5

(a) Develop an instruction card that defines the following (read the statement carefully):

‘A person is sociable if they like someone who sings folk’

The wish box of the instruction card should match with the following question:

Sociable
Name
Susan

(b) Develop an instruction card that defines the following (read the statement carefully):

‘A person is happy if they sing folk and someone likes them’

The wish box of the instruction card should match with the following question:

Happy
Name
Bob Dylan

6.3 Using an instruction card to ask multiple questions of the same fact card

In section 6.2 we saw how an instruction card can be used to link two fact cards together, by asking two questions and using variables to make links between the information in the wish box and the two questions. In the cultured example above, the ?Other? variable was used in both questions to show that the person who is liked is also the person who sings. The ?Who? variable was used in the wish box and in the first question to show that the person who does the liking is the person who is cultured. We will now see how an instruction card can be used to make links between information held in different data rows of the same fact card, as well as between different fact cards. Consider the statement below in relation to the Likes fact card, fact card 4.

‘Two people are friends if they like each other’

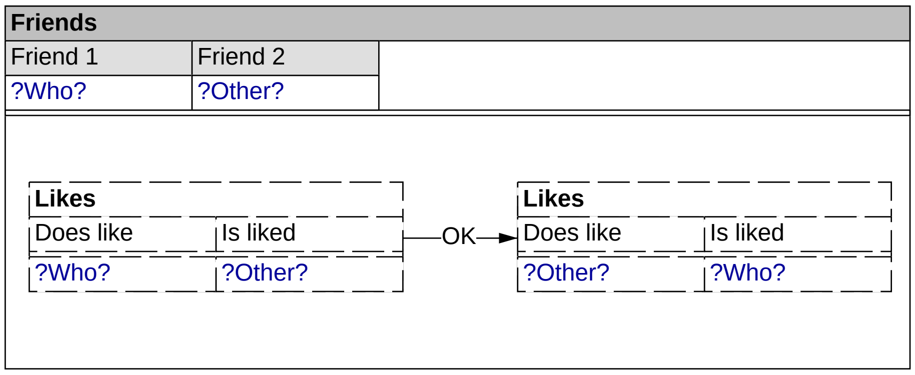
We can look at fact card 4 to see who might fit this definition of being friends. For two people to be friends, the Likes relationship has to be in both directions. According to the fact card, Jenny and Peter are friends, because Jenny likes Peter and Peter likes Jenny. However, John and Pavarotti do not fit our definition of being friends, because although John likes Pavarotti, we do not have a data row to say that Pavarotti likes John.

We will now construct an instruction card that can work out whether two people are friends. We will start by constructing the wish box. This time the wish box needs to contain two columns, one for each of the two friends. We need to use two different variable names, as otherwise we could only find out about someone who was friends with themselves. The wish box is shown below.

Friends

Friend 1	Friend 2
?Who?	?Other?

The process box will need to contain two questions, to check whether the ?Who? person likes the ?Other? person, and to check whether the ?Other? person likes the ?Who? person. As we have two questions in the process box, we join them using an OK arrow. The instruction card is shown below.



If we now asked the question (in English ‘Are Jenny and Peter friends?’):

Friends

Friend 1	Friend 2
Jenny	Peter

the question processor would first match the question against the wish box of the instruction card. This results in ?Who? taking on the value Jenny, and ?Other? taking on the value Peter. The question processor then moves on to the first question in the process box. As the variables have taken on values, the question will now be (in English ‘Does Jenny like Peter?’):

Likes

Does like	Is liked
Jenny	Peter

This matches against the third data row of fact card 4. As the question matched a data row in the fact card, the question processor moves along the OK arrow to the next question. The next question will be (in English ‘Does Peter like Jenny?’):

Likes

Does like	Is liked
Peter	Jenny

This question matches the fourth data row of fact card 4. This means that Jenny and Peter are friends and the reply is:

Status = OK

If we asked the question (in English ‘Are Susan and Bob Dylan friends?’):

Friends	
Friend 1	Friend 2
Susan	Bob Dylan

the question processor would first match the question against the wish box, meaning ?Who? takes on the value Susan, and ?Other? takes on the value Bob Dylan. The question processor then moves on to the first question in the process box, which will be (in English 'Does Susan like Bob Dylan?'):

Likes	
Does like	Is liked
Susan	Bob Dylan

As this matches the second data row of fact card 4, the question processor follows the OK arrow to the next question. The next question will be (in English 'Does Bob Dylan like Susan?'):

Likes	
Does like	Is liked
Bob Dylan	Susan

As this does not match any of the data rows in the Likes fact card, the reply for this question, and the Friends question as a whole is:

Status = Fail

In other words, although Susan likes Bob Dylan, they are not friends, because Bob Dylan doesn't like Susan, at least according to the knowledge in the Likes fact card.

6.4 Instruction cards and their relation to house rules

You may be wondering how the work carried out by the question processor when using an instruction card, relates to the house rules described at the end of sections 3 and 4. Basically, the process of working through an instruction card has three steps:

1 Matching the initial question to the wish box.

2 Repeatedly using the house rules shown at the end of section 4, for each question encountered in the process box.

3 Giving the answer to the initial question.

The process of answering a question directed at an instruction card is supported by the question processor's temporary memory, called the workspace. As you may have gathered from the examples above, processing an instruction card places certain demands on memory:

- Remembering the initial question to which an answer is required.
- Remembering the answer for each intermediate question encountered in the process box.
- Remembering the values of variables and passing them from question to question.

All this information is held in the workspace while the question processor looks for an answer to the initial question. For this assignment, you are working without the help of a computer, and will have to do this job for yourselves. Although the question processor's workspace is a temporary store, unlike human working memory it has a relatively large capacity, so you may have difficulty holding all the information in your head when taking on the role of the question processor. For this reason, we would advise you when working out an answer for one of your own instruction cards, to make a note of:

1 The initial question.
2 Each question in the process box encountered along the way.
3 The value of each variable.

When you've written an instruction card, keeping a written record of how it would be processed should make you feel more confident that it will work in the way you intend.

6.5 Why use instruction cards?

Using an instruction card rather than making another fact card has a number of advantages:

Psychological plausibility
Reaction time experiments (such as the experiment by Collins and Quillian in Part I of *Perception and Representation*) show us that accessing different kinds of knowledge takes different amounts of time, the implication being that their processing requirements are different. Using instruction cards to represent knowledge that can be derived from existing fact cards means that fewer fact cards need to be held in the database, and also means that accessing certain kinds of knowledge may have greater processing demands. This is because the information is derived through a process described in an instruction card, rather than being retrieved directly from a fact card. Collins and Quillian term this phenomenon 'cognitive economy'.

Internal consistency
Having a fact card listing all the cultured people could cause updating problems if we added more information about who likes who. Every time an extra data row was added to either the Likes or Sings fact cards, a check would have to be made to find out whether the Cultured fact card also needed to be changed. If this check was not carried out, then the data held in different fact cards could become inconsistent or even contradictory.

Abstraction level
Using an instruction card allows us not only to find out who is cultured, but also allows us to represent *why*. A fact card would not tell us why John is cultured. An instruction card indicates why someone is cultured (i.e. because they like someone who sings opera).

Summary of section 6

- Instruction cards describe a process by which something can be found out.
- Instruction cards are made up of a wish box and a process box.
- The wish box describes what questions the instruction card can be used to answer.
- The process box describes how the answer is found.
- The process is defined as one or more questions.
- An OK arrow is used to join questions together in the process box.
- Instruction cards can be used to derive extra information from an existing fact card.
- They can also be used to make a link between different fact cards, or between different data rows in the same fact card.

TMA 03-D
At this point complete TMA 03-D.

In case you were wondering…

A final point about instruction cards is illustrated if we consider the following question (in English ‘Who is friends with who?’):

Friends	
Friend 1	Friend 2
?Who?	?Other?

Although we know that Jenny and Peter are friends, the answer to this question would be:

Status = Fail

When answering this question, the question processor would first match the question to the wish box. This matching process does not cause either of the variables in the instruction card to take on a value. The question processor now tries to answer the first question in the process box. As neither of the variables have taken on values the question will be (in English ‘Name two people who like each other’):

Likes	
Does like	Is liked
?Who?	?Other?

This will match against the first data row of the Likes fact card, fact card 4, meaning ?Who? takes on the value John, and ?Other? takes on the value Pavarotti. As a match was found, the question processor follows the OK arrow to the next question. As the variables have taken on values, the question will be (in English ‘Does Pavarotti like John?’):

Likes	
Does like	Is liked
Pavarotti	John

This does not match any of the data rows, so the reply will be:

Status = Fail

If fact card 4 was reordered so that one of the data rows describing Peter and Jenny was placed first, then the question processor would be able to find them as being friends from a question that only contained variables. In order for the question processor to find an answer regardless of the order in which data rows appeared in a fact card, we would need a facility in Hank so that the question processor could try each data row in turn until it found an answer. Hank does have this facility and you will meet it at summer school. This issue does not need to be considered for this TMA, and the point is only mentioned here in case this problem had occurred to you.

7 Modelling schema theory in Hank

In this section we will use fact cards and instruction cards to build a model of schema theory. The Hank model will be developed to represent three important aspects of schema theory: category membership, default values and specific values. Before continuing, you might find it useful to re-read section 3 of Part 1 of the *Memory* book.

7.1 Representing category membership

A central notion of schema theory is that new information is interpreted, structured and understood in terms of what we already know. Cognitive theories based on schema theory include Schank's theory of conceptual dependency (*Language* book), which offers an explanation of how we understand sentences. Also, there are Schank and Abelson's scripts (*Language* book) which present a theory of how common events are understood as a stereotypical sequence of actions. An important property of schema theory is that some object or event is classified with similar events having similar properties. This allows us to infer missing details. For example, if we hear that someone visited a restaurant we assume that certain events happened; sitting at a table, reading a menu, eating a meal, and paying the bill. Similarly, if we hear that someone has attended a picnic we will assume other information such as; sandwiches were eaten, the weather was nice, and that the picnic happened outside. According to schema theory, we carry out this process of inference by comparing the new event to similar events we already know, which have certain stereotypical properties.

Let's suppose that we wanted to represent the statement below in Hank, in order that the program will know the location of picnics by default.

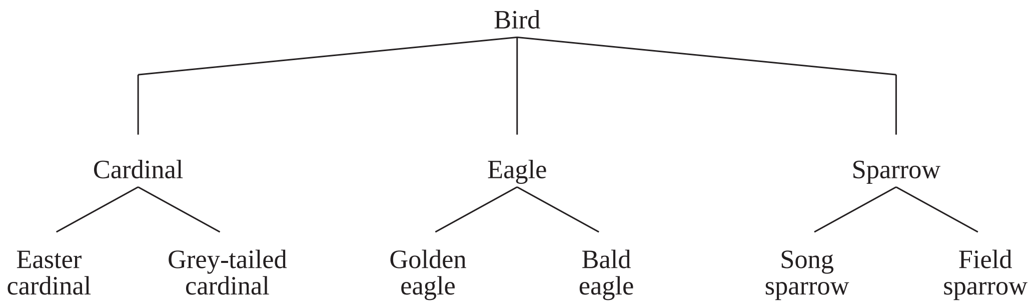
'All picnics are held outside'

We can represent this in Hank in a number of ways. One possible way would be to construct a new fact card, called possibly 'Location' and list in the first column all the picnics we know about, and in the second column the place where they are held. For all of them the location will be 'Outside'. This fact card is shown below.

Location	
Picnic	Place
Jane's birthday	Outside
Bob's picnic	Outside
Works outing	Outside
Sue's party	Outside

This fact card would give us correct answers to questions about the location of picnics, but once again we have a knowledge representation problem. The statement 'All picnics are held outside' does not refer to any specific picnics. It is about picnics in general. This representation works within limits, but is not consistent with schema theory. According to schema theory, common properties of objects can be inferred from their **category membership**. The fact card above links specific picnics directly to the location being 'Outside'. However, according to schema theory, a particular event should inherit the default value by being a member of a category (i.e. on the basis of category membership). In other words, we can infer that Bob's picnic is held outside, simply because it is a picnic — that is, a member of the category 'picnic'. Therefore, we need to start our schema model by having a representation of category membership, for the picnic category. Before constructing category membership for picnics we will look at the concept of category membership in general, and how it can be represented in Hank.

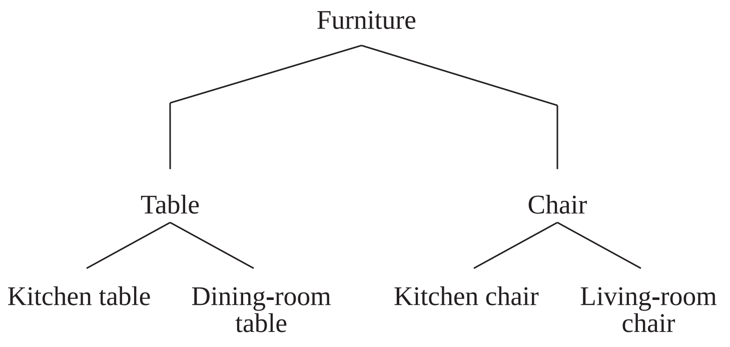
Category membership is a commonly used concept in cognitive psychology. It is introduced in Part I of the *Perception and Representation* book and can be seen in Rosch's hierarchy of birds. The diagram is shown below. Here the nodes represent different categories of bird, and each link represents a relationship between two bird categories. There is an implicit 'Is a kind of' relationship between the linked categories, with the lower of any two connected nodes being a subcategory of the higher node. For example 'Cardinal' is a subcategory of 'Bird', and 'Grey-tailed cardinal' is a subcategory of 'Cardinal'.



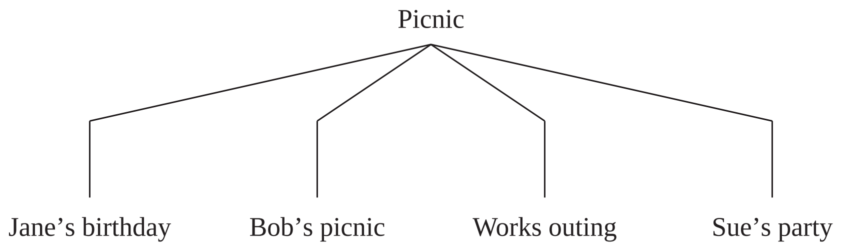
To turn this into a fact card in Hank, we use the name of the relation ('Is a kind of') as the name of the fact card, and turn each link into a separate data row in the fact card, remembering to be consistent about the direction of category membership. The words 'Subcategory' and 'Category' can be used as column labels. The fact card we construct would look like this:

Is a kind of	
Subcategory	Category
Cardinal	Bird
Eagle	Bird
Sparrow	Bird
Easter cardinal	Cardinal
Grey-tailed cardinal	Cardinal
Golden eagle	Eagle
Bald eagle	Eagle
Song sparrow	Sparrow
Field sparrow	Sparrow

SAQ 6
Represent the hierarchy below in the form of a fact card.



Our four picnics (i.e. Jane's birthday, Bob's picnic, Works outing and Sue's party) being members of the picnic category can also be represented in the form of a hierarchy. This is shown below.



Using the same approach as above, we can construct a fact card to represent the category membership of the four picnics. This is shown in fact card 6 below. Our four picnics being members of the picnic category can be thought of as being a part of a larger hierarchy. For example, we could think of the picnic category itself being a subcategory of a 'Social events' category, but for our work in this TMA, we only need to use this small snapshot.

Fact card 6

Is a kind of	
Subcategory	Category
Jane's birthday	Picnic
Bob's picnic	Picnic
Works outing	Picnic
Sue's party	Picnic

This is a much better starting point than the Location card we considered (and rejected) earlier, because it explicitly represents that each of these events is a kind of picnic. We can now be consistent with schema theory, and represent that all picnics are held outside, and that this applies to all the members of the picnic category (i.e. Jane's birthday, Bob's picnic, Works outing and Sue's party). The information about the picnic being outside does not link to each individual picnic, it applies to picnics in general.

7.2 Representing default values

So far we have represented category membership in schema theory. We will now move on to represent **default values** about picnics. Three default values about picnics are written as statements below:

‘All picnics are held outside’

‘At picnics you eat sandwiches’

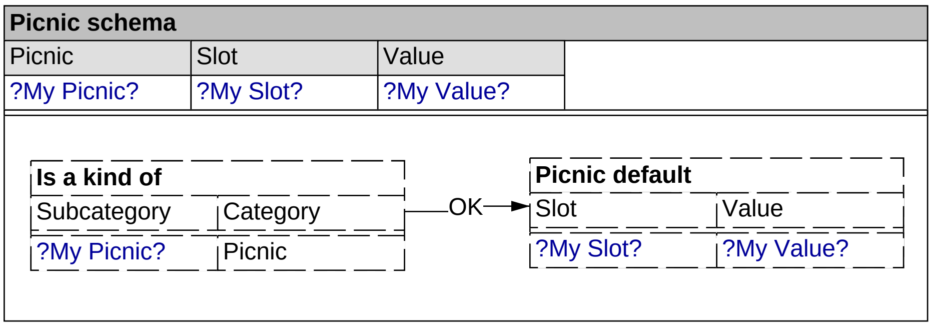
‘The weather is sunny at picnics’

When turning these statements into Hank, we need to appreciate the notion of slots and values in schema theory. Important slots for a picnic schema are location, food and weather. The default values that match these slots are outside, sandwiches and sunny respectively. Our next step, then, is to construct another fact card that relates each slot to its respective default value. This is shown below.

Fact card 7

Picnic default	
Slot	Value
Location	Outside
Food	Sandwiches
Weather	Sunny

To emphasize how this fact card relates to schema theory, we have used the words ‘Slot’ and ‘Value’ as column labels. We have the two fact cards needed to show how particular picnics can inherit default values. To make it work we now need an instruction card. When inheriting a default value, such as the location of a specific picnic, the instruction card will have to perform two tasks. The first task is to make sure that the event is a member of the picnic category. The second task is to find the relevant default value of the picnic category. For example, to find out where Jane’s birthday is held, the first task is to check that Jane’s birthday is a member of the picnic category. The second task is to find out where picnics are normally held. Therefore, our instruction card for inheriting the default location for a picnic contains two questions. The first question asks if the event is a member of the picnic category. This will be an ‘Is a kind of’ question as the information is stored in the ‘Is a kind of’ fact card. The second question finds the default location for picnics. This will be a ‘Picnic default’ question as the information is stored in the ‘Picnic default’ fact card. The completed instruction card is shown below:



If we now ask the question (in English ‘Where was Jane’s birthday held?’):

Picnic schema		
Picnic	Slot	Value
Jane's birthday	Location	?My Value?

we will get the answer:

Status = OK
?My Value? = Outside

The question processor answers this question in the following way. First, the question is matched against the wish box of the instruction card. This results in the variable ?My Picnic? taking on the value 'Jane's birthday', and the variable ?My Slot? taking on the value 'Location'. The question processor then moves to the first question in the process box. As the ?My Picnic? variable has taken on a value, the question will be (in English 'Is Jane's birthday a picnic?'):

Is a kind of	
Subcategory	Category
Jane's birthday	Picnic

This question matches the first data row of fact card 6. This means that 'Jane's birthday' is represented as a member of the picnic category. The question processor then follows the OK arrow to the next question. As the ?My Slot? variable has taken on a value, the question will be (in English 'Where are picnics located?'):

Picnic default	
Slot	Value
Location	?My Value?

This question matches the first data row of fact card 7, resulting in the ?My Value? variable taking on the value 'Outside'. As the questions in the process box have successfully matched against fact cards, the reply will be:

Status = OK
?My Value? = Outside

Jane's birthday therefore inherits the default location of being outside, because Jane's birthday is a member of the picnic category (represented in the 'Is a kind of' fact card) and outside is the default location for picnics (represented in the 'Picnic default' fact card).

So far we have seen how Hank can model the inheritance of default values, but there are occasions where a default value may need to be overridden. A good example is an ostrich, which is a bird and inherits bird properties such as having wings, a beak and feathers, but does not inherit the property of being able to fly. In the next section we will extend the picnic schema model to deal with specific values, as well as default values.

7.3 Representing specific values

We will illustrate how Hank can deal with **specific values** by continuing with the picnic schema, but moving on to another property of picnics. The 'Picnic default' fact card also contains the information that sandwiches are eaten at picnics. This holds for three of the four picnics in our example, but not for Bob's picnic. Hot dogs are eaten at Bob's picnic. We cannot add this information into the 'Picnic default' fact card as this is information about one specific picnic, not about picnics in general. In schema theory, this is an example of a specific value. We need to store these specific values in a new fact card which we will call 'Picnic specific' (fact card 8 on the back cover has an extra data row, which we will add later in this section).

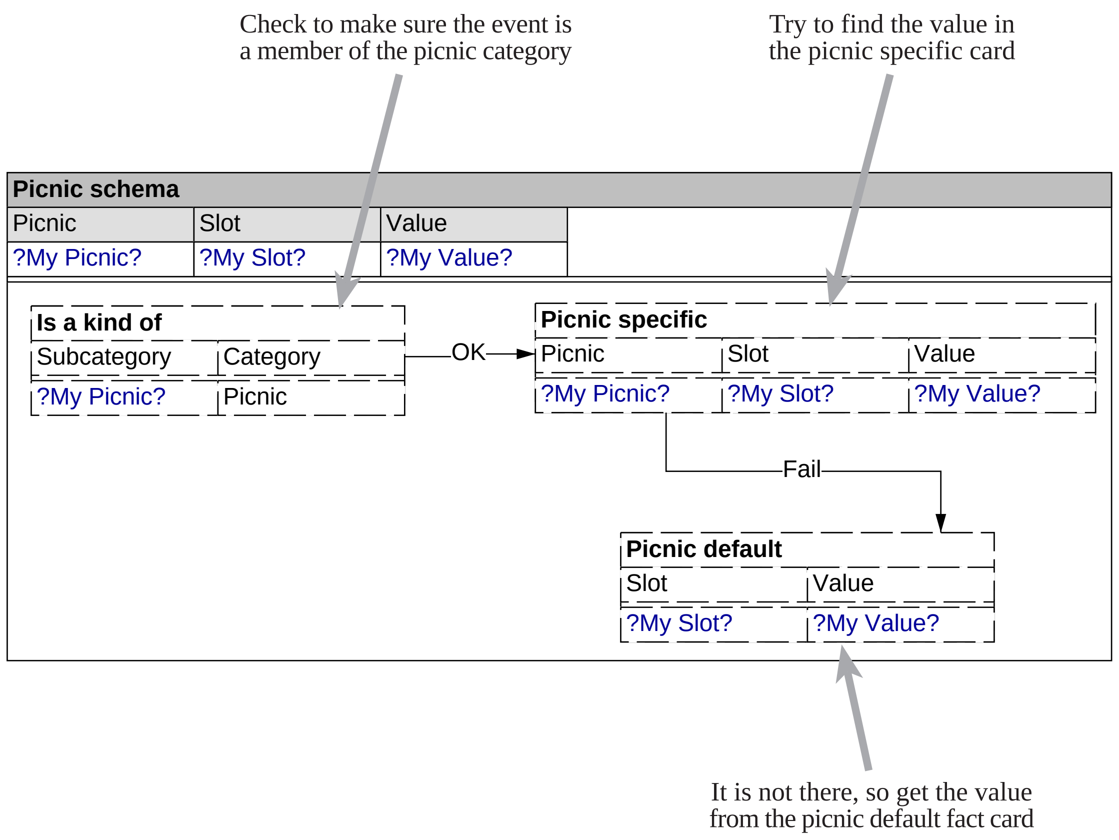
Fact card 8

Picnic specific		
Picnic	Slot	Value
Bob's picnic	Food	Hot dogs

The instruction card now needs to be extended so that specific as well as default information can be inherited. First, as before, a check must be made to make sure the event is a member of the picnic category. Second, a check must be made to see if the required information is stored as a specific value, and if so, take the specific value. Third, if the required information is not a specific value then the default information for a picnic can be found.

Our new instruction card, incorporating specific as well as default values, is a little more complex than those we have seen previously, as there are now two ways in which the instruction card can work rather than one. In the last section when we constructed an instruction card to describe cultured people, there was only one method by which someone could fit the definition of being cultured: they had to like someone, *and* that someone had to be an opera singer. Similarly, in section 7.2, there was only one way in which information about a picnic could be accessed: the event has to be a member of the picnic category (i.e. stored in the 'Is a kind of' fact card), *and* the default information for the required slot had to be found (i.e. stored in the 'Picnic default' fact card). For our new instruction card, there are two ways in which information about a picnic can be accessed. The particular event has to be a picnic, but the information we require may be stored in *either* the 'Picnic specific' fact card, *or* in the 'Picnic default' fact card.

In order to construct an instruction card that can have two different routes for accessing information, we need to introduce another feature of the Hank language, called the **Fail arrow**. So far we have seen how questions in the process box of an instruction card can be linked together using an OK arrow. The OK arrow means that if the question successfully matches a data row in the fact card, then we follow the arrow to the next question. If the question cannot be answered then the reply is 'Fail', and we do not follow the arrow to the next question. For our cognitive model of schema theory (and for many other cognitive models) we need to carry on and ask a further question when a question fails. To do this we need a Fail arrow, so if the question processor *fails* to find the information in the 'Picnic specific' fact card, it carries on, and tries to find it in the 'Picnic default' fact card. Our new 'Picnic schema' instruction card looks like this:



This ‘Picnic schema’ instruction card, plus the three related fact cards, shows how a schema can be represented in Hank. The process represented in the instruction card has three parts: (i) checking category membership, (ii) checking for a specific value, and (iii) getting the default value if there is no specific value.

If, using the instruction card and fact cards above, we asked the question (in English ‘What was eaten at Bob’s picnic?’):

Picnic schema		
Picnic	Slot	Value
Bob’s picnic	Food	?My Value?

we would get the answer:

Status = OK
?My Value? = Hot dogs

The question processor would first match the question against the wish box of the instruction card. This results in the ?My Picnic? variable taking on the value ‘Bob’s picnic’, and the ?My Slot? variable taking on the value ‘Food’. The question processor then moves to the first question in the process box, which due to the ?My Picnic? variable having taken on a value is (in English ‘Is Bob’s picnic a picnic?’):

Is a kind of	
Subcategory	Category
Bob’s picnic	Picnic

This matches the second data row of fact card 6, so we can move along the OK arrow to the next question. As the ?My Picnic? and ?My Slot? variables have taken on values, the question is (in English 'What is eaten at Bob's picnic?'):

Picnic specific		
Picnic	Slot	Value
Bob's picnic	Food	?My Value?

This matches the first data row of fact card 8, and the ?My Value? variable takes on the value 'Hot dogs'. The question processor only follows the Fail arrow if it fails to find a matching data row, so because it matched successfully here, the process is complete.

On the other hand, if we asked the question (in English 'What is eaten at Sue's party?'):

Picnic schema		
Picnic	Slot	Value
Sue's party	Food	?My Value?

The question processor would first match the question against the wish box of the instruction card. This results in the ?My Picnic? and ?My Slot? variables taking on the values 'Sue's party' and 'Food' respectively. The question processor then moves to the first question in the process box, which will be (in English 'Is Sue's party a picnic?'):

Is a kind of	
Subcategory	Category
Sue's party	Picnic

This matches the fourth data row of the 'Is a kind of' fact card, so the question processor follows the OK arrow to the next question, which is (in English 'What is eaten at Sue's party?'):

Picnic specific		
Picnic	Slot	Value
Sue's party	Food	?My Value?

This does not match any of the data rows in fact card 8. As the question processor failed to find a match to the question, it follows the Fail arrow to the next question, which will be (in English 'What is eaten at picnics?'):

Picnic default	
Slot	Value
Food	?My Value?

This matches the second data row of fact card 7, and the ?My Value? variable takes on the value 'Sandwiches'. The reply therefore is:

Status = OK
?My Value? = Sandwiches

It is important to note that the ordering of the questions in the 'Picnic schema' instruction card is of vital importance to the model. Category membership must be checked first. If the event is not a member of the picnic category then the picnic schema as a whole is inappropriate. The 'Picnic specific' question is then asked before the 'Picnic default' question. The process should only inherit the default value if there is no relevant specific value.

As well as replacing default information with specific information about a particular picnic, specific values can also be used to add extra information to particular picnics for which there is no slot in the default schema. For example, we do not have a slot in our picnic schema to indicate how much it costs to attend the picnic. It tends not to be an issue for the vast majority of picnics. However, let's assume that the works outing does have a cost and it is expensive. This information can also be added to the 'Picnic specific' fact card, by thinking up a new slot that applies only to the works outing which we can call 'Cost'. This is shown below.

Fact card 8 (updated)

Picnic specific		
Picnic	Slot	Value
Bob's picnic	Food	Hot dogs
Works outing	Cost	Expensive

Now if we ask the question:

Picnic schema		
Picnic	Slot	Value
Works outing	Cost	?My Value?

we would get the answer:

Status = OK
?My Value? = Expensive

The works outing would pass the category membership test (i.e. the 'Is a kind of' question) and then follow the OK arrow to the 'Picnic specific' question. The question matches with the second data row of the fact card. As a match was found with the 'Picnic specific' fact card, there is no need to use the default information.

However, if we asked:

Picnic schema		
Picnic	Slot	Value
Bob's picnic	Cost	?My Value?

we would get the answer:

Status = Fail

as the cost information only applies to the works outing and in our default information we do not have a slot relating to how much it costs to attend picnics in general.

SAQ 7
Add the following pieces of information to the 'Picnic specific' fact card. Decide whether each piece of information replaces an existing default value, or adds an extra slot for a specific picnic.

'It was raining at Sue's party'

'Bob Dylan provided the entertainment at Jane's birthday'

A clarification...

Part I of the *Memory* book uses the term 'optional value' rather than 'specific value'. Specific value is a more accurate and less confusing term. A default value of picnics is that sandwiches are eaten, though this does not apply to Bob's picnic, at which hot dogs are eaten. The information that hot dogs rather than sandwiches are eaten at Bob's picnic is not 'optional' in the everyday sense of the word. A more accurate description is that eating hot dogs is 'specific' information applying solely to Bob's picnic. The specific value that hot dogs are eaten at Bob's picnic overrides the default value that states that sandwiches are eaten at all picnics. For this reason, we use the term specific value rather than optional value.

7.4 Psychological considerations

In sections 7.2 and 7.3 we have worked through how schema theory can be modelled in Hank. This has illustrated how properties can be used as specific and as default values, as a result of category membership. An important concept to notice is the distinction between specific values and default values. In the example in section 7.2 the Hank program represents all picnics as being outside. This is an example of a default value. In the fact card presenting default information three default properties were given, i.e. picnics are located outside, sandwiches are eaten, and the weather is sunny. Individual picnics, though each belonging to the picnic category, will vary as to the number of default values that apply. Our first example was Bob's picnic, which although a picnic, has hot dogs rather than sandwiches. A picnic could be envisaged that fits very few of the default values.

There is a further notion in schema theory, which we have not explicitly represented in our program: **fixed values**. A fixed value is a default value that can never be overridden by a specific value, because if that particular property was not true, then the event or object would no longer be a member of the category. For example, in our model, we didn't have a specific value stating that a particular picnic is held anywhere other than outside. If a social event was not held outside, then it would no longer belong in the picnic category (i.e. it would not appear in the 'Is a kind of' fact card as being a picnic). If Sue's party was held in a nightclub rather than outside, we would no longer categorize it as being a picnic. We would probably categorize it as being a nightclub event, and then draw on a separate set of default values (e.g. tickets have to be shown at the door, there are bouncers present, and it has a cloakroom facility).

This issue of fixed values relates to the notions of defining and typical features discussed in Part I of the *Perception and Representation* course book. Defining features are those that are essential to category membership. Typical features are default values which may not necessarily hold. In our picnic example, we can think of the location being outside as a defining feature, and other features, such as the food being sandwiches, as typical features.

The aim of section 7 has been to encourage you to think about the extent to which schema theory provides a good explanation of how we represent knowledge of the world around us. As we have seen, schema theory offers an explanation of how we store stereotypical, as well as more unusual, attributes of events, people, places and objects. The advantage of using the cognitive modelling technique is that it forces us to think about how schema theory can be represented (the knowledge representation issue) and made to work in practice. When reading about schema theory in a book and its account of how statements such as:

‘All picnics are held outside’

‘Sandwiches are eaten at picnics’

‘Hot dogs are eaten at Bob’s picnic’

can be made sense of, and used by us in everyday life, the cognitive processes described can sound rather trivial. Hiding behind the account of schema theory (or any other cognitive theory) are implicit assumptions about how this knowledge can actually be represented and processed. Cognitive modelling is concerned with bringing out these assumptions by building a model of a theory and seeing it work.

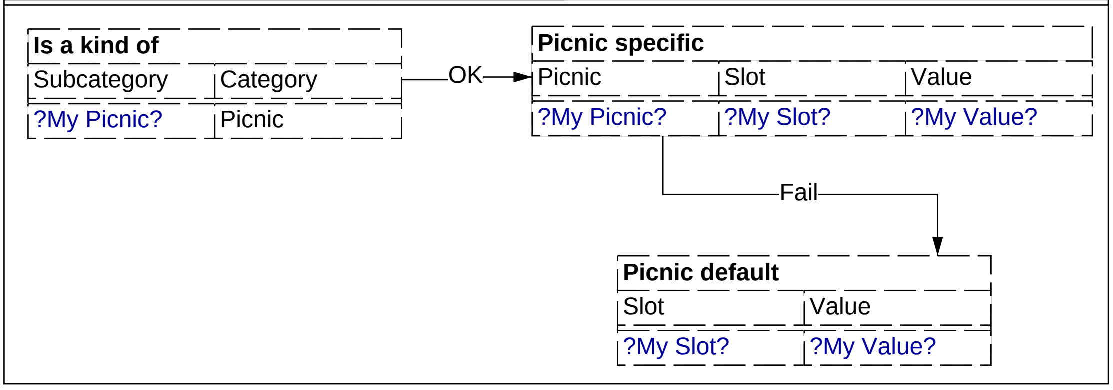
Summary of section 7

- Hank can be used to model schema theory.
- The model incorporates category membership, specific information and default information.
- Category membership checks to make sure that the event is a member of the category to which the schema applies.
- Default information represents stereotypical properties, such as sandwiches being eaten at picnics.
- Specific information represents exceptions to default information (such as hot dogs being eaten at Bob’s picnic) or additional information not included in the default information (such as the works outing being expensive).

8 Extending the picnic schema

This is the instruction card we constructed for our picnic schema:

Picnic schema		
Picnic	Slot	Value
?My Picnic?	?My Slot?	?My Value?



By asking a picnic schema question we can access specific and default values pertaining to particular picnics. For example, if we asked (in English ‘What was eaten at Jane’s birthday?’):

Picnic schema		
Picnic	Slot	Value
Jane’s birthday	Food	?My Value?

from the default value for food we would get the reply:

Status = OK
?My Value? = Sandwiches

If we asked the question (in English ‘What food was eaten at Bob’s picnic?’):

Picnic schema		
Picnic	Slot	Value
Bob’s picnic	Food	?My Value?

from the specific value for the food at Bob’s picnic we would get the reply:

Status = OK
?My Value? = Hot dogs

This forms the core of how a schema is represented in Hank. We will now show how the schema can be incorporated with other instruction cards and fact cards to allow further inferences to be made. This relates to some of the work you will have to do for TMA 03-E.

For this illustration, we will need a new fact card to say who attended each of the picnics. This is shown below.

Fact card 9

Attended	
Person	Picnic
Peter	Sue’s party
Ingrid	Jane’s birthday
Judith	Bob’s picnic
Sandy	Works outing
Jill	Jane’s birthday

The instruction card and fact cards making up our picnic schema, plus our new fact card 9, can allow extra inferences to be made. For example, we know that when we use our picnic schema model to find out what was eaten at Bob’s picnic, we get the reply ‘Hot dogs’. From fact card 9 we can see that Judith attended Bob’s picnic. We can infer from these two pieces of information that Judith ate hot dogs, because she attended Bob’s picnic (derived from fact card 9) and because hot dogs were the food at Bob’s picnic (from our picnic schema model). Similarly, we can infer that Ingrid ate sandwiches, as she attended Jane’s birthday (from fact card 9) and sandwiches were the food at Jane’s birthday (from our picnic schema model).

So, in order to find out what someone ate we need to ask two questions. First we need to find out which picnic they attended. Second, we need to find out what was eaten at that particular picnic. To find out what Judith ate, our first question would be (in English ‘Which picnic did Judith attend?’):

Attended	
Person	Picnic
Judith	?My Picnic?

which would give the reply:

Status = OK
?My Picnic? = Bob’s picnic

Then, to find out what was eaten at the picnic Judith attended, we would ask (in English ‘What was eaten at Bob’s picnic?’):

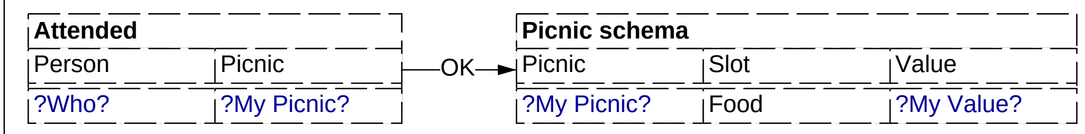
Picnic schema		
Picnic	Slot	Value
Bob’s picnic	Food	?My Value?

and the reply would be:

Status = OK
?My Value? = Hot dogs

So far, all the instruction cards we have constructed ask questions that match on to fact cards. The questions asked in the process box of an instruction card can also match on to the wish box of another instruction card. We can therefore construct a further instruction card to find out what people ate, which links together a fact card (i.e. fact card 9) and an instruction card (i.e. ‘Picnic schema’). Our new instruction card for inferring what someone ate is shown below.

Ate	
Person	Food
?Who?	?My Value?



Attended	
Person	Picnic
?Who?	?My Picnic?

Picnic schema		
Picnic	Slot	Value
?My Picnic?	Food	?My Value?

Using our new instruction card, we can ask (in English ‘What did Sandy eat?’):

Ate	
Person	Food
Sandy	?My Value?

The question processor would first match the question against the wish box of the Ate instruction card. This would result in the ?Who? variable taking on the value ‘Sandy’. The question processor then moves to the first question in the process box. As the ?Who? variable has taken on the value ‘Sandy’ the question will be (in English ‘Which picnic did Sandy attend?’):

Attended	
Person	Picnic
Sandy	?My Picnic?

This question matches the fourth data row of fact card 9, and the ?My Picnic? variable takes on the value ‘Works outing’. As a matching data row was found, the question processor follows the OK arrow to the next question. As the ?My Picnic? variable has taken on a value, the question will be (in English ‘What food was eaten at the works outing?’):

Picnic schema		
Picnic	Slot	Value
Works outing	Food	?My Value?

This question matches the wish box of the Picnic schema instruction card. The ?My Picnic? variable in the wish box of the Picnic schema instruction card takes on the value ‘Works outing’. The ?My Slot? variable takes on the value ‘Food’. The question processor then moves to the first question of the process box of the Picnic schema instruction card. As the ?My Picnic? variable has taken on a value, the question will be (in English ‘Is the works outing a picnic?’):

Is a kind of	
Subcategory	Category
Works outing	Picnic

As this question matches the third data row of fact card 6, the question processor follows the OK arrow to the next question. As the variables ?My Picnic? and ?My Slot? have taken on values, the question is (in English ‘What food was eaten at the works outing?’):

Picnic specific		
Picnic	Slot	Value
Works outing	Food	?My Value?

Because this does not match any of the data rows of fact card 8, the question processor follows the Fail arrow to the next question. As the ?My Slot? variable has taken on a value, the question is (in English ‘What is eaten at picnics?’):

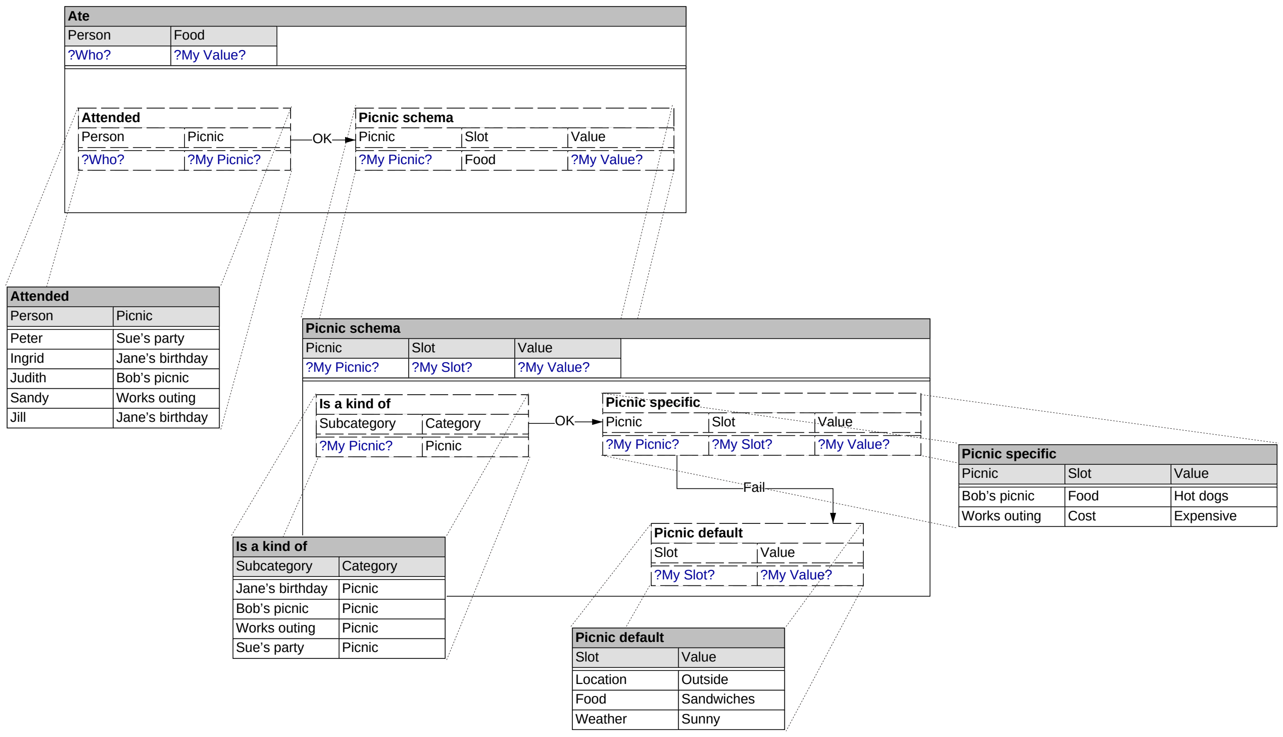
Picnic default	
Slot	Value
Food	?My Value?

this matches the second data row of fact card 7, and the ?My Value? variable takes on the value ‘Sandwiches’. As we have now completed answering the Picnic schema question, we can finish replying to the Ate question asked at the beginning. The reply will be:

Status = OK
?My Value? = Sandwiches

We have therefore found out through a process of inference that Sandy ate sandwiches, by finding out which picnic he attended, and then using our Picnic schema model to find out what was eaten there.

The connections between the instruction cards and fact cards used in our picnic example are represented graphically below.



Summary of section 8

- The schema model can be combined with related fact cards and instruction cards to allow further inferences to be made.
- Questions in the process box of an instruction card can match the wish box of another instruction card as well as a data row in a fact card.

TMA 03-E
At this point complete TMA 03-E.

9 Current issues in cognitive modelling

In the *Offprints Booklet*, you'll find 'Trends and debates in cognitive psychology', written by George Miller. This article was originally written in 1981, and since then the debates and issues have changed a bit. In this section, we'll return to the issues raised by Miller, and show how cognitive modelling in psychology has changed over the intervening years.

We'll also discuss some helpful defences and criticisms of cognitive modelling and the related field of artificial intelligence. Of these, perhaps the most persuasive criticisms are of the 'strong' artificial intelligence view that computers running the right programs — or the right cognitive models — really are living (if not breathing) things with minds in their own right. We'll look at one of these criticisms, Searle's 'Chinese Room' argument, in a bit of detail, and show that there are some interesting lessons to be drawn from these criticisms of artificial intelligence — lessons which can throw a new light on cognitive modelling.

Artificial intelligence and cognitive psychology

Miller talks about artificial intelligence — the study of how to build programs that show intelligent behaviour — rather more than we have done. Miller himself argued that the goals of artificial intelligence and of cognitive modelling are different, and that because of this the fields may begin to diverge. It is still true that many in the field of artificial intelligence think about 'intelligence in general' (French, 1990) as a desirable goal, and try to step outside, or even beyond, human intelligence. Miller's view of artificial intelligence was dominant in its time: 'indeed, a truly general theory of intelligence will not be achieved until we can characterise human intelligence as a special case' (Miller, 1981). This is still the most common view in the artificial intelligence community — and perhaps represents one of the points which are beginning to separate it from the cognitive modelling approach we are using within this project.

So, as Miller suggested there might be, there has been an element of divergence between the goals of artificial intelligence and of cognitive modelling. But there have also been a number of challenges to the principles behind artificial intelligence, and some of these problems have rubbed off on cognitive modelling. We'll come on to some of the bigger challenges later in this section, when we discuss the 'Chinese Room' model, and discuss artificial intelligence a bit more in this light. For now, we'll just look briefly at one of these challenges, which was anticipated by Miller, the problems of consciousness and testability.

Consciousness

Although Miller mentioned consciousness as an issue, it is only fairly recently that consciousness has become an important theme in psychology. Today, it is being taken much more seriously and much more actively, both in cognitive psychology and cognitive science, and elsewhere in neuroscience, biology, and even physics. Studying consciousness has become rather fashionable since Miller's article was written.

Miller's article uses consciousness as one possible inescapable incompleteness within artificial intelligence, yet there is little — if any — evidence for this. Although it is true that consciousness has proved remarkably hard to pin down in any form that can be modelled using similar techniques to those we've used in this project, cognitive theories of consciousness are beginning to emerge (e.g. Baars, 1988).

Searle's (1980) 'Chinese Room' argument (discussed below) also has implications for consciousness. In the middle of Searle's argument is a distinction he draws between *simulation* and *replication*: 'No one supposes that computer simulations of a fire will burn the neighborhood down or that a computer simulation of a rainstorm will leave us all drenched' (Searle, 1980). By analogy, Searle argues that a computer model of consciousness is not consciousness itself. In practice, although arguments from consciousness (and there are also similar arguments from emotions, and so on) can seem intuitively very strong, when analysed in detail they turn out to rest mostly on individual intuitions — and this is certainly true of Searle's.

At the moment, consciousness remains pretty much a mystery — and a rather ill-defined mystery at that. It is still far too early to assume that there is an inevitable incompleteness in the approach of cognitive modelling which can be put firmly at the door of consciousness. At the moment, all that can be said with conviction is that nobody has succeeded in building a system that is conscious and which is capable of convincing us that it is conscious.

Testability

Miller discusses the issue of testability in some detail. Here the question is quite simple: how do we evaluate a cognitive model? Miller's view is that even though verbal reports may be flawed, under the right conditions they can be good enough to test for psychological plausibility. However, verbal reports are not the only kind of evidence available when evaluating a model. Let's take Collins and Quillian's model of conceptual categories (in *Perception and Representation*) as an example. Collins and Quillian evaluated the response times required to verify statements, and compared those response times to estimated response times from a cognitive model.

Clearly, no direct comparison of response times is legitimate — for one thing, the speed of a computer can vary immensely without any change in a program, and therefore measuring computer speed is not psychologically valid. Instead, a kind of 'normalized' comparison, looking for trends in the number of steps needed to run a model, or even designing the model so that a predicted human-equivalent time is associated with each step, may be more legitimate. This second strategy is adopted by ACT-R (Anderson and Lebiere, 1998), the latest in the ACT series of models (*Memory*, Part IIB, section 2). Figure 2 shows a comparison between experimental data and a model's predictions for one simple experiment using ACT-R. In this case, the values compared are the times taken for each move when solving successfully the four-ring Towers of Hanoi problem. Figure 2 clearly shows the exceptionally close correspondence between this particular model's predictions and the experimental results, and does seem to show that this particular model is pretty strong. So while verbal reports may be one way of looking at cognitive models, they are definitely not the only way; there are many other ways of evaluating cognitive models.

Can machines think? Searle's 'Chinese Room' argument

So far in this project, we have taken a fairly weak line on cognitive models; we haven't made any claims about whether these models in any sense are entities that have minds in their own right. Even so, this is an important issue; could a computer running a very large cognitive model really be taken as an entity with its own mind and morals?

Here we leave cognitive psychology as a discipline, and more or less enter **cognitive science**, a more interdisciplinary area involving collaborations between psychologists, philosophers, artificial intelligence researchers, neuroscientists, linguists, and more or less anybody else who is interest in how people come to have minds in the first place.

In Eysenck and Keane's *Cognitive Psychology: A Student's Handbook* you'll find a brief description of Searle's 'Chinese Room' thought experiment. If you read this description, and compare it to the way we have described Hank in this project, you should notice that there is a distinct similarity between the two. This is not an accident.

Put simply, Searle's thought experiment goes a bit like this. Searle imagines himself locked inside a room, which can communicate with the outside world through something like a letter-box. People outside can ask questions of the room, by writing them down (in Chinese) and posting them through the letter-box. Inside the room, Searle has a set of rules and procedures, and a book of instructions written in English. When a question arrives inside the room, Searle reads the instructions, and uses the rules to look for particular patterns in the questions. For example, he may have a rule which says 'when you see *squiggle-squiggle* in the question, write *squoggle-squoggle* for the answer'.

Now, to someone outside the room, the system may seem to understand Chinese perfectly. Searle's argument is that, inside the room, there is nothing there that genuinely does *understand* Chinese. After all, Searle (inside the room) doesn't understand Chinese at all himself; he only has the instruction book — which is written in English. So, according to Searle, simply because you are generating the appearance of understanding so well that

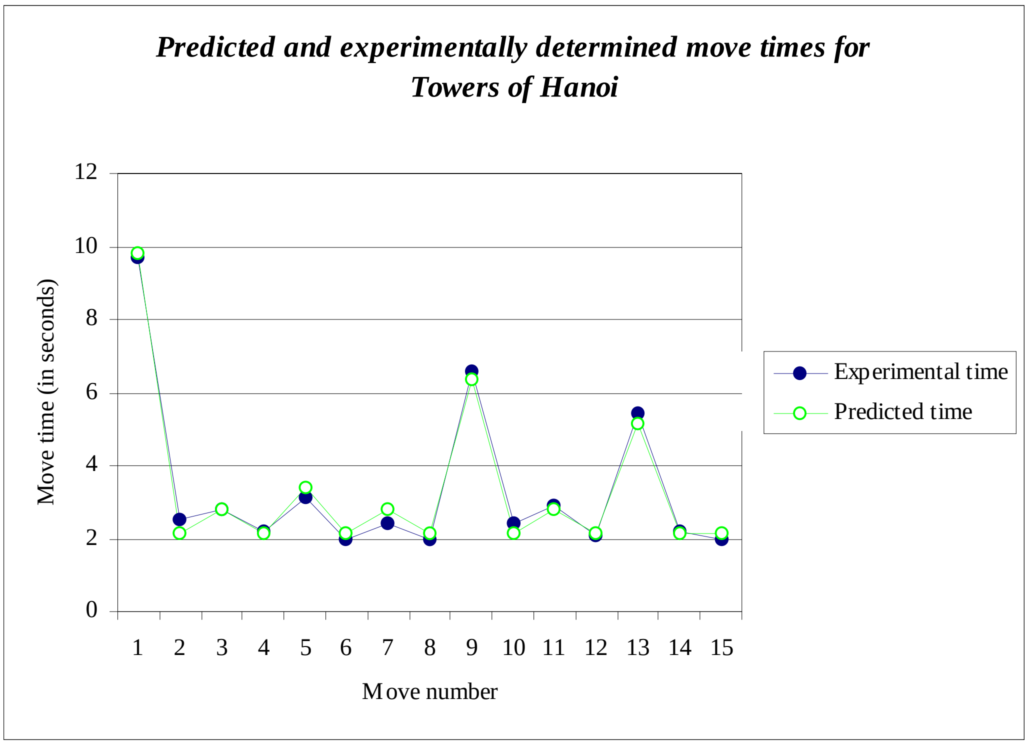


Figure 2 Comparison of predicted (by ACT-R, Anderson and Lebiere, 1998) and experimentally determined (by Anderson, Kushmerick and Lebiere, 1993) move times for the four ring Towers of Hanoi problem

you've managed to fool everybody, that doesn't necessarily mean that you actually do understand. There are a lot of problems with Searle's argument; one reply, for example, suggests that Searle does have a kind of understanding — the kind of understanding needed to use the instruction book (Boden, 1988). But even if his argument is wrong in places, Searle does raise some issues that are important to cognitive modelling as an approach.

Searle himself distinguishes between *weak* and *strong* artificial intelligence. According to weak artificial intelligence, computers are just tools that happen to be particularly valuable when it comes to studying the mind. Strong artificial intelligence, on the other hand, argues that an 'appropriately programmed computer really *is* a mind, in the sense that computers given the right programs can literally be said to *understand* and have other cognitive states' (Searle, 1980, p.417, original emphasis). In fact, Searle doesn't have a problem with weak artificial intelligence (which is, more or less, cognitive modelling) — it is strong artificial intelligence that gets his goat.

Clark (1988) points out that there are two kinds of account which a model like those build in Hank can provide — models can be either *descriptive* or *causal*. A descriptive model is just that — a model. A causal model is much stronger — for example, if we were to take Schank's model of language understanding as a causal model, it would imply that somewhere inside us there really were things that functioned as his primitive acts. The difference between a descriptive and a causal account is like the difference between a Lego house and a real house. A Lego house does not pretend to be the kind of thing you could actually live in. And in just the same way, cognitive models do not pretend to be real minds! Even so, we can use Lego houses to describe the differences between bungalows, cottages, and so on — the differences that exist in real houses — without going to the cost of actually building them. Lego houses, and computational models, can be very useful if you take them as descriptions rather than as real houses or as real minds.

To return to Hank; there is a clear similarity between the question processor and Searle (in the Chinese Room). Both have a database of procedures, and both have an instruction book which they use to apply those procedures to generate apparently intelligent behaviour. Yet neither the question processor nor Searle (in the room) have true understanding. Despite this, are there any conclusions we can draw about the models that we build, and which Hank runs for us?

At the descriptive level of weak artificial intelligence, Searle's argument is now a bit of a red herring. After all, people have been doing cognitive modelling for many years now, and it really does seem to work. Even though there is no question processor actually inside people's heads, we can build models in Hank, just like Lego, which may be helpful to cognitive psychology. Here, the proof of the pudding is in the eating, and cognitive modelling does seem to help in practice, at least sometimes.

Strong artificial intelligence is far more controversial. On one level, the argument is a bit pointless. As Searle himself says: '"Could a machine think?" The answer is, obviously, yes. We are precisely such machines' (Searle, 1980). So, presumably, machines can think, at least in principle. The argument hinges on what kind of machine is needed to think; for example, does it have to be a brain, or will an appropriately programmed computer do? And then the arguments start to get really nasty.

Summary of section 9

- Cognitive modelling has advanced since Miller's article was written.

- Cognitive modelling is now an established part of cognitive science and cognitive psychology — for the most part ‘strong’ artificial intelligence plays no part in it.
- There are many ways of evaluating cognitive models, not just verbal reports.
- There is a difference between cognitive modelling (weak artificial intelligence) and the project of building machines that think (strong artificial intelligence).
- Cognitive modelling is less vulnerable to argument, and can be evaluated by its success even today.
- Searle’s (1980) ‘Chinese Room’ argument attacks the idea of strong artificial intelligence, suggesting it is impossible, *even in principle,* to program a computer to truly understand.
- Searle’s argument is not completely sound, although it is still an open question whether it is, *in practice*, possible to program a computer to truly understand.
- Consciousness is a popular and often appealing way of criticizing artificial intelligence and cognitive modelling, but it is both controversial and dangerous.

References

Anderson, J.R., Kushmerick, N. and Lebiere, C. (1993) ‘The Tower of Hanoi and goal structures’, In Anderson, J.R. (ed.) *Rules of the Mind*, Hillsdale, Lawrence Erlbaum.

Anderson, J.R. and Lebiere, C. (1998) *The Atomic Components of Thought*, Hillsdale, Lawrence Erlbaum.

Baars, B.J. (1988) *A Cognitive Theory of Consciousness*, Cambridge, Cambridge University Press.

Boden, M.A. (1988) *Computer Models of Mind*, Cambridge, Cambridge University Press.

Clark, A. (1988) ‘Thoughts, sentences, and cognitive science’, *Philosophical Psychology,* **1**(3), 263-273.

French, R.M. (1990) ‘Subcognition and the limits of the Turing test’, *Mind,* **99**, 53-65.

Miller, G.A. (1956) ‘The magical number seven plus or minus two’, *Psychological Review,* **63**, 81-97.

Miller, G. (1981) ‘Trends and debates in cognitive psychology’, *Cognition,* **10**, 215-225.

Searle, J.R. (1980) ‘Minds, brains, and programs’, *Behavioural and Brain Sciences,* **3**, 417-424.

Turkle, S. (1988) ‘Artifical intelligence and psychoanalysis: a new alliance’, *Daedalus,* **117**(1), 241-268.

Acknowledgements

We would like to thank the D309 course team members, Sandy Aitkenhead, Jill Cohen, Judith Greene, Peter Naish and Ingrid Slack for their many helpful suggestions. They provided extensive feedback on numerous drafts of this TMA and the design of Hank itself. We are grateful to the many other people who offered advice on the design of Hank and this TMA — in particular, Trevor Collins and Clayton Lewis, who undertook a detailed analysis of this assignment. We would also like to thank the students and tutors who gave us feedback on an earlier version of this assignment and the residential school version of the cognitive modelling project.

Index of concepts

artificial intelligence, 6
card name, 11
category membership, 40
cognitive models, 3
cognitive science, 58
column labels, 12
computer metaphor, 6
computers, 5
data rows, 12
database, 8
default values, 43
fact cards, 9, 11, 21
Fail arrow, 45
fixed values, 49
house rules, 8, 16, 20
instruction cards, 9, 26
instructions, 5
knowledge representation, 21
matching, 13
models, 3
OK arrow, 32
process box, 27
processes, 3
question processor, 8
questions, 13
representations, 4
specific values, 44
symbol structures, 11
symbols, 4
values, 12
variables, 17
wish box, 27
workspace, 8

Answers to SAQs

SAQ 1

We would represent the telephone book as below, using columns for name, address and number. For your own answer you may have used a different card name, and different column labels.

Telephone book		
Name	Address	Number
Adams, R.	49 Low Street	52287
Briggs, B.	27 Bay Street	35635
Hawkins, S.	4 Green Street	67456
James, W.	65 Leaf Lane	73463
Lewis, K.	12 High Street	21341
Peters, F.	2 Flower Lane	85676

SAQ 2

The Hank question for ‘Does a rectangle have 5 sides?’ is:

Sides	
Name	Number
Rectangle	5

and the reply would be:

Status = Fail

as the data row in the question does not match any of the data rows in fact card 1.

The Hank question for ‘Did Jenny give Bill a beer?’ is:

Who gave what to who		
From	To	Object
Jenny	Bill	Beer

and the reply would be:

Status = OK

as the question matches the second data row of fact card 3.

SAQ 3

(a) Illegal. A variable cannot be used as a column label.
(b) Illegal, A variable cannot be used as a card name.
(c) Legal. This question does not contain any variables.
(d) Legal. All the variables appear in the data row of the question.

SAQ 4

If you just looked at the first of the three statements you might be tempted to design a fact card called something like ‘Car details’ and have three columns for the owner, car model, and car colour. When you look at the second and third statements you will see that this would not be a good representation as we do not know the colour of Ben’s car, or the model of Mary’s car. We therefore need to split the information into two fact cards. One fact card relates the owner to the make of the car, and the other relates the owner to the colour of the car.

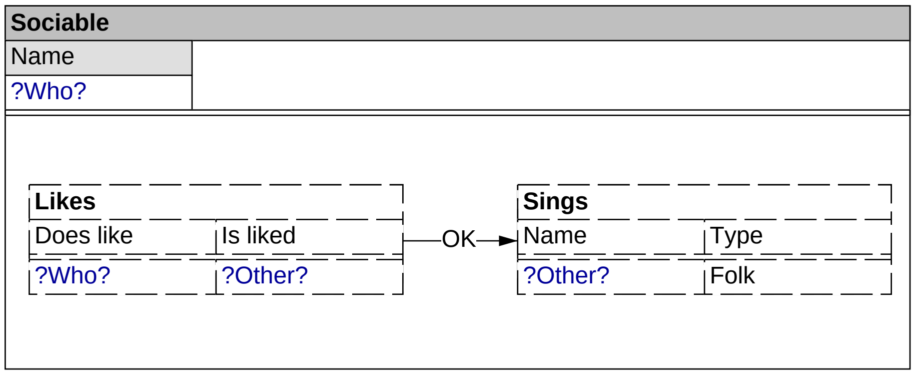
Our versions are shown below. You may have used different card names or column labels. Also, you may have placed the data rows in a different order. Any of these differences between your answer and our answer are fine, and just indicate personal preference.

Car colour	
Owner	Colour
Joan	Blue
Mary	Red

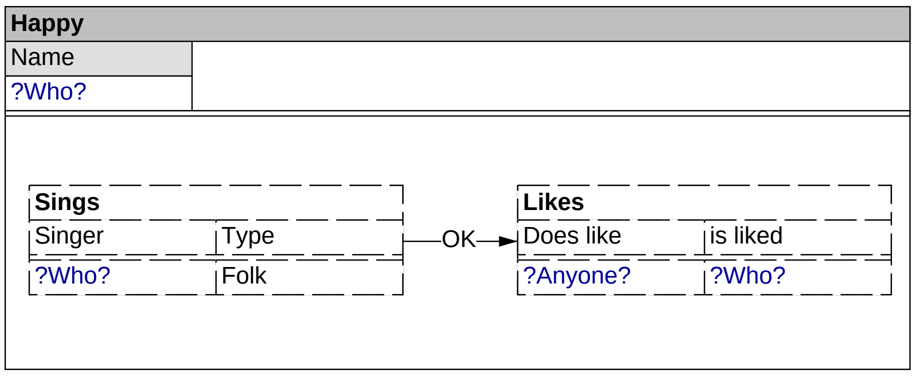
Car model	
Owner	Make
Ben	Escort
Joan	Mondeo

SAQ 5

(a) The variable in the wish box should be the same as the one in the Does like column of the likes question. The variable in the Is liked column of the Likes question should be the same as the one in the Name column of the Sings question. The Type column should contain the value Folk. Our version is shown below.



(b) The variable in the wish box should also appear in the Name column of the Sings question, as the person who is happy is the person who sings. The variable should also appear in the Is liked column of the Likes question, as the happy person is liked by someone. The variable in the Does like column of the Like question does not appear in the wish box, as we are not interested in who does the liking. Our version is shown below.



SAQ 6

Our version of the furniture hierarchy is shown below. You may have used a different card name or different column labels. Also, you may have placed the data rows in a different order. Any of these differences between your answer and our answer are fine, and just indicate personal preference.

Is a kind of	
Subcategory	Category
Kitchen table	Table
Dining-room table	Table
Kitchen chair	Chair
Living-room chair	Chair
Table	Furniture
Chair	Furniture

SAQ 7

If we added those two further pieces of information to the Picnic specific fact card it would look something like this.

Picnic specific		
Picnic	Slot	Value
Bob's picnic	Food	Hot dogs
Works outing	Cost	Expensive
Sue's party	Weather	Raining
Jane's birthday	Entertainment	Bob Dylan

For the information on Sue's picnic, the slot must be called Weather, as this is the default value that it replaces. For the information about Jane's birthday, we need to think up a new slot name, as this adds an extra slot, applying only to Jane's birthday. We chose the slot name 'Entertainment'. You may have chosen a slightly different name such as 'Music'.

Fact card 1

Sides	
Name	Number
Triangle	3
Rectangle	4

Fact card 2

Shape
Example
Triangle
Rectangle
Pentagon

Fact card 3

Who gave what to who		
From	To	Object
Peter	Joan	Apple
Jenny	Bill	Beer
Anna	RSPCA	Money

Fact card 4

Likes	
Does like	Is liked
John	Pavarotti
Susan	Bob Dylan
Jenny	Peter
Peter	Jenny

Fact card 5

Sings	
Name	Type
Pavarotti	Opera
Bob Dylan	Folk

Fact card 6

Is a kind of	
Subcategory	Category
Jane’s birthday	Picnic
Bob’s picnic	Picnic
Works outing	Picnic
Sue’s party	Picnic

Fact card 7

Picnic default	
Slot	Value
Location	Outside
Food	Sandwiches
Weather	Sunny

Fact card 8

Picnic specific		
Picnic	Slot	Value
Bob’s picnic	Food	Hot dogs
Works outing	Cost	Expensive

Fact card 9

Attended	
Person	Picnic
Peter	Sue’s party
Ingrid	Jane’s birthday
Judith	Bob’s picnic
Sandy	Works outing
Jill	Jane’s birthday