MICHAEL NEGNEVITSKY

THIRD EDITION

ARTIFICIAL INTELLIGENCE

A GUIDE TO INTELLIGENT SYSTEMS

Artificial Intelligence

Frame-based expert systems 5

In which we introduce frames as one of the common methods used for representing knowledge in expert systems, describe how to develop a frame-based expert system and illustrate the theory through an example.

5.1 Introduction, or what is a frame?

Knowledge in a computer can be represented through several techniques. In the previous chapters, we considered rules. In this chapter, we will use **frames** for the knowledge representation.

What is a frame?

A frame is a data structure with typical knowledge about a particular object or concept. Frames, first proposed by Marvin Minsky in the 1970s (Minsky, 1975), are used to capture and represent knowledge in a frame-based expert system. Boarding passes shown in Figure 5.1 represent typical frames with knowledge about airline passengers. Both frames have the same structure.

Each frame has its own name and a set of **attributes**, or **slots**, associated with it. *Name*, *weight*, *height* and *age* are slots in the frame *Person*. *Model*, *processor*, *memory* and *price* are slots in the frame *Computer*. Each attribute or slot has a value attached to it. In Figure 5.1(a), for example, slot *Carrier* has value *QANTAS AIRWAYS* and slot *Gate* has value *2*. In some cases, instead of a particular value, a slot may have a procedure that determines the value.

In expert systems, frames are often used in conjunction with production rules.

Why is it necessary to use frames?

Frames provide a natural way for the structured and concise representation of knowledge. In a single entity, a frame combines all necessary knowledge about a particular object or concept. A frame provides a means of organising knowledge in slots to describe various attributes and characteristics of the object.

Earlier we argued that many real-world problems can be naturally expressed by IF-THEN production rules. However, a rule-based expert system using a

QANTAS BOARDING PASS	
Carrier:	QANTAS AIRWAYS
Name:	MR N BLACK
Flight:	QF 612
Date:	29DEC
Seat:	23A
From:	HOBART
To:	MELBOURNE
Boarding:	0620
Gate:	2

(a)

AIR NEW ZEALAND BOARDING PASS	
Carrier:	AIR NEW ZEALAND
Name:	MRS J WHITE
Flight:	NZ 0198
Date:	23NOV
Seat:	27K
From:	MELBOURNE
To:	CHRISTCHURCH
Boarding:	1815
Gate:	4

(b)

Figure 5.1 Boarding pass frames

systematic search technique works with facts scattered throughout the entire knowledge base. It may search through the knowledge that is not relevant to a given problem, and, as a result, the search may take a great deal of time. If, for example, we are searching for knowledge about Qantas frequent flyers, then we want to avoid the search through knowledge about Air New Zealand or British Airways passengers. In this situation, we need frames to collect the relevant facts within a single structure.

Basically, frames are an application of object-oriented programming for expert systems.

What is object-oriented programming?

Object-oriented programming can be defined as a programming method that uses **objects** as a basis for analysis, design and implementation. In object-oriented programming, an object is defined as a concept, abstraction or thing with crisp boundaries and meaning for the problem at hand (Blaha and Rumbaugh, 2004). All objects have identity and are clearly distinguishable. *Michael Black*, *Audi 5000 Turbo*, *HP Pavilion P6555A* are examples of objects.

An object combines both data structure and its behaviour in a single entity. This is in sharp contrast to conventional programming, in which data structure and the program behaviour have concealed or vague connections.

Object-oriented programming offers a natural way of representing the real world in a computer, and also illuminates the problem of data dependency, which is inherent in conventional programming (Weisfeld, 2008). When programmers create an object in an object-oriented programming language, they first assign a name to the object, then determine a set of attributes to describe the object's characteristics, and at last write procedures to specify the object's behaviour.

A knowledge engineer refers to an object as a **frame**, the term introduced by Minsky, which has become the AI jargon. Today the terms are used as synonyms.

5.2 Frames as a knowledge representation technique

The concept of a frame is defined by a collection of **slots**. Each slot describes a particular attribute or operation of the frame. In many respects, a frame resembles the traditional 'record' that contains information relevant to typical entities. Slots are used to store values. A slot may contain a default value or a pointer to another frame, a set of rules or procedure by which the slot value is obtained. In general, slots may include such information as:

1. **Frame name**.
2. **Relationship of the frame to the other frames**. The frame *IBM Aptiva S35* might be a member of the class *Computer*, which in turn might belong to the class *Hardware*.
3. **Slot value**. A slot value can be symbolic, numeric or Boolean. For example, in the frames shown in Figure 5.1, the slot *Name* has symbolic values, and the slot *Gate* numeric values. Slot values can be assigned when the frame is created or during a session with the expert system.
4. **Default slot value**. The default value is taken to be true when no evidence to the contrary has been found. For example, a car frame might have four wheels and a chair frame four legs as default values in the corresponding slots.
5. **Range of the slot value**. The range of the slot value determines whether a particular object or concept complies with the stereotype requirements defined by the frame. For example, the cost of a computer might be specified between $750 and $1500.
6. **Procedural information**. A slot can have a procedure (a self-contained arbitrary piece of computer code) attached to it, which is executed if the slot value is changed or needed. There are two types of procedures often attached to slots:
   (a) *WHEN CHANGED* procedure is executed when new information is placed in the slot.
   (b) *WHEN NEEDED* procedure is executed when information is needed for the problem solving, but the slot value is unspecified.

   Such procedural attachments are often called **demons**.

Frame-based expert systems also provide an extension to the slot-value structure through the application of **facets**.

What is a facet?

A facet is a means of providing extended knowledge about an attribute of a frame. Facets are used to establish the attribute value, control end-user queries, and tell the inference engine how to process the attribute.

In general, frame-based expert systems allow us to attach value, prompt and inference facets to attributes. **Value facets** specify default and initial values

of an attribute. **Prompt facets** enable the end-user to enter the attribute value on-line during a session with the expert system. And finally, **inference facets** allow us to stop the inference process when the value of a specified attribute changes.

What is the correct level of decomposition of a problem into frames, slots and facets?

Decomposition of a problem into frames, frames into slots and facets depends on the nature of the problem itself and the judgement of the knowledge engineer. There is no predefined 'correct' representation.

Figure 5.2 illustrates frames describing desktop computers. The topmost frame represents the class *Computer* and the frames below describe instances *HP Pavilion P6555A* and *Acer Aspire M3910*. Two types of attributes are used here: *string* [Str] for symbolic information and *numeric* [N] for numeric data. Note default and initial value facets attached to the slots *Optical Drive*, *Keyboard*, *Warranty* and *Stock* in the class *Desktop*. The attribute names, types, default and initial values are the properties inherited by instances.

What are the class and instances?

The word 'frame' often has a vague meaning. The frame may refer to a particular object, for example the desktop *HP Pavilion P6555A*, or to a group of similar objects. To be more precise, we will use the **instance-frame** when referring to a particular object, and the **class-frame** when referring to a group of similar objects.

A class-frame describes a group of objects with common attributes. *Animal*, *person*, *car* and *computer* are all class-frames. In AI, however, the abbreviation 'class' is often used instead of the term 'class-frame'.

Each frame in a frame-based system 'knows' its class. In other words, the frame's class is an implicit property of the frame. For example, instances in Figure 5.2 identify their class in the slot *Class*.

If objects are the basis of the frame-based systems, why bother with classes?

Grouping objects into classes helps us to represent a problem in an abstract form. Minsky himself described frames as 'data structures for representing **stereotyped** situations'. In general, we are less concerned with defining strictly and exhaustively the properties of each object, and more concerned with the salient properties typical for the entire class. Let us take, for example, the class of birds. Can a bird fly? A typical answer is yes. Almost all birds can fly, and thus we think of the ability to fly as being an essential property of the class of birds, even though there are birds, such as ostriches, which cannot fly. In other words, an eagle is a better member of the class *bird* than an ostrich because an eagle is a more typical representative of birds.

Frame-based systems support class inheritance. The fundamental idea of inheritance is that attributes of the class-frame represent things that are *typically* true for all objects in the class. However, slots in the instance-frames can be filled with actual data uniquely specified for each instance.

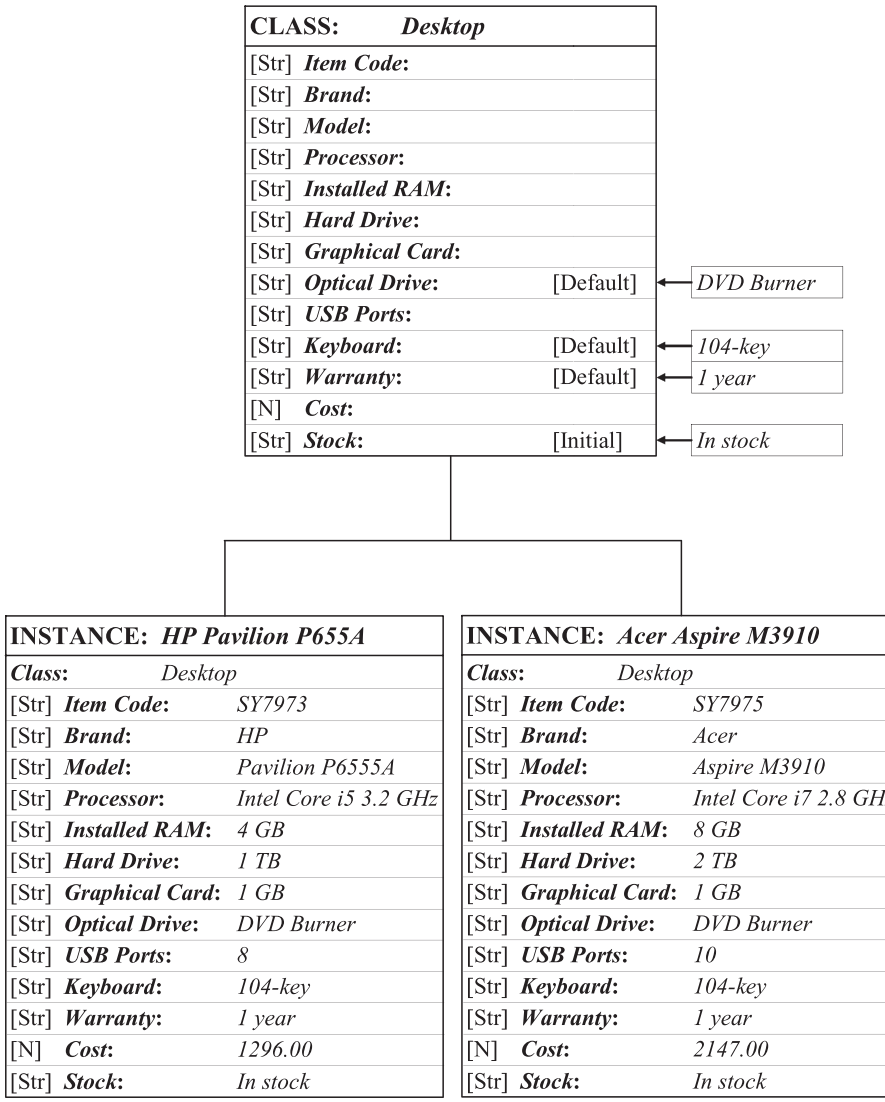

Figure 5.2 *Desktop* class and instances

Consider the simple frame structure represented in Figure 5.3. The class *Passenger car* has several attributes typical for all cars. This class is too heterogeneous to have any of the attributes filled in, even though we can place certain restrictions upon such attributes as *Engine type*, *Drivetrain type* and *Transmission type*. Note that these attributes are declared as *compound* [C]. Compound attributes can assume only one value from a group of symbolic values, for