Chapter 4

Frames and Inheritance

Representing knowledge in graph-like structures has a rich tradition in philosophy and psychology. At the end of the nineteenth century, the philosopher Charles S. Peirce used a graph-like notation for the representation of logical sentences. This approach to representing human knowledge has been further pursued since by many researchers, yielding explicit psychological models of human memory and intellectual behaviour. In particular the area of natural language processing has contributed much to the research on the representation of information in graph-like structures, there called *semantic nets* or *associative nets*; in fact, the earliest use of graph-based representations in computers was for machine translation. In the early 1960s, Ross Quillian for example used the semantic net formalism for representing meanings of English words in terms of associative links to other words, yielding a dictionary-like representation; he developed a program for finding relationships between words by traversing the net. Through this work, Ross Quillian has given a major impetus to the research on graph-based representations and their use in AI systems; he is generally credited with the development of the semantic net in its original form.

For handling more complicated problem domains and for dealing with more sophisticated forms of inference, the semantic net formalism as devised by Ross Quillian soon proved to be too limited. Much of the later work on semantic nets therefore has been directed towards more structured formalisms, again mostly for natural language processing. Semantic nets have seldom been used for building expert systems. Nevertheless, we shall briefly discuss some characteristics of the formalism, since the semantic net is often viewed as a precursor of the frame formalism, which is much more frequently applied within expert systems.

The basic idea underlying the notion of frames has already been posed at the beginning of this century by the psychologist Otto Selz. He considered human problem solving as the process of filling in the gaps of partially completed descriptions. The present notion of frames was introduced half-way the 1970s by Marvin Minsky for exerting semantic control in a pattern-recognition application. Since its introduction, however, the frame formalism has been employed in several other kinds of knowledge-based systems as well. The general idea of a frame-based system is that all knowledge concerning individuals or classes of individuals including their interrelationships, is stored in a complex entity of representation, called a *frame*. Instead of the term frame, the terms *unit*, *object*, and *concept* are also often used in literature. A set of frames representing the knowledge in a domain of interest, is organized hierarchically in what is called a *taxonomy*. Such a taxonomy forms the basis of a method of automated reasoning called *inheritance*. The frame formalism and its associated inheritance are the primary topics of this chapter. To prepare for a thorough treatment of these subjects, we shall first discuss the semantic net formalism briefly in Section 4.1.

4.1 Semantic Nets

A *semantic net* is usually depicted as a labelled directed graph, consisting of vertices and labelled arcs between vertices; such a graph is sometimes further restricted by requiring its being acyclic. Several disciplines have influenced the original idea of a semantic net as it was introduced in the 1960s: each discipline has brought its own interpretation of the vertices and arcs, and each discipline has adapted the notion of the semantic net in certain ways to arrive at a more structured formalism suitable for its own purposes. As a consequence, there is hardly any consensus as to what a semantic net is, nor is there any consensus as to what meaning should be ascribed to the basic elements of such a semantic net. Since the semantic net formalism is seldom used in expert systems, we will introduce it in a simple form, just to give the reader an idea about what graph-based representations are like.

4.1.1 Vertices and labelled arcs

We have mentioned before that a semantic net is usually depicted as a labelled, directed graph. Each vertex in the graphical representation of a semantic net is taken to represent a *concept*. The arcs of the graph represent binary relations between concepts. Let us give some informal examples of how knowledge is represented in a semantic net.

EXAMPLE 4.1

Consider the following statement concerning the human body:

> 'The heart is part of the cardiovascular system'

This statement comprises two concepts: the concept 'heart' and the concept 'cardiovascular system'. These concepts are related in the sense that the first concept, the 'heart', forms an anatomical part of the second concept, the 'cardiovascular system'. This knowledge is represented by means of the graph shown in Figure 4.1. The concepts are depicted by ellipses, labelled *heart* and *cardiovascular system*; the relation between the concepts is represented by means of an arc labelled *part-of*.

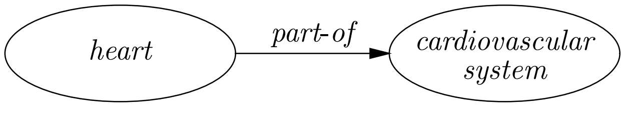


Figure 4.1: A graphical representation of a semantic net.

EXAMPLE 4.2

In the semantic net depicted in Figure 4.2, two different kinds of relation are used in representing information concerning the cardiovascular system of the human body: the 'part-of' relation and the 'is-a' relation.

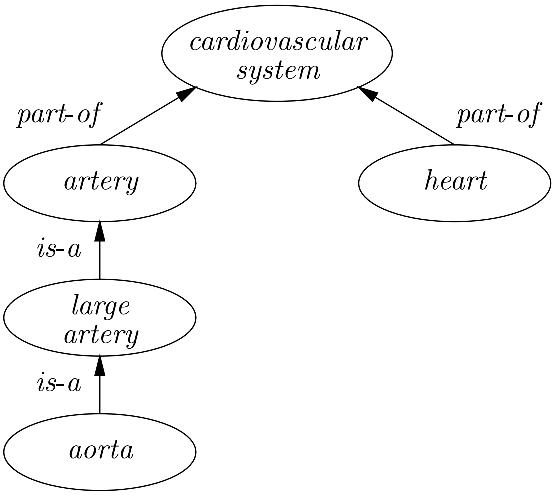


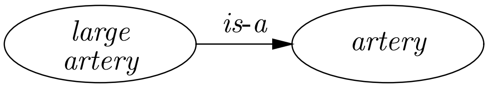
Figure 4.2: Some information concerning the cardiovascular system in a semantic net.

In the preceding example we encountered the is-a relation. This is a quite common relation between concepts. It reflects the two different senses in which a concept can be used; in this book, the term concept is used to denote either an *individual object* or a *class of objects*. The is-a relation may be used as follows:

- To express that a class of objects is a subclass of another class of objects, such as in the statement

  'A large artery is an artery'.

  This statement is depicted in a semantic net as follows:

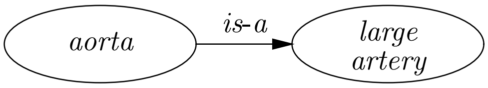
  

  In this case, the is-a part of the statement defines a *set inclusion relation*.

- To express that a specific object is a member of a certain class of objects, such as in the statement

  'The aorta is a large artery'.

  This statement is depicted as follows:

  Here, the is-a part of the statement defines a *membership relation* between an element and a set of elements.

In the early semantic net formalism, no explicit distinction was made between the different uses of the is-a relation, called the *is-a link* in semantic net terminology: individual objects

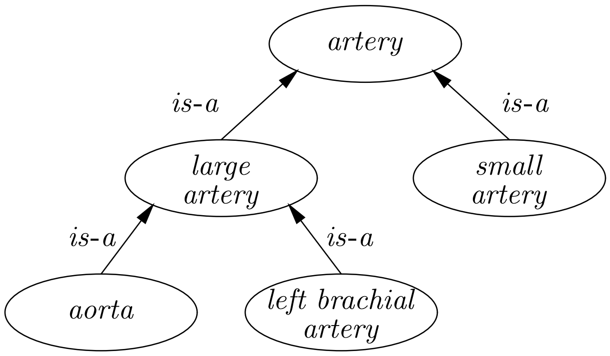


Figure 4.3: Counting specific objects.

and classes of objects were handled identically. The following example illustrates that this may lead to problems.

EXAMPLE 4.3

In Figure 4.3 some information concerning the arteries of the cardiovascular system is represented. Now consider a knowledge-based system comprising the information as shown. Suppose that we ask this system on how many specific arteries information is available. If we assume that the system 'knows' that information on individual objects is contained in the leaves of the net, the system will answer 3: the *aorta*, the *left brachial artery*, and the *small artery*. The system is not able to distinguish between the *small artery* representing a class of objects, and the individual objects *aorta* and *left brachial artery*.

The preceding example gives us a valid reason for distinguishing between different types of is-a link. From now on, we distinguish between the *subset-of link* and the *member-of link*.

Before proceeding, we define the notion of a semantic net more formally.

Definition 4.1 *A* semantic net *S is a labelled graph* $S = (V(S), A(S), \lambda)$ *where* $V(S)$ *is the set of* vertices *of* S *and* $A(S) \subseteq V(S) \times V(S)$ *is the set of* arcs *of* S*;* λ *is the* labelling function $\lambda : A(S) \to L(S)$ *associated with* S *where* $L(S)$ *is the set of* arc labels.

EXAMPLE 4.4

Consider the semantic net from Figure 4.1 once more. This net is defined by $S = (V(S), A(S), \lambda)$ where

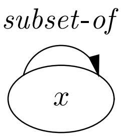
$$V(S) = \{\textit{heart}, \textit{cardiovasculari-system}\}$$
$$A(S) = \{(\textit{heart}, \textit{cardiovascular-system})\}$$
$$\lambda(\textit{heart}, \textit{cardiovascular-system}) = \textit{part-of}$$

In the preceding, we have defined a semantic net as a mere syntactical object: it has no meaning as yet. In order to assign a meaning to a semantic net, we have to define a proper interpretation for it. Note the analogy with a logical formula being a syntactical object and its interpretation (see the Sections 2.1 and 2.2).

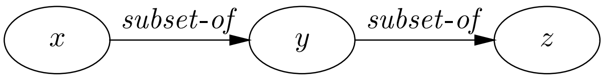
We start by giving an example of a possible interpretation for the subset-of link. We assign to the relation defined by the subset-of links the meaning of the usual set inclusion relation $\subseteq$. The relation $\subseteq$ has the following properties:

- *Reflexivity*: for each X, we have $X \subseteq X$.
- *Anti-symmetry*: for each X, Y, if $X \subseteq Y$ and $Y \subseteq X$, then $X = Y$.
- *Transitivity*: for each X, Y, Z, if $X \subseteq Y$ and $Y \subseteq Z$, then $X \subseteq Z$.

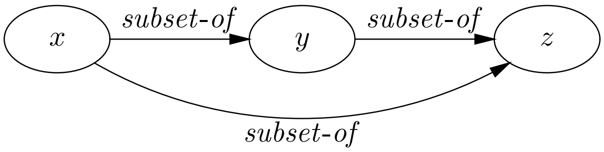
Any binary relation having these properties is called a *partial order*. With each vertex $x \in V(S)$ taking part in a subset-of link — note that from this observation we have that the vertex represents a class of objects — we associate a set of elements $I(x)$ from a (semantic) domain of discourse D, that is, $I(x) \subseteq D$. We now may interpret $\lambda(x, y) = \textit{subset-of}$ as $I(x) \subseteq I(y)$. From the reflexivity of the set inclusion relation we have that we may add to or delete from a semantic net arcs of the form (x, x) for which $\lambda(x, x) = \textit{subset-of}$:



This is called a *trivial cycle*. From the transitivity of the set inclusion relation it furthermore follows that if we have $\lambda(x, y) = \textit{subset-of}$ and $\lambda(y, z) = \textit{subset-of}$ in a specific net, then we may add $\lambda(x, z) = \textit{subset-of}$ to the net without changing its meaning: the two nets



and



therefore express the same information.

Similar to the interpretation of the subset-of link, vertices $u \in V(S)$ taking part in the left-hand side of a member-of link — from this we have that u represents an individual object — have associated an element $I(u)$ from the domain D, that is, $I(u) \in D$. The relation defined by the member-of links now is assigned the meaning of the usual membership relation $\in$; that is, we interpret $\lambda(u, v) = \textit{member-of}$ as $I(u) \in I(v)$.

It will be evident that for a semantic net to have a neat semantics we have to define a proper interpretation for each type of link used in the net. Especially if no restrictions have been imposed on the types of links, this will be a cumbersome endeavour. It is no wonder therefore that since the introduction of the semantic net idea, researchers have sought after more restricted special-purpose net formalisms. Here, we do not pursue the subject of the declarative semantics of a semantic net any further.

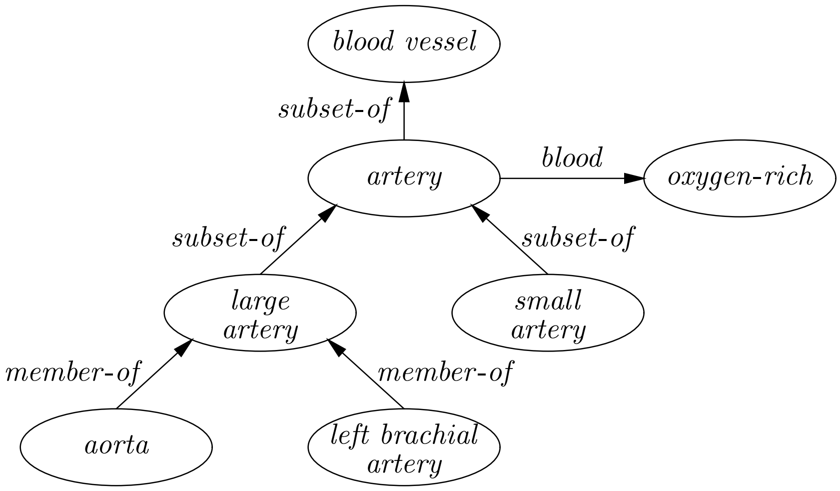


Figure 4.4: Inheritance.

4.1.2 Inheritance

The subset-of and member-of links of a semantic net may be exploited to derive new information from it, that is, they may be used as the basis for an inference engine. We illustrate the use of these links in reasoning with the help of an example.

EXAMPLE 4.5

Consider the semantic net shown in Figure 4.4. Among others, the following two statements are represented in the net:

'A large artery is an artery'
'An artery is a blood vessel'

From these two statements we may derive the statement

'A large artery is a blood vessel'

exploiting the transitivity of the relation defined by the subset-of links. Furthermore, the statement

'The aorta is an artery'

can be derived from the net using the semantics of both the member-of and subset-of link.

Exploiting the semantics of the member-of and subset-of links in the manner discussed informally in the preceding example forms the basis of a reasoning mechanism called (*property*) *inheritance*: a concept *inherits* the properties of the concepts 'higher' in the net through these member-of and subset-of links. The general idea is demonstrated in the following example.

EXAMPLE 4.6

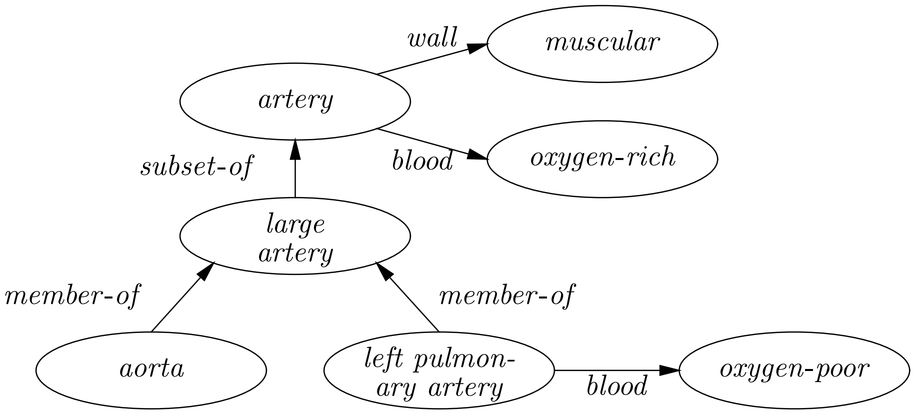


Figure 4.5: An exception to a general property.

Consider the semantic net shown in Figure 4.4 once more. Using property inheritance, we may derive from it the following statement:

> 'The aorta contains oxygen-rich blood'

The concept 'aorta' has inherited the property 'contains oxygen-rich blood' from the concept 'artery' which is found higher in the net.

In Section 4.3 we shall discuss the principle of inheritance more formally.

The semantic net is a natural formalism for expressing knowledge in which the basic concepts are organized in a hierarchical manner. Several problems, however, arise from the rigidity of the principle of inheritance as introduced above. We give two examples to illustrate some of these problems.

EXAMPLE 4.7

Consider Figure 4.5, again showing some information concerning the arteries. Among other information, it has been specified that arteries in general have muscular walls and transport oxygen-rich blood. An exception to the latter property of arteries is for example the left pulmonary artery which is an artery containing oxygen-poor blood.

Using the member-of and subset-of links shown in the net the aorta inherits the properties of the arteries: the aorta has a muscular wall and transports oxygen-rich blood. Using a similar argument, the left pulmonary artery inherits these two properties as well. The left pulmonary artery, however, transports oxygen-poor blood! So, the property that arteries transport oxygen-rich blood should not be inherited by the left pulmonary artery. When employing the principle of inheritance discussed so far, the inheritance of this property cannot be prevented. A possible solution to this problem is to store the information that an artery contains oxygen-rich blood explicitly with each artery for which this property holds. This is shown in Figure 4.6. A major drawback of this solution is that the general property has been lost.

EXAMPLE 4.8

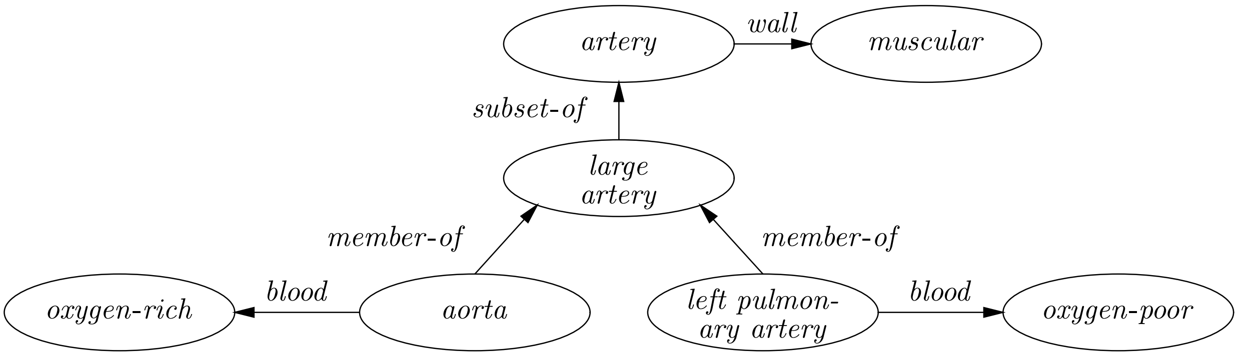


Figure 4.6: Loss of a general property.

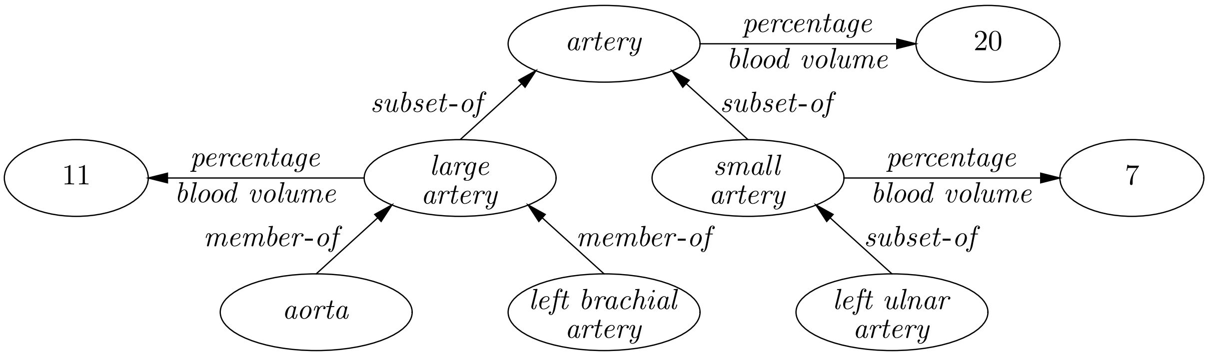


Figure 4.7: Inheritance of properties relevant to a class as a whole.

In the foregoing examples we have discussed properties which are relevant to individual objects. In the semantic net shown in Figure 4.7 some information has been stored that is relevant to a class of objects as a whole and not to the individuals belonging to it. For example, in the net we have represented the information that the large arteries together contain approximately 11% of all the blood the human body contains. This information is only relevant to the class as a whole and not to a single large artery. So, this property should not be inherited by the aorta and the left brachial artery. Furthermore, the information that all arteries together contain 20% of the total blood volume should not be inherited by the class of the small arteries: the latter class only contains 7% of the total blood volume. Again, inheritance cannot be prevented.

It has been mentioned before that the semantic net formalism has undergone many changes since its introduction. The resulting formalisms on the one hand are more restricted: only a limited number of predefined link-types is allowed, each having a clear semantics. On the other hand, many new features have been added to the formalism. In particular the principle of inheritance has been revised in order to make inheritance of properties more flexible. Furthermore, some facilities for representing procedural knowledge have been added to the semantic net. Many of these extensions bear a close resemblance to features of frames. Therefore, we shall not discuss these features in relation to semantic nets here, but only in relation to the frame formalism in Section 4.2.

4.1.3 The extended semantic net

Before we turn our attention to knowledge representation in frames, we conclude this section with a discussion of an interesting type of semantic net: the *extended semantic net*. The extended semantic net was developed by A. Deliyanni and R.A. Kowalski as an alternative representation formalism for the clausal form of logic with a restriction to binary predicate symbols. It should be noted that the restriction to binary predicates is not an essential one: any atom containing an n-placed predicate symbol can be replaced by a conjunction of atoms involving binary predicates only. If $n > 2$, $n + 1$ new predicates are needed to represent the original information; if $n = 1$, only a single new predicate is required.

EXAMPLE 4.9

Consider the three-placed predicate symbol *Bloodpressure*, which is supposed to have the following intended meaning:

$$\begin{aligned} \mathit{Bloodpressure}(x, y, z) = \ & \text{`the mean blood pressure in } x \text{ lies between} \\ & y \text{ mmHg and } z \text{ mmHg'} \end{aligned}$$

The clause

$$\mathit{Bloodpressure}(\mathit{artery}, 40, 80) \leftarrow$$

for example, can be replaced by the following four clauses

$$\begin{aligned} & \mathit{Info}(\mathit{fact}, \mathit{bloodpressure}) \leftarrow \\ & \mathit{Subject}(\mathit{fact}, \mathit{artery}) \leftarrow \\ & \mathit{Lowerbound}(\mathit{fact}, 40) \leftarrow \\ & \mathit{Upperbound}(\mathit{fact}, 80) \leftarrow \end{aligned}$$

in which only binary predicate symbols have been used to express the same information. We have introduced the new constants *fact* and *bloodpressure*; the new binary predicate symbols should be read as

$$\begin{aligned} \mathit{Info}(w, \mathit{bloodpressure}) &= \text{`}w \text{ is information about blood pressure'} \\ \mathit{Subject}(w, x) &= \text{`}x \text{ is the subject of the information } w\text{'} \\ \mathit{Lowerbound}(w, y) &= \text{`}y \text{ is the lower bound of } w\text{'} \\ \mathit{Upperbound}(w, z) &= \text{`}z \text{ is the upper bound of } w\text{'} \end{aligned}$$

EXAMPLE 4.10

Consider the unary predicate symbol *Artery* with the following intended meaning:

$$\mathit{Artery}(x) = \text{`}x \text{ is an artery'}$$

The clause

$$\mathit{Artery}(\mathit{aorta}) \leftarrow$$

may be replaced by the clause

$$\mathit{Isa}(\mathit{aorta}, \mathit{artery}) \leftarrow$$

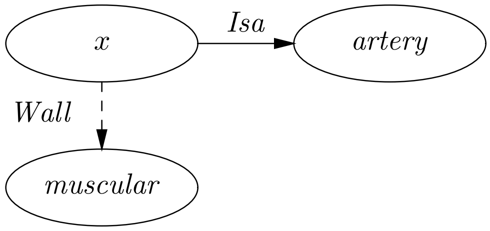


Figure 4.8: The extended semantic net for the clause $Wall(x, muscular) \leftarrow Isa(x, artery)$.

We have mentioned before that the extended semantic net provides an alternative syntax for the clausal form of logic: the arguments to the predicate symbols occurring in a set of clauses are taken as the vertices, and the binary predicate symbols themselves are taken as the labels of the arcs of a directed graph. The direction of the arc expresses the order of the arguments to the predicate symbol. The conclusions and conditions of a clause are represented by different types of arcs: conditions are denoted by pulled arcs and conclusions are indicated by dashed arcs.

EXAMPLE 4.11

The extended semantic net shown in Figure 4.8 represents the clause

$$Wall(x, muscular) \leftarrow Isa(x, artery)$$

A particular constant may occur in more than one clause. If we represent all occurrences of a constant by means of a single vertex, then it is not always apparent in the representation discussed above to which clauses a particular vertex belongs. This is why an extended semantic net representing a set of clauses is divided into a number of subnets, each representing a single clause.

EXAMPLE 4.12

The following set of clauses

$$\{\, Wall(x, muscular) \leftarrow Isa(x, artery),\ \ Isa(y, artery) \leftarrow Isa(y, large\text{-}artery)\}$$

has been represented in Figure 4.9. The net is partitioned into two subnets: for each clause from the given clause set we have one corresponding subnet. Note that if a similar situation arises concerning a variable, we can simply rename the variables to avoid the problem.

A pleasant consequence of the syntactical correspondence between the clausal form of logic and the extended semantic net is that the inference rules that are defined for the clausal form of logic can be applied for manipulation of arcs and vertices in an extended semantic net.

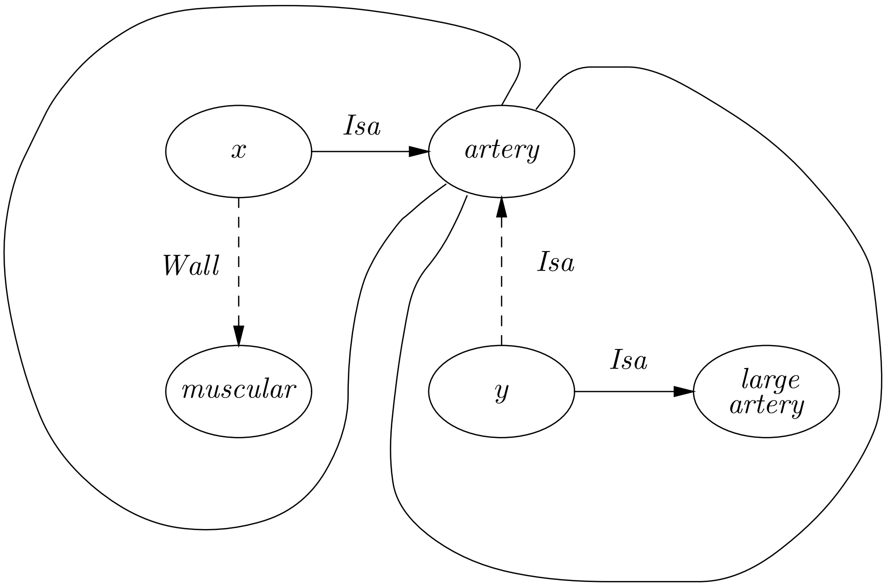


Figure 4.9: Partition of an extended semantic net into subnets.

4.2 Frames and single inheritance

In a frame-based system all knowledge relevant to a concept is stored in a complex entity of representation called a *frame*. Frames provide a formalism for explicitly grouping all knowledge concerning the properties of individual objects or classes of objects. Within a frame, part of the properties is specified as reference information to other, more general frames. This reference information is represented by means of is-a links which are quite similar in concept to the is-a links in a semantic net. This way, the knowledge in a domain of interest is organized hierarchically in what is called a *frame hierarchy*, *frame taxonomy*, or *taxonomy* for short. A taxonomy often is depicted graphically as a directed, acyclic graph once more bearing a close resemblance to a semantic net. However, in the graph representation of a frame taxonomy only the is-a links are shown (as was true for the semantic net, trivial cycles are not shown in the graph since they do not represent additional information); the knowledge concerning an individual object or a class of objects itself is part of the internal structure of the vertices of the graph. In contrast with the semantic net, in a frame representation different components are distinguished, all having a special status allowing them to be treated explicitly as such. For example, the is-a links in a frame taxonomy are represented and treated in a way different from other components.

4.2.1 Tree-shaped frame taxonomies

As has been mentioned above, frames are organized in a taxonomy in which the vertices represent frames and in which every arc denotes an is-a link between two frames. In the frame formalism which will be used in this book, two types of frames are distinguished:

- *class frames*, or *generic frames*, which represent knowledge concerning classes of objects;
- *instance frames*, which represent knowledge concerning individual objects.

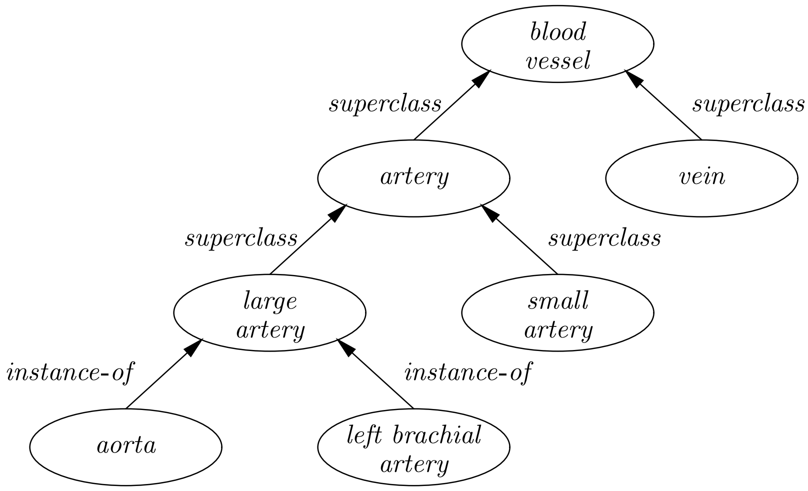


Figure 4.10: A tree-shaped taxonomy.

Class frames have much in common with the record datatype as, for example, provided in the Pascal programming language, and instances are similar to filled-in record variables.

Since there are two types of frames, we also distinguish two types of is-a links by means of which a frame indicates its relative position in the frame taxonomy:

- an *instance-of link*, which is an is-a link between an instance frame and a class frame;
- a *superclass link*, which is an is-a link between two class frames defining a partial order on the set of class frames in a frame taxonomy.

These is-a links are similar in meaning to the member-of and subset-of links, respectively, distinguished for semantic nets in the previous section. Their formal meaning will be discussed in the following.

In the present section, we consider frame taxonomies that can be represented as trees. Section 4.3 discusses the more general graph-shaped taxonomies. An example of a tree-shaped taxonomy is shown in Figure 4.10. In this figure, a frame is represented as an ellipse; the internal structure of a frame is not shown. In a tree-shaped frame taxonomy capturing knowledge concerning a given domain, the root of the tree represents the most general description of the domain: the other frames represent descriptions of concepts that are more specific. The descendants of a certain frame in the taxonomy therefore are often called *specializations* of that frame. The ancestors of a frame in the taxonomy are called its *generalizations*. When we restrict the discussion to classes of objects only, specializations are generally called *subclasses* and generalizations are called *superclasses*. We shall use the terms *superframe* and *subframes* for the parent and children of a given frame, respectively.

EXAMPLE 4.13

Consider the frame taxonomy shown in Figure 4.10 once more. The vertex representing the frame with the name *blood vessel* is the father (and therefore ancestor) of the vertices representing the frames *artery* and *vein*. So, the frame with the name *blood vessel* is the generalization of the frames *artery* and *vein*; it equally is a generalization of the frame with the name *aorta*. The frame with the name *small artery* is a specialization of the *artery* frame.

A frame representing an individual object cannot be specialized any further. Therefore, in a tree-shaped frame taxonomy an instance is always a leaf of the tree.

EXAMPLE 4.14

Consider Figure 4.10 once more. The frames with the names *aorta* and *left brachial artery* cannot be specialized any further, since these frames represent individual objects and therefore are instances. Except for the vertices representing these two frames, the tree has another two leaves: the frames *small artery* and *vein*. These two frames are generic: the descriptions given in these frames may be further specialized. Note that although an instance is always a leaf of the tree, not every leaf is an instance.

We shall now turn our attention to the internal structure of a frame. We assume that each frame in a taxonomy has a unique name. The information specific to the concept represented by a frame is laid down in so-called *attributes* or *slots*. So, attributes offer a means for representing the properties of individual objects or classes of objects. In the following definition, we shall present a syntax of a language for the representation of frames; from then on we shall be able to be more precise in discussing frames and their formal meaning.

Definition 4.2 *A* frame *is a statement having the following form:*

$$
\begin{array}{lll}
\langle\text{frame}\rangle & ::= & \langle\text{class}\rangle \mid \langle\text{instance}\rangle \\[1ex]
\langle\text{class}\rangle & ::= & \textbf{class } \langle\text{class-name}\rangle \textbf{ is} \\
 & & \quad \textbf{superclass } \langle\text{super-specification}\rangle; \\
 & & \quad \langle\text{class-attributes}\rangle \\
 & & \textbf{end} \\[1ex]
\langle\text{instance}\rangle & ::= & \textbf{instance } \langle\text{instance-name}\rangle \textbf{ is} \\
 & & \quad \textbf{instance-of } \langle\text{super-specification}\rangle; \\
 & & \quad \langle\text{instance-attributes}\rangle \\
 & & \textbf{end} \\[1ex]
\langle\text{super-specification}\rangle & ::= & \langle\text{class-name}\rangle \mid \textbf{nil} \\[1ex]
\langle\text{class-attributes}\rangle & ::= & \langle\text{declaration}\rangle\ \{;\langle\text{declaration}\rangle\}^* \mid \langle\text{empty}\rangle \\[1ex]
\langle\text{instance-attributes}\rangle & ::= & \langle\text{attribute-value-pair}\rangle\ \{;\langle\text{attribute-value-pair}\rangle\}^* \mid \langle\text{empty}\rangle \\[1ex]
\langle\text{declaration}\rangle & ::= & \langle\text{attribute-type-pair}\rangle \mid \langle\text{attribute-value-pair}\rangle \\[1ex]
\langle\text{attribute-type-pair}\rangle & ::= & \langle\text{attribute-name}\rangle : \langle\text{type}\rangle \\[1ex]
\langle\text{attribute-value-pair}\rangle & ::= & \langle\text{attribute-name}\rangle = \langle\text{value}\rangle
\end{array}
$$

$\langle\text{type}\rangle$ ::= **int** | **real** | **string** | $\langle\text{set}\rangle$ | $\langle\text{class-name}\rangle$

$\langle\text{value}\rangle$ ::= $\langle\text{elementary-constant}\rangle$ | $\langle\text{instance-name}\rangle$

$\langle\text{empty}\rangle$ ::=

A ⟨super-specification⟩ *equal to the special symbol* **nil** *is used to indicate that the frame concerned is the root of the tree-shaped taxonomy. As a type, a* ⟨set⟩ *consists of elementary constants and instance names, separated by comma's and enclosed in curly brackets. An elementary constant is either a real or integer constant, or a string of non-blank characters, that is, an instance of one of the predefined (or standard) classes* **real***,* **int***, and* **string***. The* ⟨instance-name⟩ *value of an attribute refers to a uniquely defined instance in the taxonomy.*

For the moment we assume that an attribute-type or attribute-value pair for an attribute only occurs once in a frame taxonomy. Later on we shall drop this restriction. In the preceding definition, we have stated for ease of exposition that a class frame is the root of a tree-shaped taxonomy if it has a super-specification equal to **nil**, where it actually is a subclass of the most general class **nil**. This more accurate interpretation of the symbol **nil**, however, is only important in frame taxonomies in which more than one most general class frame not equal to **nil** occurs; if we did not consider **nil** as the most general class, then the graph representation of such taxonomy would be a forest of trees instead of just a tree.

As can be seen in the preceding definition, the definition of an instance frame is composed of the specification of the class to which the instance belongs followed by a collection of attribute-value pairs. Together they give a description of an individual concept in the domain of discourse. Let us give an example of such an instance.

EXAMPLE 4.15

We consider the left brachial artery which is one of the arteries in the human body. It is known that the left brachial artery has an approximate diameter of 0.4 cm, that it is localized in the upper arm, and that it contains oxygen-rich blood. All this information is captured by the following instance frame:

instance *left-brachial-artery* **is**
  instance-of *artery*;
  diameter = 0.4;
  location = *arm*;
  blood = *oxygen-rich*
end

We have used the attributes *diameter*, *location*, and *blood* for the representation of the mentioned properties of the individual concept 'left brachial artery'. Note that all three attributes have been assigned actual values. The values 0.4 and *oxygen-rich* are assumed to be elementary constants. The value *arm* of the attribute *location* is an instance of the class frame *limb*:

instance *arm* **is**
  superclass *limb*;
  position = *superior*
end

The value *superior* again is an elementary constant.

The information specified in the attribute parts of instance frames has to accord with the following rules. All attributes occurring in the instances of a class frame must have been declared in the attribute part of that class or in one of its generalizations; the values which have been filled in for the attributes in the instance must be of the appropriate attribute type as defined by the classes in the taxonomy. Note that these rules provide part of the meaning of the instance-of link.

EXAMPLE 4.16

Consider the instance *left-brachial-artery* from the preceding example once more. The class to which *left-brachial-artery* belongs is defined as follows:

> **class** *artery* **is**
>   **superclass** *blood-vessel*;
>   *location* : $\{arm, head, leg, trunk\}$
> **end**

This class frame provides a type declaration for the attribute *location*: it indicates that the *location* attribute is only allowed to take a value from the set $\{arm, head, leg, trunk\}$. Note that the value *arm* given for the *location* attribute in the *left-brachial-artery* instance is indeed of the correct type. Not all attributes occurring in the instance have been declared in the class frame *artery*; so, the *diameter* and *blood* attributes must have been declared in some of the generalizations of the *artery* class. In the previous example we have mentioned that the instance frame *arm* belongs to the class *limb*. This class frame for example is defined as follows:

> **class** *limb* **is**
>   **superclass nil**;
>   *position* : $\{inferior, superior\}$
> **end**

From the superclass specification **nil** we have that this class is the root of a tree-shaped taxonomy.

In the preceding example, the class frames we considered had attribute-type pairs only in their declaration part. However, the syntax definition indicates that also attribute-value pairs are allowed in the declaration part of a class frame. The following example illustrates this idea.

EXAMPLE 4.17

Consider the previous examples once more. Since most arteries contain oxygen-rich blood, there is no purpose in repeating this information for all individual arteries separately. In this case, it appears to be convenient to fix the value *oxygen-rich* for the attribute *blood* in advance in the class frame *artery*. We then obtain the following alternative definition for the *artery* frame:

> **class** *artery* **is**
>   **superclass** *blood-vessel*;
>   *location* : {*arm*,*head*,*leg*,*trunk*};
>   *blood* = *oxygen-rich*
> **end**

The instance frame *left-brachial-artery* may now be simplified to

> **instance** *left-brachial-artery* **is**
>   **instance-of** *artery*;
>   *diameter* = 0.4;
>   *location* = *arm*
> **end**

without affecting the intended meaning.

Although only informally, we have now fully described the meaning of the instance-of link. We turn our attention to the superclass link. Recall that we have distinguished two different types of attribute information in a frame taxonomy: information about attribute types and information about attribute values. Accordingly, in assigning a meaning to the superclass link we have to distinguish between these different types of attribute information. First of all, the superclass link defines a partial order on the class frames in a taxonomy and may be applied for reasoning about attribute values much in the same way we have seen for semantic nets. Secondly, however, the superclass link may be viewed as defining a relation which restricts the semantic contents of the frame taxonomy as we have shown in the preceding example. This is essentially a higher-order relation. These two different ways of interpreting the superclass link are best treated separately. We shall therefore first study the meaning of attribute values in a tree-shaped taxonomy and show how it is possible to reason about such attribute values. From now on, we shall, for ease of exposition, completely disregard the fact that classes may contain type information until Section 4.2.4 in which we shall return to attribute types.

We have mentioned before that the semantic net formalism we have briefly discussed in Section 4.1 may be viewed as a precursor of the frame formalism. We take a closer look at the relationship between the two formalisms. This relationship can be examined more readily if we depict a frame taxonomy in a graph as we have done with the semantic net. The following example shows the general idea.

EXAMPLE 4.18

Figure 4.11 shows four frames in a tree-shaped taxonomy. The frames are represented as boxes; the internal structure of each of the frames is depicted. The arcs in the graph represent the instance-of and superclass links between the frames. Note that the frames themselves already indicate their position in the taxonomy explicitly; the graphical representation of the taxonomy therefore contains redundant information. From a graph representing a frame taxonomy we can easily derive an equivalent semantic net. Figure 4.12 shows the semantic net equivalent to the taxonomy depicted in Figure 4.11. Note that although the corresponding semantic net essentially comprises the same information as the original frame taxonomy, the apparent modularity of the taxonomy has been lost.

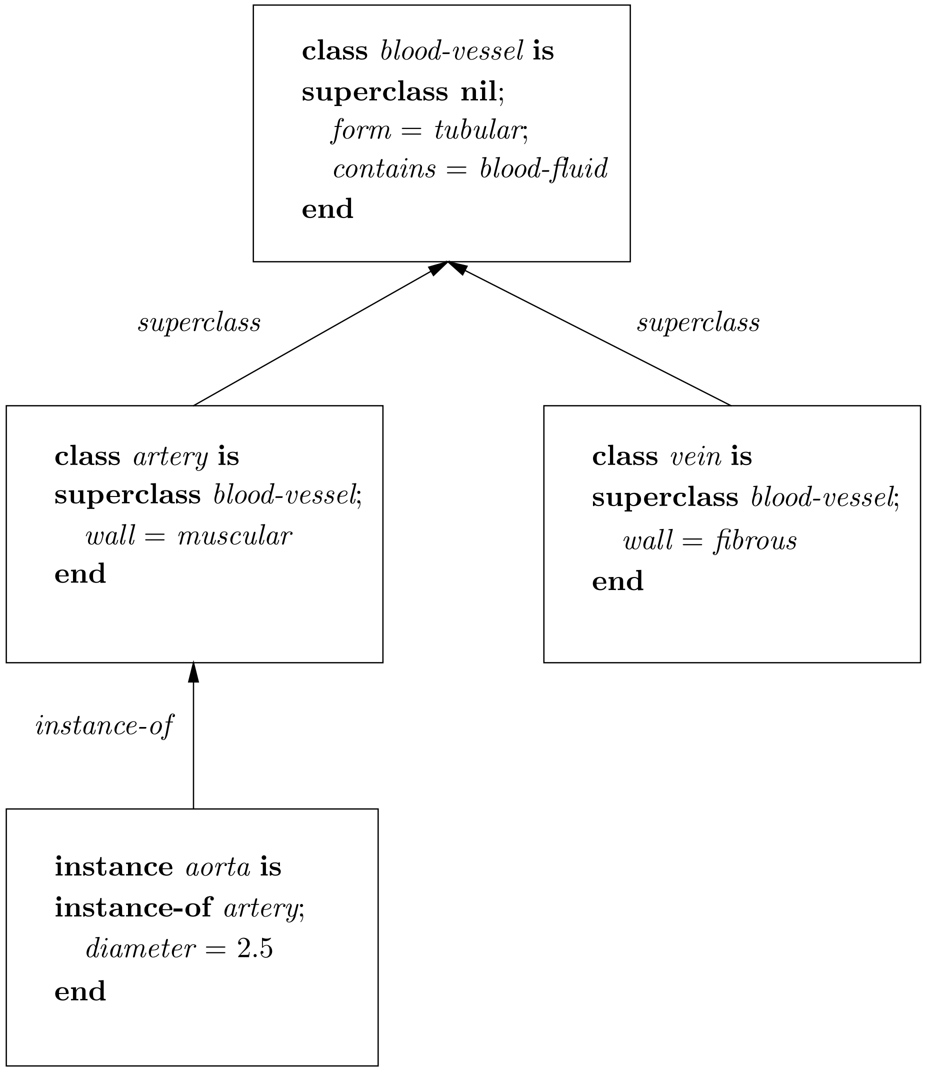


Figure 4.11: A tree-shaped taxonomy showing the internal structure of the frames.

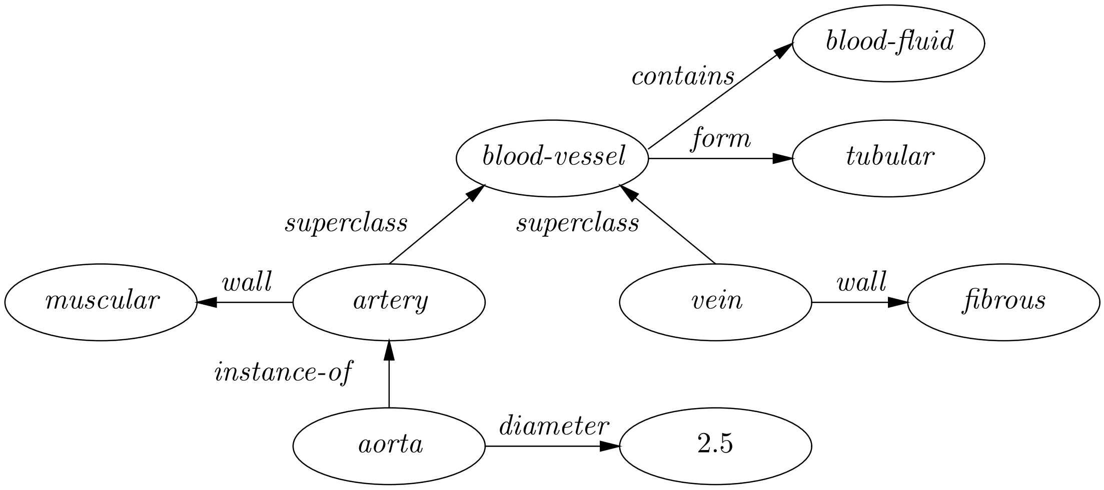


Figure 4.12: The semantic net corresponding to the taxonomy shown in Figure 4.11.

Based on the frame formalism defined above we shall discuss the meaning that can be associated with the frame formalism. The discussion takes the following example for a starting point.

EXAMPLE 4.19

Suppose that we want to represent in a frame the following information concerning the vascular system: the aorta is an artery having a diameter of 2.5 cm. Using our frame formalism, this information may be represented as follows:

> **instance** *aorta* **is**
>   **instance-of** *artery*;
>   *diameter* = 2.5
> **end**

The information that an artery is a blood vessel having a muscular wall is represented in the following class frame:

> **class** *artery* **is**
>   **superclass** *blood-vessel*;
>   *wall* = *muscular*
> **end**

To conclude, the following class frame shows that a blood vessel is tubular in form and contains blood:

> **class** *blood-vessel* **is**
>   **superclass nil**;
>   *form* = *tubular*;
>   *contains* = *blood-fluid*
> **end**

The last frame furthermore indicates that the *blood-vessel* frame is the root of the taxonomy concerned.

The information that is specified in a frame taxonomy can also be expressed in first-order predicate logic, roughly by complying with the following directions:

- take the names of the instances as constants;
- take the names of the class frames as unary predicate symbols;
- translate an instance-of link into a predicate symbol having a constant for an argument;
- translate a superclass link into a logical implication;
- take the attribute names as unary function symbols;

- translate an attribute-value pair into an equality between a function term and a constant.

EXAMPLE 4.20

Assuming a suitable interpretation, the following formulas represent the same information as the frames in the foregoing example do:

$$\begin{array}{l} \mathit{artery}(\mathit{aorta}) \\ \mathit{diameter}(\mathit{aorta}) = 2.5 \end{array}$$

$$\begin{array}{l} \forall x(\mathit{artery}(x) \to \mathit{blood\text{-}vessel}(x)) \\ \forall x(\mathit{artery}(x) \to \mathit{wall}(x) = \mathit{muscular}) \end{array}$$

$$\begin{array}{l} \forall x(\mathit{blood\text{-}vessel}(x) \to \mathit{form}(x) = \mathit{tubular}) \\ \forall x(\mathit{blood\text{-}vessel}(x) \to \mathit{contains}(x) = \mathit{blood\text{-}fluid}) \end{array}$$

The semantics of first-order predicate logic may now be exploited to define a semantics for the frame formalism. Under the assumption that each attribute only occurs once in the taxonomy, we may ascribe a meaning based on first-order predicate logic to the set of frames of this taxonomy using the following general translation scheme:

$$\begin{array}{lcl} \textbf{class } C \textbf{ is} & & \\ \quad \textbf{superclass } S; & & \forall x(C(x) \to S(x)) \\ \quad a_1 = b_1; & \Rightarrow & \forall x(C(x) \to a_1(x) = b_1) \\ \quad \vdots & & \vdots \\ \quad a_n = b_n & & \forall x(C(x) \to a_n(x) = b_n) \\ \textbf{end} & & \end{array}$$

$$\begin{array}{lcl} \textbf{instance } I \textbf{ is} & & \\ \quad \textbf{instance-of } C; & & C(I) \\ \quad a_1 = b_1; & \Rightarrow & a_1(I) = b_1 \\ \quad \vdots & & \vdots \\ \quad a_n = b_n & & a_n(I) = b_n \\ \textbf{end} & & \end{array}$$

Under the mentioned assumption we have that there always exists an interpretation I of the thus obtained logical formulas which is a model. In the next section we shall study the case in which the restriction that an attribute-value pair for an attribute occurs only once in a taxonomy has been dropped.

We conclude this section with a discussion of the inference method associated with the frame formalism. We start by examining the derivations that can be made from the corresponding logical formulas by means of a sound and complete collection of inference rules.

EXAMPLE 4.21

Consider the formulas given in the previous example once more. From the formulas

$$artery(aorta)$$
$$\forall x(artery(x) \to wall(x) = muscular)$$

we can derive the formula

$$wall(aorta) = muscular$$

using modus ponens. Similarly, from the set of formulas in the preceding example the following formula can be derived:

$$blood\text{-}vessel(aorta)$$

When closely examining these derivations, we see that the information holding for arteries in general is explicitly said to hold for the aorta in particular: since we know that the aorta is an artery, the aorta *inherits* this information from the arteries. Similarly, the aorta inherits the information specific to blood vessels.

In the foregoing example we have demonstrated the reasoning behaviour with logical formulas representing the information stored in a given frame taxonomy. This reasoning behaviour is modelled in an inference method for frames called *single inheritance*. In case of a tree-shaped taxonomy, we speak of *single* inheritance to stress the fact that each frame has at most one superframe. In contrast, the inference method associated with more general, graph-shaped taxonomies is called *multiple* inheritance; we shall discuss multiple inheritance in Section 4.3. Informally speaking, in single inheritance all information that holds for a particular frame is determined by traversing the taxonomy from the frame itself to the root of the taxonomy, that is, the most general frame, and successively collecting the attributes with their associated value that are found in the encountered frames. This may be viewed as exploiting the transitivity property of the superclass relation. This procedure terminates as soon as the information in the root of the taxonomy has been processed. The function shown below describes the recursive inference procedure:

```
function Inherit(frame, attribute-value-pairs)
   if frame = nil then
     return(attribute-value-pairs)
   end;
   attribute-value-pairs ← attribute-value-pairs ∪ AttributePart(frame);
   return(Inherit(Superframe(frame), attribute-value-pairs))
end
```

The parameters `frame` and `attribute-value-pairs` take as values a frame name and a collection of attribute-value pairs, respectively. If the parameter `frame` equals **nil**, then either the taxonomy is empty or the root of the taxonomy has been reached: in both cases all attribute-value pairs holding for the frame concerned have been collected in the second argument `attribute-value-pairs`. If the parameter `frame` differs from the value **nil**, then all attribute-value pairs specified in the frame `frame` are extracted from it using the function `AttributePart` and added to `attribute-value-pairs`. The information holding for the superframe of the given frame `frame` is subsequently determined by means of a recursive call to the `Inherit` function.

EXAMPLE 4.22

Consider Figure 4.11 again. In the instance frame with the name *aorta* the attribute-value pair

$$diameter = 2.5$$

has been specified. Using the `Inherit` function described in the foregoing, the instance inherits the attribute-value pair

$$wall = muscular$$

from its superframe *artery*. From the superframe *blood-vessel* of the frame *artery*, the instance inherits the following two attribute-value pairs:

contains = *blood-fluid*
form = *tubular*

4.2.2 Exceptions

In the previous section we have introduced a semantics for the frame formalism based on first-order predicate logic. To this end, we assumed that attributes occurred only once in a frame taxonomy. This assumption, however, renders the frame formalism not flexible enough for coping with all practical applications. In this section we therefore abandon this rather restrictive assumption and investigate the problems that arise from doing so; we will assume, however, that in a given frame an attribute can only take one value at a time. Allowing attributes to occur more than once in a frame taxonomy increases the expressive power of the formalism: it has become possible to state *exceptions* to information that holds in general but for some special cases. The following example shows the way an exception may be represented; it furthermore discusses the consequence of the introduction of exceptions into the formalism with respect to its semantics.

EXAMPLE 4.23

We have said in the preceding section that most arteries contain oxygen-rich blood. The following class frame captures this knowledge:

class *artery* **is**
  superclass *blood-vessel*;
  blood = *oxygen-rich*
end

However, it is known that the left and right pulmonary arteries are exceptions to this property of arteries: the pulmonary arteries have almost all properties arteries have but, opposed to arteries in general, they transport oxygen-poor blood. Restricting the discussion to the left pulmonary artery only, this information has been specified in the following instance frame:

instance *left-pulmonary-artery* **is**
  instance-of *artery*;
  blood = *oxygen-poor*
end

We now have expressed that the value *oxygen-poor* of the attribute *blood* is an exception to the value *oxygen-rich* of the attribute *blood* that has been specified in the superframe *artery* of the instance: informally speaking, the 'general' value has been surpassed. Note that the attribute *blood* is no longer unique in the taxonomy.

Applying the general translation scheme for converting these two frames into formulas in first-order predicate logic, we obtain the following set of formulas:

$$\begin{array}{l} \mathit{artery}(\mathit{left\text{-}pulmonary\text{-}artery}) \\ \mathit{blood}(\mathit{left\text{-}pulmonary\text{-}artery}) = \mathit{oxygen\text{-}poor} \\ \\ \forall x(\mathit{artery}(x) \rightarrow \mathit{blood\text{-}vessel}(x)) \\ \forall x(\mathit{artery}(x) \rightarrow \mathit{blood}(x) = \mathit{oxygen\text{-}rich}) \end{array}$$

This set of logical formulas is inconsistent, since by means of modus ponens we can derive the following logical consequences:

$$\begin{array}{l} \mathit{blood}(\mathit{left\text{-}pulmonary\text{-}artery}) = \mathit{oxygen\text{-}poor} \\ \mathit{blood}(\mathit{left\text{-}pulmonary\text{-}artery}) = \mathit{oxygen\text{-}rich} \end{array}$$

The inconsistency now follows from the equality axioms (these are assumed to be implicitly present). We assume that the unique name assumption holds, that is, symbols (function symbols, predicate symbols, and constants) with different names are assumed to be different. Now observe that in any model for the logical formulas shown above the constants *oxygen-rich* and *oxygen-poor* are equal. This, however, contradicts the unique name assumption.

In the foregoing example we have demonstrated that in the frame formalism exceptions are represented by locally surpassing attribute values. Furthermore, it has been shown that in case we allow multiple occurrences of attributes the translation of the frame formalism into first-order predicate logic may render an inconsistent set of formulas; it is not possible to fully capture the notion of exceptions by standard first-order predicate logic. The meaning of the frame formalism allowing for exceptions, however, can be described using a non-standard logic, such as for example the non-monotonic logic developed by D. McDermott and J. Doyle, or by the default logic developed by R. Reiter. We do not enter into these theories in detail; we merely give a sketch of their respective general idea.

We first consider the *non-monotonic logic* of McDermott and Doyle. In non-monotonic logic, first-order predicate logic is extended with a special *modal operator* M. The truth of a formula $M(f(x) = c)$ now means that the formula $f(x) = c$ is *possibly* true; in other words, it is not possible to derive from the given set of formulas, formulas $f(x) = d$ with $d \neq c$. In our example, the formula *blood*(*left-pulmonary-artery*) = *oxygen-poor* must be true in all models for our set of logical formulas. It therefore is undesirable that the formula *blood*(*left-pulmonary-artery*) = *oxygen-rich* can be derived, since this would lead to an inconsistency. Using the modal operator M we can block the derivation of the latter formula. The new formulas representing the given information now are as follows:

$$\begin{array}{l} \mathit{artery}(\mathit{left\text{-}pulmonary\text{-}artery}) \\ \mathit{blood}(\mathit{left\text{-}pulmonary\text{-}artery}) = \mathit{oxygen\text{-}poor} \\ \\ \forall x(\mathit{artery}(x) \rightarrow \mathit{blood\text{-}vessel}(x)) \\ \forall x(\mathit{artery}(x) \wedge M(\mathit{blood}(x) = \mathit{oxygen\text{-}rich}) \rightarrow \mathit{blood}(x) = \mathit{oxygen\text{-}rich}) \end{array}$$

Informally speaking, these formulas state that for a constant e the formula $\mathit{blood}(e) = \mathit{oxygen\text{-}rich}$ can only be derived if no other formula $\mathit{blood}(e) = c$ with $c \neq \mathit{oxygen\text{-}rich}$ can be derived. So, the formula *blood*(*left-pulmonary-artery*) = *oxygen-rich* no longer is a logical consequence of the above-given set of formulas.

The *default logic* developed by R. Reiter equally provides a way of handling exceptions but from a different perspective than non-monotonic logic does. In default logic, special inference rules, called *defaults*, are added to first-order predicate logic. The translation of the frame formalism into default logic now yields a set of logical formulas and a set of defaults. In the present case, we obtain the following set of logical formulas

$$\begin{array}{l} \mathit{artery}(\mathit{left\text{-}pulmonary\text{-}artery}) \\ \mathit{blood}(\mathit{left\text{-}pulmonary\text{-}artery}) = \mathit{oxygen\text{-}poor} \\ \\ \forall x(\mathit{artery}(x) \rightarrow \mathit{blood\text{-}vessel}(x)) \end{array}$$

and the following default

$$\frac{\mathit{artery}(x) : \mathit{blood}(x) = \mathit{oxygen\text{-}rich}}{\mathit{blood}(x) = \mathit{oxygen\text{-}rich}}$$

A default consists of a *prerequisite*, in our case the formula $\mathit{artery}(x)$, and a set of so-called *justifications*, here the formula $\mathit{blood}(x) = \mathit{oxygen\text{-}rich}$; these are specified above the line. It furthermore contains a *consequent*, here $\mathit{blood}(x) = \mathit{oxygen\text{-}rich}$, specified below the line. In this example, the default expresses that given the satisfiability of the prerequisite $\mathit{artery}(x)$ for some x in the domain and given that there are no formulas which contradict the justification $\mathit{blood}(x) = \mathit{oxygen\text{-}rich}$, then the consequent $\mathit{blood}(x) = \mathit{oxygen\text{-}rich}$ may be derived. So, in the present case *blood*(*left-pulmonary-artery*) = *oxygen-rich* cannot be derived. This is precisely what we wanted to achieve.

We conclude this section by introducing an inheritance procedure that respects the intuitive meaning of a frame formalism allowing for exceptions. It is obvious that the inheritance procedure described in the previous section cannot be applied in case attributes occur more than once in a taxonomy: this procedure might come up with conflicting information. However, only a minor modification of the procedure suffices to let it cope with exceptions. The general idea of the alteration of the inheritance procedure is as follows. Just before an attribute-value pair is added to the set of collected attribute-value pairs, it is examined whether the attribute name concerned already occurs in this set: in that case, the attribute value has been surpassed by an exception somewhere lower in the taxonomy. An attribute-value pair is only then added to the set of collected attribute values if the attribute name is not present as yet in this set. The following function describes the altered inheritance procedure more formally:

```
function Inherit(frame, attribute-value-pairs)
  if frame = nil then
    return(attribute-value-pairs)
  end;
  pairs ← AttributePart(frame);
  attribute-value-pairs ← attribute-value-pairs ∪
           NewAttributes(pairs, attribute-value-pairs);
  return(Inherit(Superframe(frame), attribute-value-pairs))
end
```

The function `NewAttributes` is used to delete from `pairs` those attribute-value pairs of which the attribute name already occurs in `attribute-value-pairs`.

The intuitive idea of this new inheritance function is that the value which holds for an attribute is given in the frame itself or in the nearest frame higher in the taxonomy providing a value for the attribute.

4.2.3 Inheritance and attribute facets

In our treatment of inheritance of attribute values in the preceding sections, we did not pay any attention to the way in which these values were obtained. In many practical applications, however, it may be important to know whether an attribute value for a given instance has been obtained by inheritance or has been explicitly specified in some way: in the latter case, the user is likely to have more confidence in the accurateness of the value than in the former case where the value has only been stated for an entire class of instances. Furthermore, it often is desirable to be able to compute attribute values based on the values of some other attributes which have been obtained during a consultation. The frame formalism discussed in the preceding sections is not able to cope with such situations. It is not surprizing therefore that most frame formalisms that are employed in systems which are actually used in practical applications, offer special language constructs, called *facets*, for the purpose of handling the situations mentioned above. In this section, we shall discuss some of the facets that are most frequently met in literature.

A facet may be viewed as a property associated with an attribute. The most common facet is the *value facet* referring to the actual value of the attribute. The value stored in a value facet of an attribute is assumed to have been established with absolute certainty. Since it is often difficult to specify with certainty in advance the values attributes of instances of a class will adopt, the initial values for class attributes are often specified in *default facets*. These default values may be overridden as the consultation proceeds. Note that our algorithm for single inheritance with exceptions already exhibited this behaviour; the difference, however, is that when facets are used, an inherited default attribute value is still marked as being a default value. In general, it depends on the characteristics of the problem area which part of the attribute values will be specified in a default facet and which part is specified in a value facet. The values specified in the default facets of attributes in a frame taxonomy together offer a typical picture of the domain of discourse.

The third facet we discuss is the *demon facet*, or *demon* for short. A demon is a procedure that will be invoked at a particular time during the manipulation of the frame in which it has been specified. The condition under which a demon is activated depends upon the type of the demon. An *if-needed demon* is activated the moment an attribute value is needed but not yet known for the attribute it is attached to. An *if-added demon* is activated the moment a value is entered into the value facet of the attribute concerned. An *if-removed demon* is invoked the moment a value is removed from the value facet of the attribute it is attached to. This way of integrating procedural and declarative knowledge is called *procedural attachment*. The frame formalism gains enormously in expressive power by the incorporation of demons. Using demons and attribute values, it for example is possible to represent local state changes due to computation: the state of the computation at a certain moment during a consultation is described by the values the attributes have at that moment.

So far, we have discussed inheritance as the only method for frame manipulation. It will be evident, however, that inheritance alone does not render a full inference engine: inheritance accounts for only a small portion of the inference engine of most frame-based systems. In many systems, demons are used to influence the overall control exerted in manipulating the frame taxonomy: an if-added demon for instance may be used to direct the control to a particular frame as a side-effect. However, great care must be taken in applying such techniques: when such side-effects are applied very often, the behaviour of the system will become difficult to fathom. This, however, is not true for every use of demons. An if-needed demon for instance can be an algorithm for asking the user for further information or for calculating a value for example in handling time-dependent information.

EXAMPLE 4.24

> Consider a real-time expert system for controlling some ongoing process that has the possibility to read off several gauges indicating the status of this process. The activation of an if-needed demon may result in reading off some of the gauges as soon as information concerning the status of the process is required.

To conclude this informal introduction to demons, we observe that frames are often used as a means of partitioning a given set of production rules: each frame then has command of a certain partition of the set of rules. In a frame taxonomy supporting this idea, a demon is used to initiate the consultation of such a partition. It is closely related to concept of *methods* in object-oriented programming. Besides the three general facets discussed in the foregoing it is of course possible to define several domain-dependent facets for a given application where appropriate.

With reference to the foregoing discussion, the following definition states a simple, implementation-oriented formalism for frames allowing for facets to be attached to attributes; once more we have refrained from type information.

Definition 4.3 *A* frame *is an expression of the following form:*

⟨frame⟩	::=	⟨class⟩ | ⟨instance⟩

⟨class⟩	::=	**class** ⟨class-name⟩ **is**
		  superclass ⟨super-specification⟩;
		  ⟨attributes⟩
		end

⟨instance⟩	::=	**instance** ⟨instance-name⟩ **is**

instance-of ⟨super-specification⟩;
⟨attributes⟩
end

⟨super-specification⟩ ::= ⟨class-name⟩ | **nil**

⟨attributes⟩ ::= ⟨attribute-facet-pair⟩ {; ⟨attribute-facet-pair⟩}* | ⟨empty⟩

⟨attribute-facet-pair⟩ ::= ⟨attribute-name⟩ = (⟨facet⟩ {, ⟨facet⟩}*)

⟨facet⟩ ::= ⟨facet-name⟩⟨value⟩ | **demon** ⟨demon-type⟩⟨demon-call⟩

⟨facet-name⟩ ::= **value** | **default**

⟨demon-type⟩ ::= **if-needed** | **if-added** | **if-removed**

⟨value⟩ ::= ⟨elementary-constant⟩ | ⟨instance-name⟩

⟨empty⟩ ::=

Again, a class specification equal to the special symbol **nil** is used to indicate that the frame concerned is the root of the tree-shaped taxonomy.

EXAMPLE 4.25

The frame shown below describes the class of arteries which is a subclass of the class of blood-vessels:

class *artery* **is**
  superclass *blood-vessel*;
  wall = (**value** *muscular*);
  blood = (**default** *oxygen-rich*);
  blood-pressure = (**default** 20);
  blood-flow = (**default** 4);
  resistance = (**demon if-needed** *R*(*blood-pressure*, *blood-flow*))
end

The frame has five attributes. For the attribute with the name *wall* a value facet has been specified, since we are absolutely certain that all arteries have a muscular wall. For the three attributes with default facets things are different. We already know that not all arteries contain oxygen-rich blood, hence the default facet for the *blood* attribute. The attribute *blood-pressure* represents the difference in blood pressure between the pressure at beginning and at the end of an artery. The value specified for this attribute as well as the value for the *blood-flow* attribute are average values for middle-sized arteries. Evidently, such knowledge is best represented in default facets. Finally, we have one attribute, the *resistance* attribute, for which an if-needed demon has been specified for calculating a value upon request. The demon call *R*(*blood-pressure*, *blood-flow*) represents the call to the procedure *R* for computing the resistance to the blood flow in the given artery using the formula

$$resistance = \frac{blood\text{-}pressure}{blood\text{-}flow}$$

The values of the attributes *blood-pressure* and *blood-flow* are passed to the procedure.

Since we now allow for various facets to be attached to attributes, it has become necessary to incorporate information concerning the order in which such facets are considered in our algorithm for single inheritance. Basically, there are two types of inheritance of attributes with facets, only differing in the order in which the facets are dealt with:

- *N-inheritance*, and
- *Z-inheritance*.

These types of inheritance owe their respective names to the way the taxonomy is traversed.

The intuition underlying N-inheritance is that any value in a value facet appearing in a frame or in one of its generalizations is closer to the real value than any value obtained from a default facet or from invoking a demon. As usual, of all the values of an attribute stored in value facets in the various frames in the frame taxonomy the most specific one will be inherited by the frame of concern. The basic idea underlying the use of Z-inheritance is that any specific attribute value, whether obtained from a value facet, from invoking a demon or from a default facet, is more reliable than any more general attribute value no matter in which facet it has been specified; however, within a given frame, values specified in a value facet are preferred over those computed by a demon or provided by a default facet.

The procedures for N- and Z-inheritance can now be described informally as follows. Applying N-inheritance, an attribute value is determined by first examining the value facet of the frame concerned. If no value facet has been specified, then the value facet attached to the corresponding attribute in the superframe of the frame is examined. This process is repeated until in a frame higher in the taxonomy a value facet has been found or the root of the taxonomy has been reached. If the process has not yielded an attribute value as yet, then the control is returned to the frame of interest, and the process is repeated for the if-needed demons. Finally, if this process has still not yielded an attribute value, the default facets will be examined in a similar way. Figure 4.13 depicts the behaviour of N-inheritance graphically. Applying Z-inheritance, an attribute value is determined by successively examining the value facet, the if-needed demon and the default facet of the frame concerned before the frames higher in the taxonomy are considered. Figure 4.14 shows the behaviour of Z-inheritance.

4.2.4 Subtyping in tree-shaped taxonomies

In Section 4.2.1 we have introduced a syntax for a frame formalism which allowed for declarations of attribute types. Until now we have disregarded such type information. In the present section, however, we shall pay attention to type information and discuss the properties of the relation defined by the superclass links now viewed as a relation between attribute types. In discussing this relation in the context of type information, it is more common to speak of the *supertype relation*, or reversely of the *subtype relation*; exploiting the subtype relation is known as *subtyping*. In the present section, we study subtyping in tree-shaped

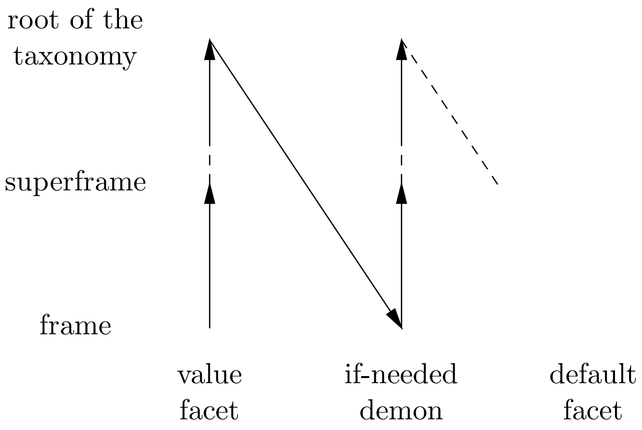


Figure 4.13: N-inheritance.

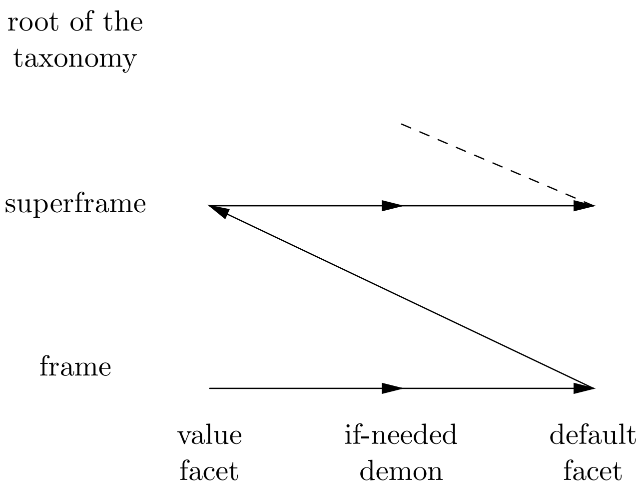


Figure 4.14: Z-inheritance.

frame taxonomies in which attribute-type pairs for a particular attribute may occur more than once.

It will be evident that it is desirable to have type information available in a fame taxonomy: attribute types may be exploited for checking entered values on being of the proper type. This way, an 'unexpected' attribute value can be detected as soon as it is entered into the attribute concerned; in that case, the attribute value is considered to be erroneous.

EXAMPLE 4.26

Consider a frame taxonomy representing information concerning the human cardiovascular system. We assume that the class frame with the name *artery* has an attribute *mean-blood-pressure* that describes the mean blood pressure in mmHg in the arteries in a normal, healthy human being. In the arteries the mean blood pressure ranges from 30 mmHg to 100 mmHg. This information may be stored as type information for the *mean-blood-pressure* attribute. When in a patient for a specific artery, say the ulnar artery, a mean blood pressure of 10 mmHg is found, then this is an unexpected value and probably some action has to be taken upon this event. The specification of an attribute type can be further specialized, that is, narrowed down, as the frame it is specified in is further specialized. Suppose for example that the *artery* class has three specializations: the *large-artery*, the *small-artery*, and the *arteriole* class. In the class representing the large arteries, the information for the *mean-blood-pressure* attribute is further specialized to the range 90 – 100 mmHg, in the *small-artery* class to the range 60–90 mmHg and in the *arteriole* class to 30–60 mmHg.

We give a more formal example in which we have specified some type information in the manner prescribed by the frame formalism.

EXAMPLE 4.27

Consider again the class of blood vessels. The following class representation shows some attribute types for this class of objects:

class *blood-vessel* **is**
  superclass nil;
  blood : {*oxygen-rich*, *oxygen-poor*};
  wall : {*muscular*, *fibrous*, *mixed*}
end

The class of arteries is a subclass of the class of blood-vessels. It is represented in the following class frame:

class *artery* **is**
  superclass *blood-vessel*;
  wall : {*muscular*, *mixed*};
  wall-thickness : **real**
end

Every attribute type specified in the class frame with the name *blood-vessel* now is taken to apply to the class *artery* as well, as long as it has not been further specialized in the *artery* class itself. Note that both classes contain a type declaration for the attribute with the name *wall*. The type declaration included in the *blood-vessel* class is more general than the one in the *artery* class.

From these two examples, we may conclude that subtyping involves the relationship between attributes occurring in the classes as well as the relationship between the types of those attributes. Before going into more detail, we first introduce some new notions that will be used in formalizing subtyping in a tree-shaped frame taxonomy. We shall see that it is convenient to have some representation of the set of attribute names that are of concern to a specific class. Since an attribute type may itself be a class, it does not suffice to simply enlist the attribute names actually occurring in a class. Therefore, we associated with a class a set of so-called attribute sequences.

Definition 4.4 *Let A be the set of all attribute names occurring in a frame taxonomy. An* attribute sequence *a is a string of the form $a_1 : \cdots : a_n$, where $a_i \in A$, $i = 1, \ldots, n$, $n \geq 0$, that is, an attribute sequence is composed of elements from A separated by semicolons. The attribute sequence comprising no elements at all, that is, for which $n = 0$, is called the* empty attribute sequence *and is denoted by ϵ. From now on, we shall use A^* to denote the (infinite) set of all attribute sequences constructed from A.*

Note that we have not imposed any restriction on for example the order in which attribute names are allowed to occur in an attribute sequence. With every class frame in a frame taxonomy we now associate a subset of the set of attribute sequences.

Definition 4.5 *Let A^* be a set of attribute sequences associated with a frame taxonomy as defined above. Let y be a class frame in the taxonomy. With y we associate the set $D(y) \subseteq A^*$, called the* domain *for y, defined by:*

(1) $\epsilon \in D(y)$;

(2) For every attribute name a specified in y, we have that $a \in D(y)$;

(3) For every attribute with the name a of type w specified in y, $D(y)$ contains the attribute sequences $a : b$ for all elements $b \in D(w)$;

(4) The set $D(z)$ of attribute sequences associated with the superframe z of y is a subset of $D(y)$.

We give an example.

EXAMPLE 4.28 ---

Consider the following class frame with the name *blood-vessel*:

class *blood-vessel* **is**
  superclass nil;
  volume : *cubic-measure*
end

This frame specifies an attribute *volume* providing information concerning the blood volume for the specializations of the class. The type *cubic-measure* of this attribute is a class frame itself. This class is defined as follows:

> **class** *cubic-measure* **is**
>   **superclass nil**;
>   *size* : **real**;
>   *unit* : $\{mm^3, cm^3, dm^3, m^3\}$
> **end**

The set of attribute names in the taxonomy consisting of these two frames is equal to $A = \{volume, size, unit\}$. The set of attribute sequences associated with the class frame *blood-vessel* is the set $D(\textit{blood-vessel}) = \{\epsilon, volume, volume : size, volume : unit\}$; the domain for the *cubic-measure* class is the set $D(\textit{cubic-measure}) = \{\epsilon, size, unit\}$.

In the following definition we introduce the notion of a type function for computing the types of the attribute sequences associated with a given frame.

Definition 4.6 *Let A^* be the set of attribute sequences in a frame taxonomy. Let K be the set of class names in that frame taxonomy (including the standard classes and the most general class* **nil***). For each class $y_i \in K$, let $D(y_i) \subseteq A^*$ be the set of attribute sequences associated with y_i as in the preceding definition. Now, for each $y_i \in K$, we define a* type function *$\tau_i : A^* \to K$ as follows:*

(1) For the empty attribute sequence ϵ, we have that $\tau_i(\epsilon) = y_i$;

(2) For each attribute sequence $a = a_1 : \cdots : a_n \in D(y_i)$, $n \geq 1$, we have that $\tau_i(a) = t$ where t is the type of the attribute with the name a_n.

*(3) For each $a \in A^* \backslash D(y_i)$, we have $\tau_i(a) =$ **nil**.*

EXAMPLE 4.29

Consider the frame taxonomy consisting of the two class frames from the previous example. The set K of classes in this taxonomy equals

$$K = \{\textbf{nil}, \textit{blood-vessel}, \textit{cubic-measure}, \textbf{real}, \{mm^3, cm^3, dm^3, m^3\}\}$$

Let $D(\textit{blood-vessel})$ be the set of attribute sequences associated with the *blood-vessel* class as in the previous example. For this class, the type function $\tau_1 : A^* \to K$ is defined by

$$\begin{aligned}
&\tau_1(\epsilon) = \textit{blood-vessel}\\
&\tau_1(volume) = \textit{cubic-measure}\\
&\tau_1(volume : size) = \textbf{real}\\
&\tau_1(volume : unit) = \{mm^3, cm^3, dm^3, m^3\}\\
&\tau_1(a) = \textbf{nil} \text{ for all } a \in A^* \backslash D(\textit{blood-vessel})
\end{aligned}$$

Let $D(\textit{cubic-measure})$ be the domain for the class frame *cubic-measure*. For this class, the type function $\tau_2 : A^* \to K$ is defined as follows:

$\tau_2(\epsilon) = \textit{cubic-measure}$
$\tau_2(\textit{size}) = \textbf{real}$
$\tau_2(\textit{iunit}) = i\{mm^3, cm^3, dm^3, m^3\}$
$\tau_2(a) = \textbf{nil}$ for all $a \in A^* \backslash D(\textit{cubic-measure})$

We now are ready to consider a type semantics for the superclass links in a taxonomy. We introduce the notion of a subtype.

Definition 4.7 *Let y_1 and y_2 be two class frames in a tree-shaped frame taxonomy. Let A^* be the set of attribute sequences in the taxonomy. Furthermore, let $D(y_i)$ be the set of attribute sequences associated with the class frame y_i, $i = 1, 2$. Now, let τ_i be the type function associated with y_i. We say that y_1 is a* subtype *of y_2, denoted by is $y_1 \leq y_2$, if the following two properties hold:*

(1) $D(y_2) \subseteq D(y_1)$;

(2) For each attribute sequence $a \in A^*$, we have that $\tau_1(a) \leq \tau_2(a)$.

We say that a taxonomy is *correctly typed* if the superclass links in the taxonomy satisfy the properties of the relation $\leq$ from the previous definition. Note that we now have that the meaning of the subtype relation can be described in terms of set inclusion $\subseteq$: we associate with each type t a subset $I(t)$ of elements from a domain of discourse U, such that if for two types t_1 and t_2 we have that $t_1 \leq t_2$, then we have that the property $I(t_1) \subseteq I(t_2)$ holds.

EXAMPLE 4.30

Consider the second example from this section once more. We have that the *artery* class is a superframe of the *blood-vessel* class. The set of attribute sequences associated with the class *blood-vessel* is equal to the set

$$D(\textit{blood-vessel}) = \{\epsilon, \textit{blood}, \textit{wall}\}$$

the domain for the *artery* class is equal to

$$D(\textit{artery}) = \{\epsilon, \textit{blood}, \textit{wall}, \textit{wall-thickness}\}$$

So, the first condition in the preceding definition is satisfied since we have that $D(\textit{blood-vessel}) \subseteq D(\textit{artery})$. The type function $\tau_1 : A^* \to K$ associated with the class *artery* is defined by:

$\tau_1(\epsilon) = \textit{artery}$
$\tau_1(\textit{blood}) = \{\textit{oxygen-rich}, \textit{oxygen-poor}\}$
$\tau_1(\textit{wall}) = \{\textit{muscular}, \textit{mixed}\}$
$\tau_1(a) = \textbf{nil}$ for all $a \in A^* \backslash D(\textit{artery})$

The type function $\tau_2 : A^* \to K$ associated with the class *blood-vessel* is defined by:

$\tau_2(\epsilon) = \textit{blood-vessel}$
$\tau_2(\textit{blood}) = \{\textit{oxygen-rich}, \textit{oxygen-poor}\}$
$\tau_2(\textit{wall}) = \{\textit{muscular}, \textit{fibrous}, \textit{mixed}\}$
$\tau_2(a) = \textbf{nil}$ for each $a \in A^* \backslash D(\textit{blood-vessel})$

The reader can easily verify that $\tau_1(a) \leq \tau_2(a)$ for each $a \in A^*$. We conclude that the frame taxonomy is correctly typed.

4.3 Frames and multiple inheritance

So far, we have only dealt with tree-shaped frame taxonomies and single inheritance. In this section, we introduce more general frame taxonomies. The frame formalism that has been defined in Section 4.2.1 is extended by admitting in class frames more than one class name in the superclass link field; this way, it is possible for a class to have more than one superclass. The graphical representation of such a taxonomy then takes the form of a general directed graph instead of a tree. In the following, we shall restrict the discussion to acyclic directed graphs, because trivial cycles obtained from the reflexivity property of the subclass relation, which are the only cycles that can be constructed when the superclass relation is viewed as a partial order, will not be explicitly indicated. The inheritance algorithm associated with graph-shaped frame taxonomies is known as *multiple* inheritance. In discussing multiple inheritance, we assume that more than one value for an attribute may have been specified in a frame taxonomy, that is, we allow for exceptions; however, we shall restrict the discussion to value facets of attributes only. Note that the theory developed in this section should hold for tree-shaped frame taxonomies and general graph-shaped taxonomies alike, since the former is just a special case of the latter.

4.3.1 Multiple inheritance of attribute values

Multiple inheritance differs from single inheritance mainly in the way it handles attributes occurring more than once in a taxonomy with different values. As in Section 4.2.2, we shall call such attribute values *exceptions*. Recall that when exceptions have been specified, conflicting information may be derived due to the inheritance of mutually exclusive values. In Section 4.2.2, however, we have seen that the problem of handling exceptions is easily solved in the case of single inheritance in a tree-shaped taxonomy: inheritance is taken as a process for finding a value for an attribute that starts with a given vertex in the tree, which moves along the branches of the tree towards the root and stops as soon as a value for the attribute of concern has been obtained. This algorithm always finds at most one attribute value. Unfortunately, the problem is much more complicated in the case of multiple inheritance in a general graph-shaped taxonomy. The algorithm for multiple inheritance in graph-shaped taxonomies in which exceptions occur has to incorporate a method for explicitly deciding which value of an attribute is to be preferred; we speak of *multiple inheritance with exceptions*. The main part of this section will be devoted to the development of such an algorithm for multiple inheritance with exceptions.

To start with, some new notions and notational conventions are introduced, including a more compact notation for the representation of frame information which has a stronger mathematical flavour than the implementation-oriented syntax introduced in the preceding section. From now on, $K = \{y_1, y_2, \ldots, y_n\}$, $n \geq 0$, will denote a fixed set of class frames, and $I = \{x_1, x_2, \ldots, x_m\}$, $m \geq 0$, will denote a fixed set of instance frames; the sets I and K are disjoint. The set of frames F is equal to $I \cup K$.

In a frame taxonomy, the superclass links are viewed as members of a relation between class frames. In the following definition we indicate that this relation may be viewed as a partial order.

Definition 4.8 *Let K denote the fixed set of class frames. The* subclass relation $\leq$ *is a binary relation on K, that is $\leq \subseteq K \times K$, that defines a partial order on the set K. For a pair $(x, y) \in \leq$, denoted by $x \leq y$, it is said that x* is a subclass of *y.*

Recall that, since the subclass relation $\leq$ defines a partial order on the set of class frames K, it satisfies the properties mentioned in Section 4.1.1.

The instance-of links are viewed as members of a relation between the set of instance frames I and the set of class frame K. We assume that an instance belongs to exactly one class. The instance-of links therefore are best formalized by means of a function.

Definition 4.9 *Let I denote the fixed set of instance frames and K the fixed set of classes. The* instance-of function $\ll: I \rightarrow K$ *is a mapping from I to K. In the sequel, we shall denote $\ll(x) = y$ by $x \ll y$; we say that x* is an instance of *y.*

The subclass relation and the instance-of function introduced in the two preceding definitions only describe reference information. The following definition introduces another relation meant to arrive at a full language for the specification of frame information.

Definition 4.10 *Let F be the set of frames such that $F = I \cup K$, where I is the set of instance frames in F, and K the set of class frames in F. Let A be a fixed set of attribute names and let C be a fixed set of constants. Then, a triple $(x, a, c) \in F \times A \times C$, denoted by $x[a = c]$, is called an* attribute-value specification. *An* attribute-value relation Θ *is a ternary relation on F, A and C, that is, $\Theta \subseteq F \times A \times C$.*

In the previous definition we have explicitly specified a set of constants C. Note, however, that this set of constants may be identified with the set of instances I, as we have done in the preceding sections. An attribute-value specification $x[a = c]$ expresses that in the frame x the attribute a has the constant value c. The notions introduced in the foregoing definitions are now used to formally define a frame taxonomy.

Definition 4.11 *Let I be the set of instances and K the set of classes. Furthermore, let A be the set of attribute names and C the set of constants. I, K, A and C are disjoint. Now, let N be the quadruple $N = (I, K, A, C)$. Furthermore, let the relations $\leq$ and Θ, and the function $\ll$ be defined as above. Then, a* taxonomy *T is a quadruple $T = (N, \Theta, \ll, \leq)$.*

We give an example of the frame formalism we have just defined and its relation with the frame formalism introduced in Section 4.2.1.

EXAMPLE 4.31

Consider the information specified in the following three classes represented in the frame formalism from Section 4.2.1:

> **class** *blood-vessel* **is**
>   **superclass nil**;
>   *contains* = *blood-fluid*
> **end**

class *artery* **is**
  superclass *blood-vessel*;
  blood = *oxygen-rich*;
  wall = *muscular*
end

class *vein* **is**
  superclass *blood-vessel*;
  wall = *fibrous*
end

instance *aorta* **is**
  instance-of *artery*;
  diameter = 2.5
end

In the specified taxonomy, we have that $I = \{aorta\}$ is the set of instance frames and that $K = \{artery, vein, blood\text{-}vessel\}$ is the set of classes. Furthermore, we have that $A = \{contains, blood, wall, diameter\}$, and $C = \{blood\text{-}fluid, oxygen\text{-}rich, muscular, fibrous, 2.5\}$. We have the following set of attribute-value specifications:

$$\begin{aligned}\Theta = \{ & blood\text{-}vessel[contains = blood\text{-}fluid],\\ & artery[blood = oxygen\text{-}rich],\\ & artery[wall = muscular],\\ & vein[wall = fibrous],\\ & aorta[diameter = 2.5]\}\end{aligned}$$

The function $\ll$ and the relation $\leq$ are defined by

$$\begin{aligned}& aorta \ll artery\\ & artery \leq blood\text{-}vessel\\ & vein \leq blood\text{-}vessel\end{aligned}$$

Now, $T = (N, \Theta, \ll, \leq)$ is the taxonomy shown above, this time represented using our new formalism.

Just as before, a taxonomy $T = (N, \Theta, \ll, \leq)$ can be represented graphically by means of an acyclic directed graph in which the vertices represent the frames in I and K, and the arcs represent the relation $\leq$ and the function $\ll$. A vertex is assumed to have an internal structure representing the collection of attribute-value specifications associated with the frame by the relation Θ. In the graphical representation, an attribute-value specification is depicted next to the vertex it belongs to; only the attribute and constant of an attribute-value specification are shown. We indicate the relation $\leq$ by means of a pulled arrow; and the function $\ll$ will be depicted by means of a dashed arrow. In the graphical representation of a taxonomy, arcs expressing the reflexivity and transitivity of the subclass relation will be left out in most cases. The omission of the arcs representing reflexivity, has no effect on the inheritance of attribute values, and is therefore permitted. However, leaving out arcs representing the transitivity property of the subclass relation, is one of the causes of problems concerning the inheritance of mutually exclusive attribute value, since this may alter the meaning of a taxonomy. Most of the remainder of this section is therefore devoted to an investigation of the consequences of this decision. Figure 4.15 shows the taxonomy from the previous example.

The relation $\leq$ defined above is now taken as the basis for reasoning with frames. We shall define so-called inheritance chains for the representation of the reasoning process that takes place in a frame taxonomy. These chains will constitute our principal device for dealing with exceptions in multiple inheritance. In the following two definitions the syntactic form of such inheritance chains and a procedure for their construction is presented. We shall first be concerned with inheritance of attribute values for classes only; later on we turn to inheritance of attribute values for instances.

Definition 4.12 *Let $T = (N, \Theta, \ll, \leq)$ be a taxonomy having the usual meaning, where $N = (I, K, A, C)$. An* inheritance chain *in T is an expression having one of the following forms:*

$$y_1 \leq \ldots \leq y_n$$
$$y_1 \leq \ldots \leq y_n[a = c]$$

where $y_i \in K$, $i = 1, \ldots, n$, $n \geq 1$, are class frames, and $y_n[a = c] \in \Theta$ is an attribute-value specification.

Note that attribute-value specifications are allowed only in isolation, or at the end of an inheritance chain. Furthermore, we observe that inheritance chains of the form $y_1 \leq \ldots \leq y_n$ are just another way of characterizing the subclass relation, obtained from its satisfying the properties of reflexivity and transitivity. Although we allow inheritance chains in which the reflexivity of the subclass relation $\leq$ is exploited, we shall not show such chains in our examples since they do not contribute to the notions we want to illustrate.

The set of all possible inheritance chains in a given frame taxonomy is constructed as described in the next definition.

Definition 4.13 *Let $T = (N, \Theta, \ll, \leq)$ be a taxonomy where $N = (I, K, A, C)$. The set Ω_T of inheritance chains in T is defined as follows:*

- *For each $y \in K$, we have $y \in \Omega_T$.*
- *For each $y[a = c] \in \Theta$ where $y \in K$, we have $y[a = c] \in \Omega_T$.*

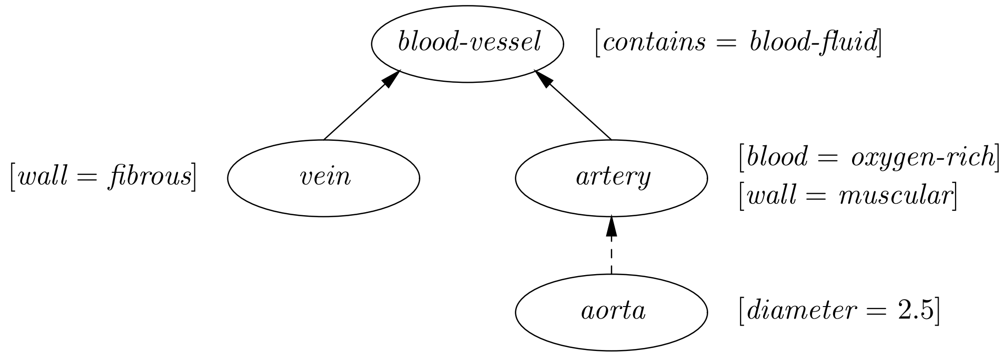


Figure 4.15: A taxonomy consisting of three classes and one instance.

- *For each pair* $(y_1, y_2) \in \leq$ *we have* $y_1 \leq y_2 \in \Omega_T$.
- *For each* $y_1 \leq \ldots \leq y_k \in \Omega_T$ *and* $y_k \leq \ldots \leq y_n \in \Omega_T$, $1 \leq k \leq n$, $n \geq 1$, *where* $y_i \in K$, $i = 1, \ldots, n$, *we have that* $y_1 \leq \ldots \leq y_n \in \Omega_T$.
- *For each* $y_1 \leq \ldots \leq y_n \in \Omega_T$ *and* $y_n[a = c] \in \Omega_T$, *where* $y_i \in K$, $i = 1, \ldots, n$, $n \geq 1$, *we have that* $y_1 \leq \ldots \leq y_n[a = c] \in \Omega_T$.

EXAMPLE 4.32

Consider the taxonomy $T = (N, \Theta, \ll, \leq)$ in which

$I = \{aorta\}$
$K = \{large\text{-}artery, artery, blood\text{-}vessel\}$
$\Theta = \{aorta[diameter = 2.5], artery[wall = muscular], large\text{-}artery[mean\text{-}pressure = 100],$
$\quad blood\text{-}vessel[contains = blood\text{-}fluid]\}$

The function $\ll$ is defined by *aorta* $\ll$ *large-artery*, and the relation $\leq$ is defined by *large-artery* $\leq$ *artery* and *artery* $\leq$ *blood-vessel*. Recall that inheritance chains in which the reflexivity of the subclass relation $\leq$ is used, will not be shown; to give an impression of how such chains look like we show one of them:

artery $\leq$ *artery* $\leq$ *artery*

Now, the set of inheritance chains Ω_T consists of the following elements:

artery
large-artery
blood-vessel
artery[*wall* = *muscular*]
large-artery[*mean-pressure* = 100]
blood-vessel[*contains* = *blood-fluid*]
large-artery $\leq$ *artery*
large-artery $\leq$ *artery*[*wall* = *muscular*]
large-artery $\leq$ *artery* $\leq$ *blood-vessel*
large-artery $\leq$ *artery* $\leq$ *blood-vessel*[*contains* = *blood-fluid*]
artery $\leq$ *blood-vessel*
artery $\leq$ *blood-vessel*[*contains* = *blood-fluid*]

Inheritance chains are viewed as descriptions of which attribute-value specifications may *possibly* be inherited by the frames in the taxonomy. We shall see shortly that in multiple inheritance with exceptions certain combinations of attribute-value specifications when actually inherited represent contradictory information. Under suitable conditions, however, certain inheritance chains may be cancelled from the set of all inheritance chains in the taxonomy, thus preventing the occurrence of a contradiction. Before discussing this idea in further detail, we introduce the notion of the conclusion of an inheritance chain which is an explicit means for establishing which attribute-value specification may be inherited from the chain.

Definition 4.14 *Let $T = (N, \Theta, \ll<, \leq)$ be a taxonomy where $N = (I, K, A, C)$. Let Ω_T be the set of inheritance chains in T. The* conclusion $c(\omega)$ *of an inheritance chain $\omega \in \Omega_T$ is defined as follows:*

- *For each $\omega \equiv y_1 \leq \ldots \leq y_n[a = c]$, we have that $c(\omega) = y_1[a = c]$.*
- *For all other ω, we have that $c(\omega)$ is not defined.*

The conclusion set $C(\Omega_T)$ *of Ω_T is defined as the set of conclusions of all elements from Ω_T, that is, $C(\Omega_T) = \{c(\omega) \mid \omega \in \Omega_T\}$.*

When the attribute-value specification $z[a = c]$ is obtained as the conclusion of an inheritance chain, we say that the value c of the attribute a has been *inherited* by z.

EXAMPLE 4.33

Consider again the set Ω_T of inheritance chains from the preceding example. The conclusion set $C(\Omega_T)$ of Ω_T then consists of the following attribute-value specifications:

large-artery[*mean-pressure* = 100]
large-artery[*wall* = *muscular*]
large-artery[*contains* = *blood-fluid*]
artery[*wall* = *muscular*]
artery[*contains* = *blood-fluid*]
blood-vessel[*contains* = *blood-fluid*]

The conclusion set $C(\Omega_T)$ of a given set of inheritance chains Ω_T may contain attribute-value specifications which only differ in their specified constant. We have already encountered the notion of exception and its related problems in Section 4.2.2. In the following example, we restate the problem in terms of inheritance chains.

EXAMPLE 4.34

In the foregoing, it has frequently been pointed out that the left and right pulmonary arteries have much in common with arteries except that they contain oxygen-poor instead of oxygen-rich blood. Now, consider the set Ω of inheritance chains containing, among other ones, the following two chains:

pulmonary-artery[*blood* = *oxygen-poor*]
pulmonary-artery $\leq$ *artery*[*blood* = *oxygen-rich*]

The conclusion set constructed from Ω contains at least the following two attribute-value specifications:

pulmonary-artery[*blood* = *oxygen-poor*]
pulmonary-artery[*blood* = *oxygen-rich*]

Clearly, if a sensible meaning is to be associated with the frame formalism, only one of these conclusions should be satisfied.

We call a conclusion set $C(\Omega_T)$ inconsistent if it contains contradictory information such as in the previous example. In the following definition the notions of consistency and inconsistency of a conclusion set are defined more formally.

Definition 4.15 *Let $T = (N, \Theta, \ll, \leq)$ be a taxonomy where $N = (I, K, A, C)$. Let Ω_T be the set of inheritance chains in T. Furthermore, let $C(\Omega_T)$ be the conclusion set of Ω_T. The conclusion set $C(\Omega_T)$ is called* inconsistent *if it contains attribute-value specifications $y[a = c_1]$ and $y[a = c_2]$, $c_1, c_2 \in C$, such that $c_1 \neq c_2$. Otherwise, the conclusion set $C(\Omega_T)$ is said to be* consistent.

Inconsistency of the conclusion set of a taxonomy indicates that inheritance in the taxonomy is not defined uniquely: only if the conclusion set is consistent, the instances of the taxonomy inherit unambiguous information from the classes they belong to. We now consider inheritance of attribute values for instances in more detail. Informally speaking, the attribute-value specifications that hold for an instance of a specific class frame are the attribute-value specifications explicitly specified in the instance itself supplemented with the attribute-value specifications holding for the class it belongs to that do not contradict the attribute-value specifications from the instance. This is defined more formally below.

Definition 4.16 *Let $T = (N, \Theta, \ll, \leq)$ be a taxonomy where $N = (I, K, A, C)$. Let Ω_T be the set of inheritance chains in T and let $C(\Omega_T)$ be the conclusion set of Ω_T. For each instance frame $x \in I$, the set $e_C(x)$ is defined by $e_C(x) = \{x[a = c] \mid x[a = c] \in \Theta\} \cup \{x[a = c] \mid x \ll y, y \in K, y[a = c] \in C(\Omega_T)$ and for all $c \neq d$, $x[a = d] \notin \Theta\}$ if $C(\Omega_T)$ is consistent; $e_C(x)$ is undefined otherwise. The* extension of Ω_T, *denoted by $E_C(\Omega_T)$, is defined by*

$$E_C(\Omega_T) = \bigcup_{x \in I} e_C(x)$$

if $C(\Omega_T)$ is consistent; $E_C(\Omega_T)$ is undefined otherwise.

EXAMPLE 4.35

Consider the taxonomy T from the first example of this section once more. Let Ω_T be the set of inheritance chains in T and let $C(\Omega_T)$ be the conclusion set of Ω_T. The extension of Ω_T is equal to the following set of attribute-value specifications:

$$\begin{aligned} E_C(\Omega_T) = \{ & \mathit{aorta}[\mathit{contains} = \mathit{blood\text{-}fluid}], \\ & \mathit{aorta}[\mathit{mean\text{-}pressure} = 100], \\ & \mathit{aorta}[\mathit{wall} = \mathit{muscular}], \\ & \mathit{aorta}[\mathit{diameter} = 2.5]\} \end{aligned}$$

A taxonomy that is inconsistent in the sense of its having an inconsistent conclusion set can sometimes be 'made' consistent by cancelling some of the inheritance chains from the set of inheritance chains in the taxonomy by using knowledge concerning the hierarchical ordering of the frames. As a consequence, certain conclusions are cancelled from the conclusion set of the taxonomy as well, thereby preventing the occurrence of some contradictory attribute values. Note that this way non-monotonic reasoning is introduced within the frame formalism.

For cancelling inheritance chains, we shall exploit the notion of an intermediary, which is introduced in the following definition.

Definition 4.17 *Let $T = (N, \Theta, \ll, \leq)$ be a taxonomy where $N = (I, K, A, C)$. Let Ω_T be the set of inheritance chains in T. A class $y \in K$ is called an* intermediary *to an inheritance chain $y_1 \leq \ldots \leq y_n \in \Omega_T$, $y_i \in K$, $i = 1, \ldots, n$, $n \geq 1$, if one of the following conditions is satisfied:*

- *We have $y = y_i$ for some i, $1 \leq i \leq n$.*
- *There exists a chain $y_1 \leq \ldots \leq y_p \leq z_1 \leq \ldots \leq z_m \leq y_q \in \Omega_T$, for some p, q, $1 \leq p \leq q \leq n$, where $z_j \neq y_i$, $i = 1, \ldots, n$, $z_j \in K$, $j = 1, \ldots, m$, $m \geq 1$, such that $y = z_k$, for some k, $1 \leq k \leq m$.*

EXAMPLE 4.36

Consider the taxonomy $T = (N, \Theta, \ll, \leq)$, where $I = \varnothing$, $K = \{$*blood-vessel*, *artery*, *oxygen-poor-artery*, *pulmonary-artery*$\}$, Θ is empty, and the relation $\leq$ is defined by

pulmonary-artery $\leq$ *oxygen-poor-artery*
pulmonary-artery $\leq$ *artery*
artery $\leq$ *blood-vessel*
oxygen-poor-artery $\leq$ *artery*

The graphical representation of the taxonomy is shown in Figure 4.16. The set of inheritance chains in T contains, among other ones, the following two chains:

pulmonary-artery $\leq$ *artery* $\leq$ *blood-vessel*
pulmonary-artery $\leq$ *oxygen-poor-artery* $\leq$ *artery*

It will be evident that the class *oxygen-poor-artery* is an intermediary to both chains.

Figure 4.16 introduced in the foregoing example is useful for gaining some intuitive feeling concerning the notion of an intermediary.

We shall see that intermediaries may be applied for solving part of the problem of multiple inheritance with exceptions. We take a closer look at the figure. It seems as if the arc between the vertices *pulmonary-artery* and *artery*, an arc resulting from the transitivity property of the subclass relation, is redundant, since all attribute-value specification from the classes *artery* and *blood-vessel* can be inherited for *pulmonary-artery* via the vertex *oxygen-poor-artery*. Therefore, the removal of this arc from the taxonomy should not have any influence on the result of multiple inheritance. Whether or not this is true is, of course, dependent on our formalization of multiple inheritance. Therefore, let us investigate whether the notion of conclusion set defined in the foregoing renders a suitable means for dealing with exceptions. We do so by means of an example.

EXAMPLE 4.37

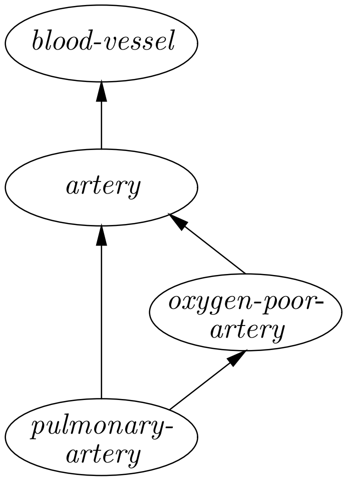


Figure 4.16: A taxonomy with an intermediary.

Consider Figure 4.16 once more. Figure 4.17 shows the taxonomy from Figure 4.16 after removal of the seemingly redundant arc. Now, suppose that the following attribute-value specifications are given:

oxygen-poor-artery[*blood* = *oxygen-poor*]
artery[*blood* = *oxygen-rich*]

Furthermore, suppose that no attribute-value specifications have been given for *pulmonary-artery*. In the taxonomy shown in Figure 4.17, the frame *pulmonary-artery* inherits only the value *oxygen-poor* for the attribute *blood*; note that this is a consequence of the way exceptions are handled in tree-shaped taxonomies. However, in Figure 4.16 the frame *pulmonary-artery* inherits both values *oxygen-poor* and *oxygen-rich* for the attribute *blood*, leading to an inconsistent conclusion set. The conclusion set of the taxonomy in Figure 4.16 therefore differs from the one obtained for the taxonomy shown in Figure 4.17, using the algorithm for single inheritance with exceptions discussed in Section 4.2.2s in the last case.

It turns out that a conclusion set only reveals the presence of exceptions in a taxonomy. We shall see that the notion of an intermediary is more useful in dealing with exceptions in multiple inheritance. In Figure 4.16 we have that the class *oxygen-poor-artery* lies in between the classes *pulmonary-artery* and *artery*, and is an intermediary to the inheritance chains in which the class *pulmonary-artery* and either or both the classes *artery* and *oxygen-poor-artery* occur. As we have suggested before, by means of intermediaries some of the inheritance chains may be cancelled rendering a different set of conclusions of the taxonomy. Such cancellation of inheritance chains is called *preclusion* and is defined more formally below.

Definition 4.18 *Let $T = (N, \Theta, \ll, \leq)$ be a taxonomy where $N = (I, K, A, C)$. Let Ω_T be the set of inheritance chains in T. A chain $y_1 \leq \ldots \leq y_n[a = c_1] \in \Omega_T$, where $n \geq 1$, is said to* preclude *a chain $y_1 \leq \ldots \leq y_m[a = c_2] \in \Omega_T$, where $m \geq 1$, $m \neq n$, and $c_1, c_2 \in C$ with $c_1 \neq c_2$, if y_n is an intermediary to $y_1 \leq \ldots \leq y_m$.*

EXAMPLE 4.38

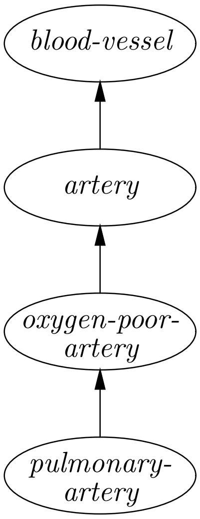


Figure 4.17: The taxonomy after removal of the redundant arc.

Consider the set Ω_T of inheritance chains consisting of the following elements:

ω_1: *pulmonary-artery* $\leq$ *oxygen-poor-artery*
ω_2: *pulmonary-artery* $\leq$ *artery*
ω_3: *pulmonary-artery* $\leq$ *oxygen-poor-artery* $\leq$ *artery*
ω_4: *pulmonary-artery* $\leq$ *oxygen-poor-artery*[*blood* = *oxygen-poor*]
ω_5: *pulmonary-artery* $\leq$ *artery*[*blood* = *oxygen-rich*]
ω_6: *pulmonary-artery* $\leq$ *oxygen-poor-artery* $\leq$ *artery*[*blood* = *oxygen-rich*]

The reader can easily verify that the inheritance chain ω_4 precludes both chains ω_5 and ω_6 since *oxygen-poor-artery* is an intermediary to the chains ω_2 and ω_3.

The notion of preclusion is used for introducing a new type of conclusion set of a set of inheritance chains.

Definition 4.19 *Let $T = (N, \Theta, \ll, \leq)$ be a taxonomy. Let Ω_T be the set of inheritance chains in T. An inheritance chain $\omega \in \Omega_T$ is said to be* inheritable *if there exists no other inheritance chain $\omega' \in \Omega_T$ which precludes ω. The set of conclusions of all inheritable chains $\omega \in \Omega_T$ is called the* inheritable conclusion set *of Ω_T and is denoted by $H(\Omega_T)$.*

From now on we take the notions of consistency and inconsistency defined for a conclusion set to apply to inheritable conclusion sets as well. We give some (more abstract) examples.

EXAMPLE 4.39 ___

Consider the taxonomy $T = (N, \Theta, \ll, \leq)$ where $I = \{x\}$ is the set of instances and $K = \{y_1, y_2, y_3\}$ is the set of classes; furthermore, Θ is defined by

$$\Theta = \{x[a_1 = c_1],\ y_1[a_2 = c_2],\ y_2[a_3 = c_3],\ y_3[a_3 = c_4]\}$$

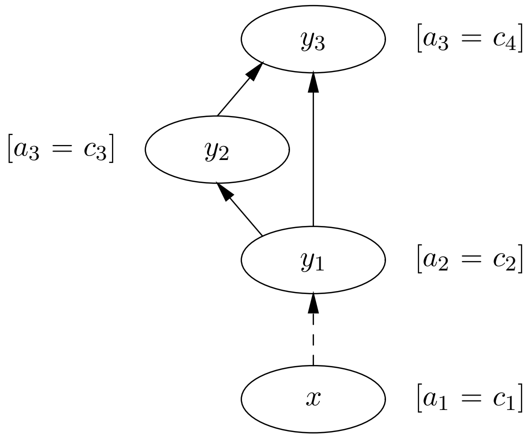


Figure 4.18: A taxonomy having a consistent inheritable conclusion set.

Herein a_1, a_2, and a_3 are distinct attribute names and c_1, c_2, c_3, and c_4 are different constants. In addition, the relation $\leq$ is defined by $y_1 \leq y_2$, $y_1 \leq y_3$ and $y_2 \leq y_3$; the function $\ll$ is defined by $x \ll y_1$. This taxonomy is depicted in Figure 4.18. The set of inheritance chains Ω_T consists of the following elements:

1. y_1
2. y_2
3. y_3
4. $y_1[a_2 = c_2]$
5. $y_2[a_3 = c_3]$
6. $y_3[a_3 = c_4]$
7. $y_1 \leq y_2$
8. $y_1 \leq y_3$
9. $y_2 \leq y_3$
10. $y_1 \leq y_2 \leq y_3$
11. $y_1 \leq y_2[a_3 = c_3]$
12. $y_1 \leq y_3[a_3 = c_4]$
13. $y_2 \leq y_3[a_3 = c_4]$
14. $y_1 \leq y_2 \leq y_3[a_3 = c_4]$

The conclusion set of Ω_T is equal to $C(\Omega_T) = \{y_1[a_2 = c_2], y_1[a_3 = c_3], y_1[a_3 = c_4], y_2[a_3 = c_3], y_2[a_3 = c_4], y_3[a_3 = c_4]\}$. Note that $C(\Omega_T)$ is inconsistent.

Consider the set Ω_T once more. We investigate which attribute-value specifications are in the inheritable conclusion set. As stated in the previous definition, an inheritance chain $\omega \in \Omega_T$ is inheritable if it is not precluded by any other chain from Ω_T. Since only a chain ending in an attribute-value specification can be precluded by another chain also ending in an attribute-value specification, examination of chains of such a form will suffice. So, we consider the following inheritance chains:

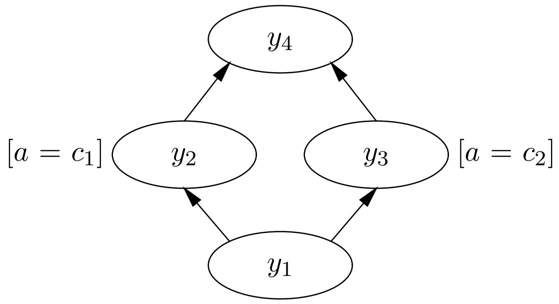


Figure 4.19: A taxonomy having an inconsistent inheritable conclusion set.

4. $y_1[a_2 = c_2]$
5. $y_2[a_3 = c_3]$
6. $y_3[a_3 = c_4]$
11. $y_1 \leq y_2[a_3 = c_3]$
12. $y_1 \leq y_3[a_3 = c_4]$
13. $y_2 \leq y_3[a_3 = c_4]$
14. $y_1 \leq y_2 \leq y_3[a_3 = c_4]$

Chain 12 is precluded by chain 11 because y_2 is an intermediary to $y_1 \leq y_3$. Furthermore, inheritance chain 13 is precluded by 5. The reader may verify that chain 14 is precluded by 11 as well. The inheritable conclusion set $H(\Omega_T)$ of Ω_T is therefore equal to $H(\Omega_T) = \{y_1[a_2 = c_2], y_1[a_3 = c_3], y_2[a_3 = c_3], y_3[a_3 = c_4]\}$. We conclude that the inheritable conclusion set is consistent.

In the next example, it will be shown that even if we apply preclusion it is still possible to obtain an inheritable conclusion set specifying contradictory information.

EXAMPLE 4.40

Consider the taxonomy $T = (N, \Theta, \ll, \leq)$ where the set I is empty, and $K = \{y_1, y_2, y_3, y_4\}$. The attribute-value relation Θ is defined by $\Theta = \{y_2[a = c_1], y_3[a = c_2]\}$ where a is an attribute, and c_1 and c_2 are distinct constants. Furthermore, the relation $\leq$ is defined by the following elements:

$$y_1 \leq y_2$$
$$y_1 \leq y_3$$
$$y_2 \leq y_4$$
$$y_3 \leq y_4$$

This taxonomy is depicted graphically in Figure 4.19. The inheritable conclusion set of the set of inheritance chains in this taxonomy is equal to $H(\Omega_T) = \{y_1[a = c_1], y_1[a = c_2], y_2[a = c_1], y_3[a = c_2]\}$. Note that $H(\Omega_T)$ is inconsistent.

From the foregoing example we have that the application of multiple inheritance using preclusion may still lead to the derivation of contradictory information. From now on, such a taxonomy will be called inconsistent.

Definition 4.20 *Let $T = (N, \Theta, \ll, \leq)$ be a taxonomy, and let Ω_T be the set of inheritance chains in T. Furthermore, let $H(\Omega_T)$ be the inheritable conclusion set obtained from Ω_T. The taxonomy T is said to be* consistent *if $H(\Omega_T)$ is consistent; otherwise T is said to be* inconsistent.

We conclude this section by introducing the refined notion of an inheritable extension of a set of inheritance chains.

Definition 4.21 *Let $T = (N, \Theta, \ll, \leq)$ be a taxonomy. Let Ω_T be the set of inheritance chains in T and let $H(\Omega_T)$ be the inheritable conclusion set of Ω_T. For each instance frame $x \in I$, the set $e_H(x)$ is defined by $e_H(x) = \{x[a = c] \mid x[a = c] \in \Theta\} \cup \{x[a = c] \mid x \ll y, y \in K, y[a = c] \in H(\Omega_T)$ and for all $c \neq d$, $x[a = d] \notin \Theta\}$ if $H(\Omega_T)$ is consistent; $e_H(x)$ is undefined otherwise. The* inheritable extension of Ω_T, *denoted by $E_H(\Omega_T)$, is defined by*

$$E_H(\Omega_T) = \bigcup_{x \in I} e_H(x)$$

if $H(\Omega_T)$ is consistent; $E_H(\Omega_T)$ is undefined otherwise.

EXAMPLE 4.41

Consider the taxonomy T from the second last example once more. The inheritable extension of Ω_T equals $E_H(\Omega_T) = \{x[a_1 = c_1], x[a_2 = c_2], x[a_3 = c_3]\}$.

4.3.2 Subtyping in graph-shaped taxonomies

In Section 4.2.4 we have discussed the subject of subtyping in tree-shaped frame taxonomies. In the present section, we extend the theory developed in that section to graph-shaped taxonomies. Recall that the subtype relation $\leq$ defines a partial order on a set of types. Before we treat subtyping in graph-shaped taxonomies in more detail, we review some properties of partially ordered sets.

Definition 4.22 *Let $S = \{t_1, t_2, \ldots, t_n\}$, $n \geq 1$, be a set on which we have a partial order $\leq$. An* upper bound *to a subset $X \subseteq S$ is an element $v \in S$ such that for each $x \in X$ we have that $x \leq v$; the* least upper bound *to X is an upper bound u for which we have that $u \leq v$ for each upper bound v to X. Similarly, a* lower bound *to a subset $X \subseteq S$ is an element $m \in S$ such that for each $x \in X$ we have that $m \leq x$; the* greatest lower bound *to X is a lower bound l for which we have that $m \leq l$ for each lower bound m to X.*

There may be more than one lower or upper bound to a subset of a partially ordered set, or even none at all. However, if lower and upper bounds do exist, then the least upper bound and the greatest lower bound are unique.

EXAMPLE 4.42

Consider the following set $S = \{\{1\}, \{1,2\}, \{1,2,3\}, \{1,2,3,4\}, \{2,3\}, \{2,3,4\}\}$ having elements which again are sets; on S we have the partial order induced by the set inclusion relation between its elements. We consider the subset $X = \{\{1,2,3\}, \{1,2,3,4\}\}$ of S. It will be evident that we have the following lower bounds to X: $\{1\}, \{1,2\}, \{1,2,3\}$ and $\{2,3\}$, since these sets are subsets of all elements of X. The greatest lower bound is equal to $\{1,2,3\}$: every other lower bound is a subset of this lower bound. In this example, we only have one upper bound, namely the set 1,2,3,4: each element of X is a subset of this upper bound. Note that this upper bound therefore at the same time is the least upper bound to X.

In Section 4.2.4 we have seen that type information in a correctly typed tree-shaped taxonomy may be considered to be a set of types which is ordered partially by the subtype relation $\leq$. For correctly typed graph-shaped taxonomies an even stronger property holds: in such taxonomies, the type information constitutes a so-called type lattice.

Definition 4.23 *A* type lattice *S is a set of types with a partial order $\leq$ such that for every two types $t_1, t_2 \in S$ there exist in S a least upper bound and a greatest lower bound. The greatest lower bound of t_1 and t_2 is denoted by $t_1 \wedge t_2$ and is usually called the* meet *of t_1 and t_2. Furthermore, the least upper bound of t_1 and t_2 is denoted by $t_1 \vee t_2$ and is usually called the* join *of t_1 and t_2. The* universal lower bound *of S, denoted by $\top$, is an element of S such that for each type $t \in S$ we have that $t \leq \top$. The* universal lower bound *of S, denoted by $\bot$, is an element of S such that for each type $t \in S$ we have that $\bot \leq t$.*

We shall see that for assigning a suitable meaning to type information, the types in a graph-shaped taxonomy should constitute a type lattice. First, we give an example of a type lattice.

EXAMPLE 4.43

Consider the graphical representation of a type lattice shown in Figure 4.20. This type lattice contains the type *vessel* having the type *blood-vessel* as a specialization. The *blood-vessel* type itself is considered to be subdivided into the types *vein* and *artery*. The type lattice furthermore specifies a kind of blood vessel, called *AV-anastomosis*, being a subtype of both the *vein* and *artery* type. Furthermore, the type lattice contains the type *blood* which describes the characteristics of blood. Two kinds of blood are distinguished: oxygen-rich blood and oxygen-poor blood having different characteristics (think for example of the range of the typical oxygen pressure in oxygen-rich and oxygen-poor blood, and of the colour of the blood which is typically darker in oxygen-poor than in oxygen-rich blood). The type *mixed-blood* describes blood having characteristics lying in between those of oxygen-poor and oxygen-rich blood. Now, note that from the type lattice we have *vein* $\vee$ *artery* = *blood-vessel*, in other words, the type *blood-vessel* is the meet of the types *vein* and *artery*. Note that the type *vessel* is an upper bound for the *vein* and *artery* types, but not the least one. Furthermore, the join of the types *vein* and *artery* is equal to the type *AV-anastomosis*, that is, we have *vein* $\wedge$ *artery* = *AV-anastomosis*.

Now recall from Section 4.2.4 that in a correctly typed tree-shaped taxonomy, the relation defined by the superclass links in the taxonomy coincides with the partial order $\leq$ on the set of

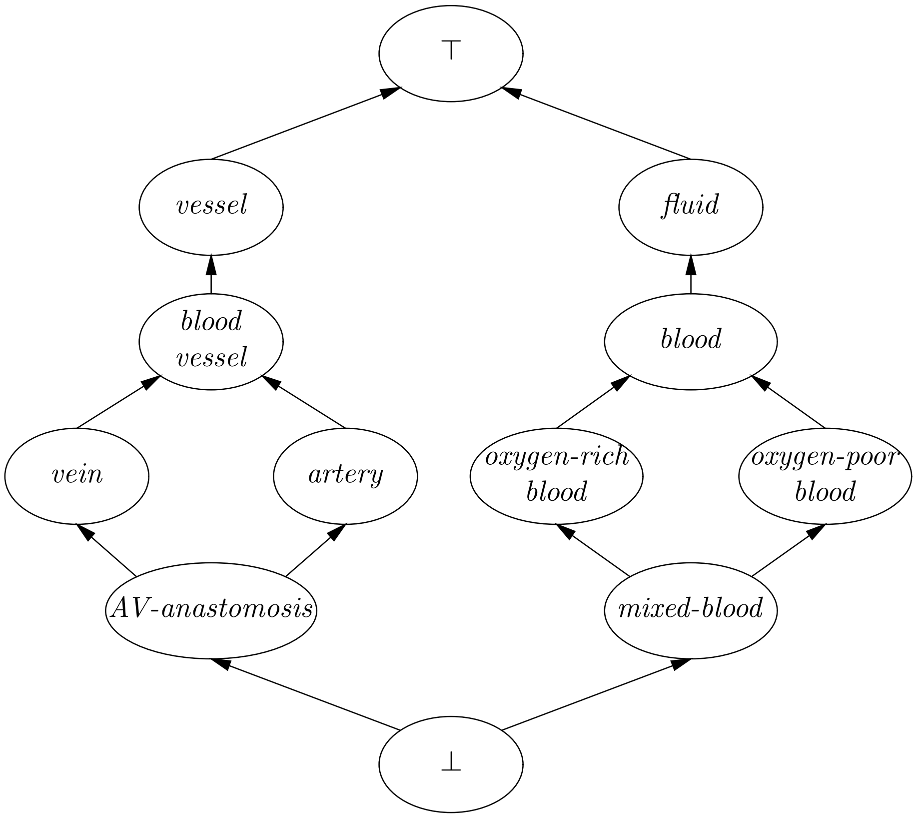


Figure 4.20: A type lattice.

types specified by it. Similarly, we have that we may view a graph-shaped taxonomy as a type lattice (including the predefined classes), and vice versa. Furthermore, recall the definitions of the notions of attribute sequence, domain, and type function, presented in Section 4.2.4. Informally speaking, we now take these notions to apply to graph-shaped taxonomies.

In Section 4.2.4 we interpreted types as sets. Here, we do so likewise: consider two types t_1 and t_2. We associated with these types the set $I(t_1)$ and $I(t_2)$, respectively, where $I(t_i) \subseteq U$, $i = 1, 2$, for some domain of discourse U. As we have mentioned before, in a graph-shaped frame taxonomy having the form of a lattice there exists a meet and a join for every pair of types. The set associated with the meet $t_1 \wedge t_2$ now has to satisfy the property $I(t_1 \wedge t_2) = I(t_1) \cap I(t_2)$; the set associated with the join $t_1 \vee t_2$ has to satisfy $I(t_1 \vee t_2) = I(t_1) \cup I(t_2)$. We conclude this section with an example giving a sketch of subtyping in a graph-shaped taxonomy.

EXAMPLE 4.44

Consider the type lattice from Figure 4.20 once more; we look upon it as a taxonomy. Suppose that the class frames *blood-vessel*, *vein*, *artery* and *AV-anastomosis* all contain an attribute-type specification concerning an attribute named *contains*, in which the specified type is one of the classes *blood*, *oxygen-rich-blood*, *oxygen-poor-blood*, and *mixed-blood*:

claiss *blood-vessel* **is**

superclass *vessel*;
contains : *blood*
end

class *vein* **is**
superclass *blood-vessel*;
contains : *oxygen-poor-blood*
end

class *artery* **is**
superclass *blood-vessel*;
contains : *oxygen-rich-blood*
end

class *AV-anastomosis* **is**
superclass {*artery*,*vein*};
contains : *mixed-blood*
end

Furthermore, the classes *blood*, *oxygen-rich-blood*, *oxygen-poor-blood* and *mixed-blood* specify a single attribute named *colour*:

class *blood* **is**
superclass nil;
colour : {*blue*, *dark-red*, *red*, *bright-red*}
end

class *oxygen-rich-blood* **is**
superclass *blood*;
colour : {*dark-red*, *red*, *bright-red*}
end

class *oxygen-poor-blood* **is**
superclass *blood*;
colour : {*blue*, *dark-red*, *red*}
end

class *mixed-blood* **is**
superclass {*oxygen-rich-blood*,*oxygen-poor-blood*};
colour : {*dark-red*, *red*}
end

In the present example, the set of attributes A in the frame taxonomy is equal to $A = \{contains, colour\}$. Let A^* be the set of attribute sequences. We now have eight domains to consider; they are denoted by D_1 up to D_8 for the classes in the order shown above. The domains for the first four classes are all equal to $\{\epsilon, contains, contains : colour\}$. The domains D_5 to D_8 for the classes starting with the class *blood* are equal to $\{\epsilon, colour\}$. The reader can easily verify that the properties required for these domains for subtyping hold. For simplicity's sake we now only consider the type functions associated with the first four classes. The type functions associated with the classes *blood-vessel*, *vein*, *artery*, *AV-anastomosis* are denoted by τ_1 up to τ_4, respectively. Recall from Section 4.2.4 that we have to verify that for all $a \in A^*$ we have:

$$\begin{aligned}
&\tau_2(a) \leq \tau_1(a)\\
&\tau_3(a) \leq \tau_1(a)\\
&\tau_4(a) \leq \tau_2(a)\\
&\tau_4(a) \leq \tau_3(a)
\end{aligned}$$

We only discuss some of these properties in detail. We start by noting that the type for the attribute *contains* in the class *blood-vessel* equals the join of the types *oxygen-rich-blood* and *oxygen-rich-blood* given for the classes *artery* and *vein*, respectively. Furthermore, the meet of the types *oxygen-poor-blood* and *oxygen-rich-blood* for the attribute *contains* in the classes *vein* and *artery* respectively is equal to *mixed-blood.* It will be evident that we have the following properties:

$$\begin{aligned}
&\tau_3(\mathit{contains}) \leq \tau_1(\mathit{contains})\\
&\tau_2(\mathit{contains}) \leq \tau_1(\mathit{contains})\\
&\tau_4(\mathit{contains}) \leq \tau_3(\mathit{contains})\\
&\tau_4(\mathit{contains}) \leq \tau_2(\mathit{contains})
\end{aligned}$$

Furthermore, note that we not only have that $\mathit{blood\text{-}vessel} = \mathit{vein} \vee \mathit{artery}$, but in addition that $\tau_1(\mathit{contains}) = \tau_2(\mathit{contains}) \vee \tau_3(\mathit{contains})$, since $\mathit{blood} = \mathit{oxygen\text{-}poor\text{-}blood} \vee \mathit{oxygen\text{-}rich\text{-}blood}$ that is: the subtyping of classes is extended from the classes to the attributes of the classes. Similarly, we have that $\tau_4(\mathit{contains}) = \tau_2(\mathit{contains}) \wedge \tau_3(\mathit{contains})$. Furthermore, we have that

$$\begin{aligned}
&\tau_4(\mathit{contains} : \mathit{colour}) \leq \tau_3(\mathit{contains} : \mathit{colour}\\
&\tau_4(\mathit{contains} : \mathit{colour}) \leq \tau_2(\mathit{contains} : \mathit{colour})\\
&\tau_4(\mathit{contains} : \mathit{colour}) = \tau_2(\mathit{contains} : \mathit{colour}) \wedge \tau_3(\mathit{contains} : \mathit{colour})
\end{aligned}$$

Checking the remaining properties is rather straightforward and is left to the reader. We conclude that the shown taxonomy is correctly typed.

4.4 Frames as a representation formalism

Frames (and semantic nets) provide a knowledge-representation formalism in which hierarchically structured knowledge can be specified in a natural way. Especially for the representation of knowledge of a descriptive nature, such as the knowledge concerning the cardiovascular system used in the examples in this chapter, the frame formalism appears to be highly suitable. The advantage of frames when compared to for example Horn clauses or production rules lies in the ease with which distinct types of knowledge can be distinguished and handled as such, and in the fact that an explicit hierarchical organization of knowledge is obtained.

In this chapter the only means of knowledge manipulation discussed was the method of inheritance. We stress once more that this knowledge-manipulation scheme in itself is not sufficient as an inference engine for all applications: is often turns out to be necessary to develop a more elaborate inference engine for the manipulation of frames in which inheritance only is part of a set of knowledge-manipulation methods. In chapter 7 we shall discuss some examples of such inference engines. For many non-trivial applications it will be necessary to use a enriched frame formalism, for example by procedural components. We have suggested before that an often employed hybrid knowledge-representation scheme is that of frames containing demons for invoking small sets of production rules. Instead of integrating frames with production rules, one could also think of a more declarative extension to the frame formalism for example by Horn clauses in combination with SLD resolution. This way a hybrid system is obtained that still has a clear declarative semantics. Most present-day frame-based systems are more alike programming languages than alike languages for knowledge-representation. A major disadvantage of many of these systems is the loss of a neat declarative semantics. Furthermore, working with these systems often requires a lot from the knowledge engineer such as a store of programming tricks.

Exercises

(4.1) Use the semantic net formalism to represent information concerning a problem domain you are familiar with. Try to define a neat semantics for the types of links you have used. What information can you derive from the net by property inheritance?

(4.2) Write a program that implements property inheritance in semantic nets. The program should be able to find for a specific object all properties that can be derived for it.

(4.3) Consider the following three frames:

class *computer-program* **is**
  superclass nil
end

class *expert-system* **is**
  superclass *computer-program*;
  synonym = *knowledge-system*;
contains = *expert-knowledge*
end

instance *MYCIN* **is**
  instance-of *expert-system*;
  implementer = *Shortliffe*
end

(a) Translate the knowledge specified in the three frames shown above into standard first-order logic with equality.

(b) Suppose that the following frame

instance *Internist-I* **is**
  instance-of *expert-system*;
  contains = *medical-knowledge*;
implementer = *Pople*
end

is added to the three frames given above. Discuss the problem that arises if we translate this frame together with the three frames shown above into standard first-order predicate logic with equality. Give a possible solution to the problem.

(4.4) Consider the following two frames:

> **class** *automobile* **is**
>   **superclass nil**;
>   *wheels* = 4;
>   *seats* = 4
> **end**
>
> **instance** *Rolls-Royce* **is**
>   **instance-of** *automobile*;
> *max-velocity* = *enough*
> **end**

Translate the knowledge specified in these two frames into a semantic net representation.

(4.5) Consider the algorithm for single inheritance with exceptions in a tree-shaped taxonomy discussed in Section 4.2.2 once more. Extend this algorithm in a straightforward manner to render it applicable to graph-shaped taxonomies. Show by means of an example that your algorithm may produce results that are incorrect from a semantic point of view.

(4.6) Develop an algorithm for N-inheritance and implement your algorithm in a programming language (for example Prolog, Lisp or Haskell).

(4.7) Consider the following frame taxonomy $T = (N, \Theta, \ll, \leq)$ where $N = (I, K, A, C)$ We have that $K = \{x, y, z\}$ is the set of classes, $I = \{w\}$ is the set of instances and $\Theta = \{x[a = 1], y[b = 2], z[b = 4]\}$ is the set of attribute-value specifications. The relation $\leq$ and the function $\ll$ are defined by:

$$
\begin{aligned}
&x \leq y \\
&z \leq x \\
&z \leq y \\
&w \ll z
\end{aligned}
$$

First, determine the set Ω_T of inheritance chains in T. Subsequently, compute the conclusion set and inheritable conclusion set of Ω_T. Is the taxonomy T consistent? If so, what is the inheritable extension of Ω_T?

(4.8) Consider the following taxonomy $T = (N, \Theta, \ll, \leq)$ where $N = (I, K, A, C)$. We have that $K = \{u, x, y, z\}$ is the set of classes, $I = \{w\}$ is the set of instances and $\Theta = \{u[a = 1], x[b = 2], y[c = 10], z[c = 20]\}$. The relation $\leq$ and the function $\ll$ are defined as follows:

$$
\begin{aligned}
&x \leq y \\
&y \leq z \\
&u \leq x \\
&u \leq y \\
&w \ll u
\end{aligned}
$$

Answer the same questions as in Exercise 4.7.