This chapter describes classification as a process that selects solutions that match data. Classification models are used for a wide range of tasks in knowledge systems.

7

Classification

7.1 Introduction

To classify something—an unknown object, phenomenon, pattern, measurement, or anything at all—is to identify it as a member of a known class. For example, a field biologist might use several features such as shape, size, color, and so on to identify a species of plant or an animal. In this case, the sample plant or animal is the unknown thing and the species is a known class. **Classification problems** begin with data and identify classes as solutions. Knowledge is used to match elements of the data space to corresponding elements of the solution space.

An essential characteristic of **classification**, as we use the term, is that it *selects* from a predefined set of solutions. This does not mean that every set of data has a unique solution or even that every set of data has a solution at all in the solution set. It just means that the process of classification involves the matching of data to a fixed set of known possible solution classes. Some data match no solutions at all.

7.1.1 Regularities and Cognitive Economies

Classes express regularities. All members of a class share certain properties. For example, all diseases of a given class may be treatable in a common way or may have in common certain kinds of causes.

Classification affords economies in cognition. Cognitive scientists have noticed that much of our mental commerce with an environment deals with classes of things rather than with unique events and objects. For example, classifying symptoms as characteristic of a particular disease can be a crucial part of diagnosis. Classifying signal patterns can be a crucial part of signal processing, in which signal elements are identified. Classifying goals and requirements can

be a crucial part of a planning process, for selecting individual actions or skeletal plans. In an order-entry task, customer orders may be classified prior to detailed processing. In knowledge engineering, classification models are used in many kinds of tasks.

To do any of these tasks an agent must reason about things that are different in their observable or given characteristics, but that can be clustered into categories for further processing. This factors a task into two steps: classify and process. Intuitively, the rules for reasoning about classes are simpler than rules about the myriad of instances.

Classes are defined in terms of necessary conditions. For example, we could define the class of "portable computers" to mean all computers that can be carried around with one hand by an average-size person. If we are presented with a computer and can pick it up easily with one hand and we are of average size, then it is portable by our criterion, even if it cannot be powered by batteries or does not fit into a briefcase.

Determination of an appropriate set of distinctions and categories is an important part of making sense of a domain and often a crucial part of building a knowledge system. However, determining the definitions of classes is not a part of classification itself. The identification of regular and meaningful patterns is a problem with its own methods, such as cluster analysis, and is outside the scope of this chapter. In classification tasks, a set of classes is given, together with necessary conditions for identifying them.

Over the past few years many knowledge systems have been built for which classification is central. Several of these systems work through recognizable phases of data abstraction, knowledge-based mapping to a taxonomy of pre-enumerated solutions, and refinement within the solution taxonomy. Each phase may have several steps, which means there may be multiple levels of data abstraction, multiple levels of associative mapping between data and intermediate solutions, and multiple levels of solution refinement. Figure 7.1 shows the flow of inferences for a bottom-up method starting with data. In other methods, the inferences are more mixed and may involve reasoning back from the solution space to the data space. Nonetheless, the description in Figure 7.1 is a useful starting framework from which we shall specialize a few variations in the following sections.

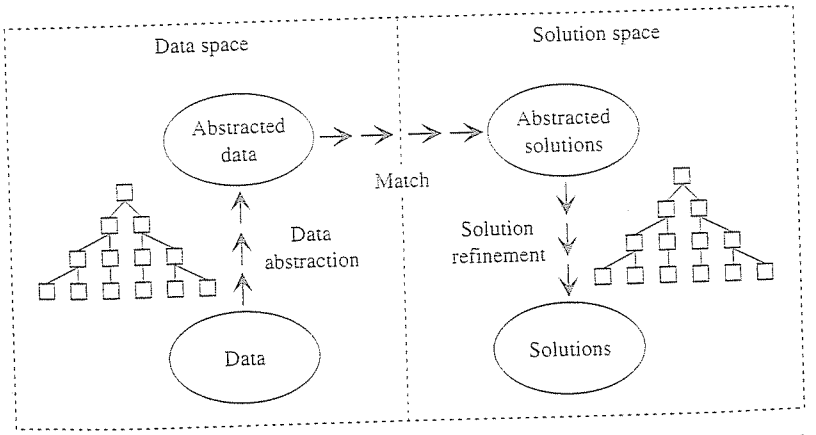


FIGURE 7.1. An exemplar of a classification domain. There is a single data space and a single solution space, with abstractions in each. (Adapted from Clancey, 1985, page 296.)

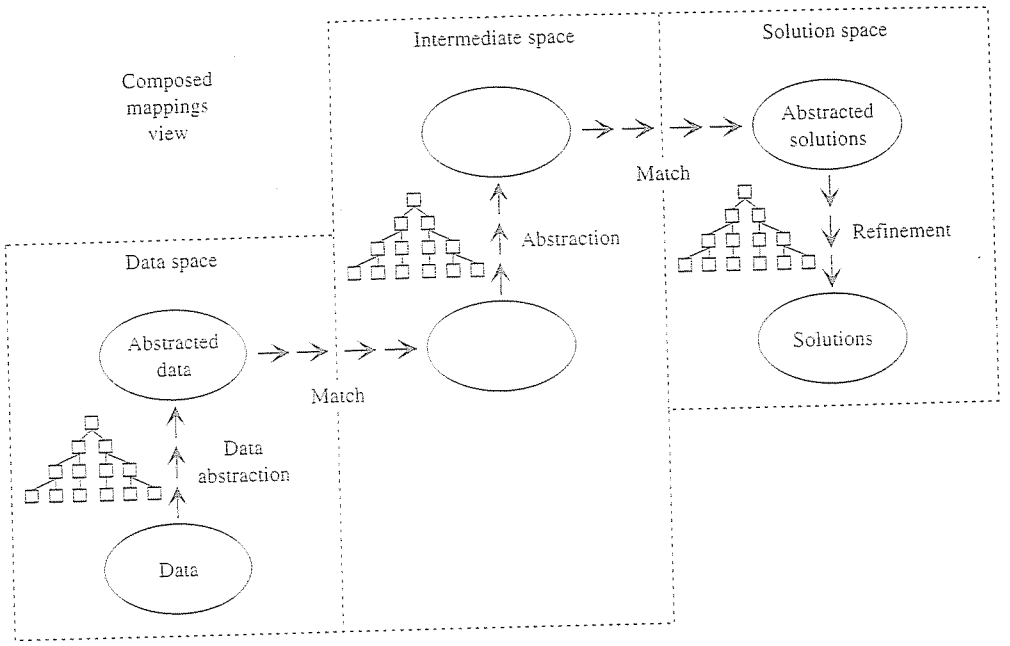
The pattern of search spaces suggested by Figure 7.1 is called **heuristic classification** to emphasize the common use of imperfect and heuristic matching to assign data to classes. Some authors prefer the term **hierarchical classification**, to emphasize the importance of reasoning by levels as well as the computational advantages that arise from indexing actions over classes of situations rather than over individual situations. The term **heuristic selection** is used to emphasize the noncombinatorial nature of discrete choice. Our approach draws heavily on a formulation and analysis by Clancey (1985). For simplicity, we refer to the model as classification.

All classification models map data to a set of predefined solutions. Figure 7.2 shows a variation of the domain in which there is an intermediate space between the data space and the final solution space. There are two common ways to present **multistage classification models**. The **composed mappings view** emphasizes that the same relations of abstraction, match, and refinement occur in the different stages. This view presents those relations the same way in each stage, with abstraction indicated by up arrows, matching indicated by right arrows, and refinement indicated by down arrows. The **intermediate spaces view** emphasizes that classification can link intermediate spaces of different character. This view makes it easier to show the differences between the spaces. For example, the intermediate space may have a causal or time-based model. The intermediate spaces view also makes it easier to show how multiple elements in one space may map to the same element in another space. Diagrams like these are useful for visualizing particular classification tasks in terms of their search spaces.

The first two phases of classification in Figure 7.1—data abstraction and match—are themselves special cases of classification. Data abstraction is a simple form of classification in which the data space is a subset of the solution space. The features of the solution classes are a subset of the features of the given data. In data matching, the data and solutions can be in different spaces. This case arises when the given data are indirectly linked to the phenomena being classified. For example, in medical diagnosis the solutions may be disease states and the data may be patient history data, such as known prior diseases and treatments, and laboratory data, such as blood pressure or analyses of body fluids. The relations that link data to solutions may involve causal chains. Solution classes are conveniently arranged in a generalization taxonomy. In other variations, one or more of the phases may be missing, such as cases with no significant data abstraction or no solution refinement.

In characterizing classification, we usually exclude the case where the solution classes are known ahead of time but are generated rather than listed explicitly. However, when the generation process is trivial and the number of solutions is small, this issue is not so important. The generator approach is not called classification when it is more complex than the classification process, such as when the solution space is a power set of some basis set, made up of combinations of the basis elements. Models for configuration, an example of such problems, are taken up in Chapter 8. Similarly, many diagnosis problems involve more than classification because they require potentially combinatorial reasoning about composite hypotheses and determination of what additional data to acquire. These issues arise in the examples in this section and we mention some simple approaches. In general, computational models for combinatorial reasoning and for selecting data to discriminate among solutions are more complex than classification itself.

The following sections discuss variations on and specializations of classification. Section 7.1 analyzes classification domains. It presents a computational model of classification, asking what makes it difficult and what role different kinds of knowledge and assumptions play. Section 7.2 presents several examples of knowledge systems based on classification. The systems built



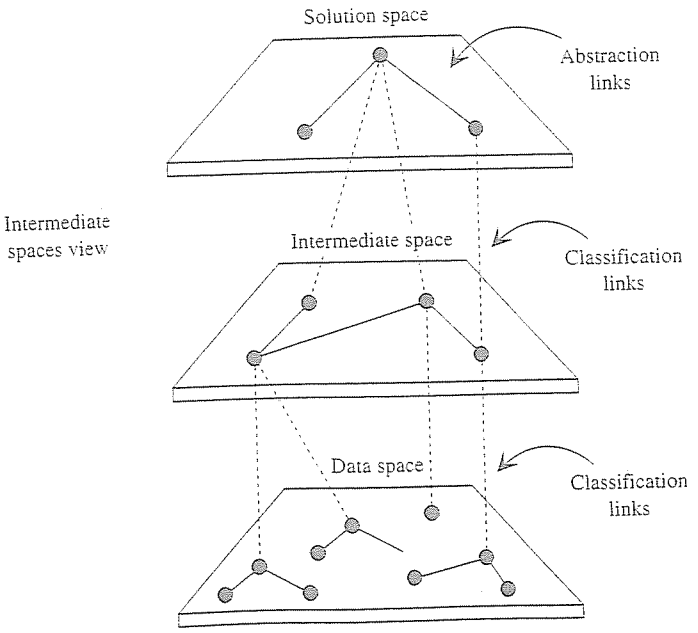


FIGURE 7.2. Two views of multistage classification models. The composed mappings view emphasizes the case where the same classification relations are used in each stage. The intermediate spaces view emphasizes the case where the intermediate spaces have different properties.

for the applications have both striking similarities and striking differences. Section 7.3 presents several specialized methods for classification, showing how they depend on different assumptions and use different knowledge.

7.2 Models for Classification Domains

In this section we consider more precisely what a classification model is, what makes classification tasks difficult, and what roles knowledge and various assumptions play.

7.2.1 A Computational Model of Classification

We begin by developing a simple finite set model for classification problems that we call a **conjunctive classification model**. This simple model is rich enough to illustrate the important phenomena of classification domains. In the following we also consider several simple extensions that do not violate the spirit of the simple model.

A conjunctive classification model has a **data space**, D, which is a finite set with elements $\{D_i\}$, and a **solution space**, S, which is a finite set with elements $\{S_j\}$. The $\{D_i\}$ can be assigned data values, representing the data to be classified in a particular problem instance. We are given values for some or all of the $\{D_i\}$. A datum can take values from the set $\{0, 1\}$. For convenience, we extend the data value notation by using D_i = "?" to indicate that no value is known for the datum. We refer to the collective values for the $\{D_i\}$ as a data "vector." The $\{S_j\}$ stand for the predetermined solution classes that are candidate solutions for any particular problem.

For every solution candidate S_j in S there is a pattern that specifies necessary conditions for consistency of solutions with data. These patterns are defined by a covering relation defined as follows:

$C(S_j, D_i) = 1$	means $S_j \Rightarrow D_i = 1$ That is, S_j cannot be a solution if D_i is 0. It is consistent with $D_i = 1$.
$C(S_j, D_i) = 0$	means $S_j \Rightarrow D_i = 0$ That is, S_j cannot be a solution if D_i is 1. It is consistent with $D_i = 0$.
$C(S_j, D_i) = ?$	if the value of datum D_i is irrelevant to the consistency of solution S_j.

Given a covering relation, we can ask several different questions about a given pattern of data. For example, we can ask which solution classes are **inconsistent** with a given data vector. A solution candidate S_j is inconsistent with the data if at least one value is known to be inconsistent with it. A solution candidate S_j is **consistent** with the data if no known values are inconsistent with it. We say candidate S_j **covers** the datum D_i in either of two cases: (1) $D_i = 1$ and $C(S_j, D_i) = 1$ or (2) $D_i = 0$ and $C(S_j, D_i) = 0$. Note that a candidate is judged consistent when no data are known that bear on it one way or the other, or if only some of the relevant data are known and all known data are consistent with it. We say a candidate **matches** the data if all the data relevant to S_j are known and the values are all consistent with the candidate. If a candidate matches all of its relevant data, we also say that the candidate **explains** the data.

This model provides an explicit way to "rule out" a candidate. Specifically, a candidate can be ruled out if any data have been observed that are inconsistent with it. In the basic model, there is no way to "rule in" a solution.

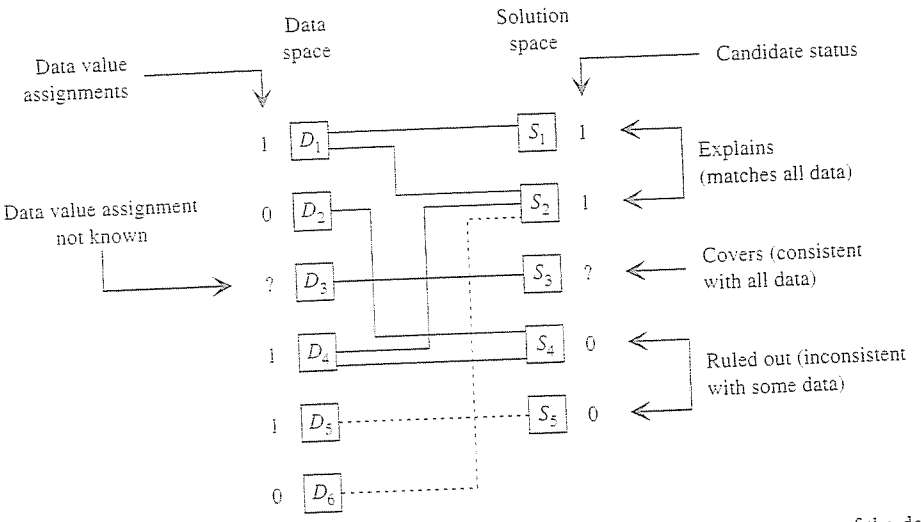


FIGURE 7.3. Graphical notation for conjunctive classification problems. The D_i are elements of the data space and the S_j are elements of the solution space. A solid line from D_i to S_j indicates that S_j is consistent with $D_i = 1$. A dashed line from D_i to S_j indicates that S_j is consistent with $D_i = 0$.

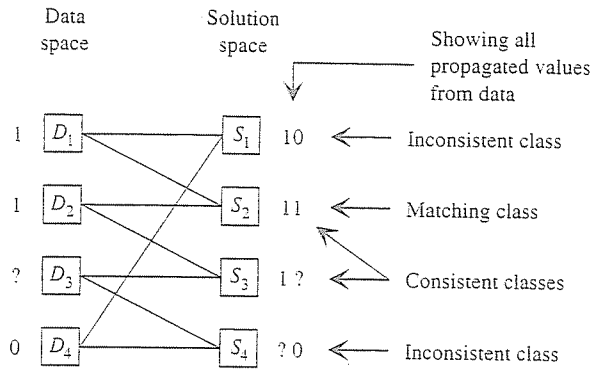
It is convenient to define a graphical notation for describing conjunctive classification problems. In Figure 7.3, the D_i and S_j are indicated by boxes on the left and right, respectively. A solid line from D_i to S_j indicates that the candidate solution S_j is consistent with the value assignment $D_i = 1$. A dashed line from D_i to S_j, used less often in our examples, indicates that the candidate solution S_j is consistent with the value assignment $D_i = 0$. The absence of a line from D_i to S_j indicates that the data value is irrelevant to the consistency of candidate solution S_j, which means S_j is consistent with any value assignment to D_i.

The symbols to the left of the D_i boxes are the given values assigned to the data. The numbers to the right of the S_j boxes indicate the status of the solution candidate, where 1 means the solution class is consistent, 0 means the solution class is inconsistent, and ? means there is inadequate evidence to decide either way. In Figure 7.3, candidate solutions S_1 and S_2 match or explain the data, candidate S_3 is consistent with or covers some data but does not match them, and candidates S_4 and S_5 are inconsistent.

One feature of this notation is that the process of determining whether a candidate is consistent requires only propagating values from the data elements along the lines to the solution candidates. For solid lines, the data values are propagated as they are given. For dashed lines, the data values are inverted as they are propagated, which means 1 is changed to 0 during propagation and 0 becomes 1. For both kinds of lines, ? propagates to ?. The status of a solution candidate S_j can then be determined as follows:

If *all* values propagated to S_j are 1's, then S_j matches or explains the data.
If *any* value propagated to S_j is 0, then S_j is not consistent with the data and S_j is ruled out.
If *all* values propagated to S_j are 1's and ?'s, then S_j is consistent with or covers the data.

We use such diagrams in the rest of this section to illustrate classification phenomena.



Covering relation

	S_1	S_2	S_3	S_4
D_1	1	1	?	?
D_2	?	1	1	?
D_3	?	?	1	1
D_4	1	?	?	1

This table indicates the consistent data patterns for each possible classification.

FIGURE 7.4. An example conjunctive-classification problem. All propagated values are shown. For the given data, only solution S_2 matches. The table to the right of the figure characterizes the data patterns that are accepted or recognized by each solution class. This table is a chart of the relation C and is equivalent to the graphical representation. It is constructed from the graph by placing a 1 in each row where there is a line from the solution to the datum and a ? in every other row.

Figure 7.4 presents another example of a conjunctive classification problem. You are encouraged to use this example to familiarize yourself with the propagation process and with judging which solution candidates are consistent. In this problem the data space and the solution space both have four elements. Each element of the data space is connected to two elements of the solution space, and vice versa.

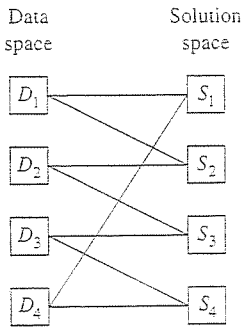
7.2.2 *Model Variations and Phenomena*

Interactions between data patterns and the covering relations give rise to a variety of phenomena in classification problems. In this section we consider some of these phenomena and others that arise in variations of our basic model.

No Solutions, Multiple Solutions, and Composite Solutions

Given a covering relation, the prototypical situation is that a data vector matches a single class. However, it is also possible that some data vectors will match no classes at all, and other vectors may match more than one class. For the covering relation in Figure 7.5, every data pattern with more than two 1's matches more than one solution. For example, the data vector (1 1 1 0) matches S_2 and S_3. Every solution class in Figure 7.5 requires that two of the data values be 1. Therefore, every pattern with fewer than two data values of 1 matches no solution class. For example, the data vector (1 0 0 0) matches no solutions.

Many classes can match or be consistent with a given data vector. Some tasks based on classification models, however, make use of additional criteria in determining solutions. This raises the general issue about how multiple matches or multiple consistent classes are to be interpreted as solutions in a larger task. Consider a case in medical diagnosis where two of the diagnostic hypotheses are "measles" and "a broken leg." For this discussion, we assume that the symptoms of measles are fever and red spots on the body. The symptoms of a broken leg are



Some nonsolutions

D_1	0	1	0	0	0
D_2	0	0	1	0	0
D_3	0	0	0	1	0
D_4	0	0	0	0	1

Some data with multiple classes

	S_3 & S_4	S_1 & S_4	S_1 & S_2	S_2 & S_3
D_1	0	1	1	1
D_2	1	0	1	1
D_3	1	1	0	1
D_4	1	1	1	0

FIGURE 7.5. An example of a conjunctive-classification problem in which some data value assignments match no classes at all and others match more than one class. Every pattern with fewer than two data values of 1 matches no solution at all, and every data pattern with more than two 1's matches more than one solution.

localized pain and characteristic swelling of the leg. We presume that X-ray data are not available. It also is quite possible, if unlikely, that a patient could have both afflictions at the same time.

In this case, a natural diagnostic conclusion is that both afflictions are present. A solution in this case takes the form "*A* plus *B*," or "measles plus broken leg." We call such interpretations **composite solutions**. Composite solutions bundle multiple classes as single solutions.

Continuing the example, the symptoms "red spots and fever" might be accounted for not only by measles, but also perhaps by chicken pox, heat rash, mumps, or other diseases. Similarly, the symptoms "pain and swelling in the leg" might be explained by other specific afflictions, such as a torn muscle or various other strains not involving a fracture of a leg bone. Thus, the following are among the alternative or competing diagnostic hypotheses: "measles plus broken leg," "measles plus torn muscle," "chicken pox plus broken leg," "chicken pox plus torn muscle," "heat rash plus broken leg," or "heat rash plus torn muscle."

As long as the number of combinations to be considered stays small, the composite solution case seems like a small extension to classification. However, when a search must potentially consider *all* combinations of classes in a solution space, the process of generating the combinations of classes typically overwhelms a classification process. We discuss such models for diagnosis in Chapter 9.

Whether solutions are defined as single classes or as sets of classes is the first of several issues in interpreting the results of classification. Figure 7.6 presents alternative policies for data interpretation. Solutions can be singleton or composite, one or many solutions can be reported, and different rules can be used for evaluating candidates with regard to each other and the evidence. The list in Figure 7.6 is not exhaustive, but it does represent different classification policies.

Hierarchical Solution Spaces

When data are presented to a classifier, classes that are inconsistent with the data can be ruled out. When the data are very sparse, however, a data vector typically does not narrow the choices very much because we cannot rule out solutions as inconsistent on the basis of missing data. For

Form of Solutions	
Singleton solutions	Each solution is made up of exactly one class.
Composite solutions	Multiple classes can be combined as a set to form a single solution. This substantially extends and complicates the basic classification model.
Completeness of Search	
Satisficing search	Only one solution (singleton or composite) is reported even if multiple ones are possible.
Exhaustive search	All competing solutions (singleton or composite) are reported.
Termination Criteria	
Resource-limited search	Search ends when budgets for time, observations, expense, or risk are exhausted. Whatever candidates are under consideration are reported.
Inclusion Criteria	
Positive coverage	A solution is included if it covers some data and is not inconsistent with any data.
Conservative inclusion	A solution is included unless it is ruled out by evidence (even if it does not explain *any* data).
Complete explanation	To be included a solution must explain all the acquired data.
Probability threshold	To be included a solution must be consistent with observations and have a prior probability above some threshold.
Ranking Criteria	
Reasoning by elimination	A solution is accepted only when all other candidates have been ruled out by inconsistency with data.
Preponderence of evidence	One solution dominates another if the former covers more of the acquired evidence.
Minimal sets	If two composite solutions cover the acquired data and the set of classes included in one composite solution is a subset of the other set, then only the smaller composite solution is reported. This is also known as a parsimony rule.
Single-solution assumption	Solutions are mutually exclusive. Operationally, this means that once an explanatory solution is found, other solutions are immediately ruled out.
Ranked solutions	An additional function is used to rank alternative candidates, based on a priori information.

FIGURE 7.6. Some goals for combining classes and evaluating solutions that can be imposed on classification models.

example, the data vector (? 1 ? ? ? ? ?) in Figure 7.7 is consistent not only with solution S_2, but also with all the other classes. Judgment depends on further data. In this case and also when there are more possible data than can reasonably be acquired all at once or at the beginning, the data can be acquired incrementally. In such iterative scenarios, it is sometimes possible to arrange the classes so data acquisition is systematic and exploits hierarchical relations among the classes.

Sometimes classes can be arranged so sparse data initially available are enough to determine rough classes and hierarchical elimination. When data are added, increased specificity is possible, corresponding to a descent through the levels of a hierarchy. Figure 7.7 illustrates a

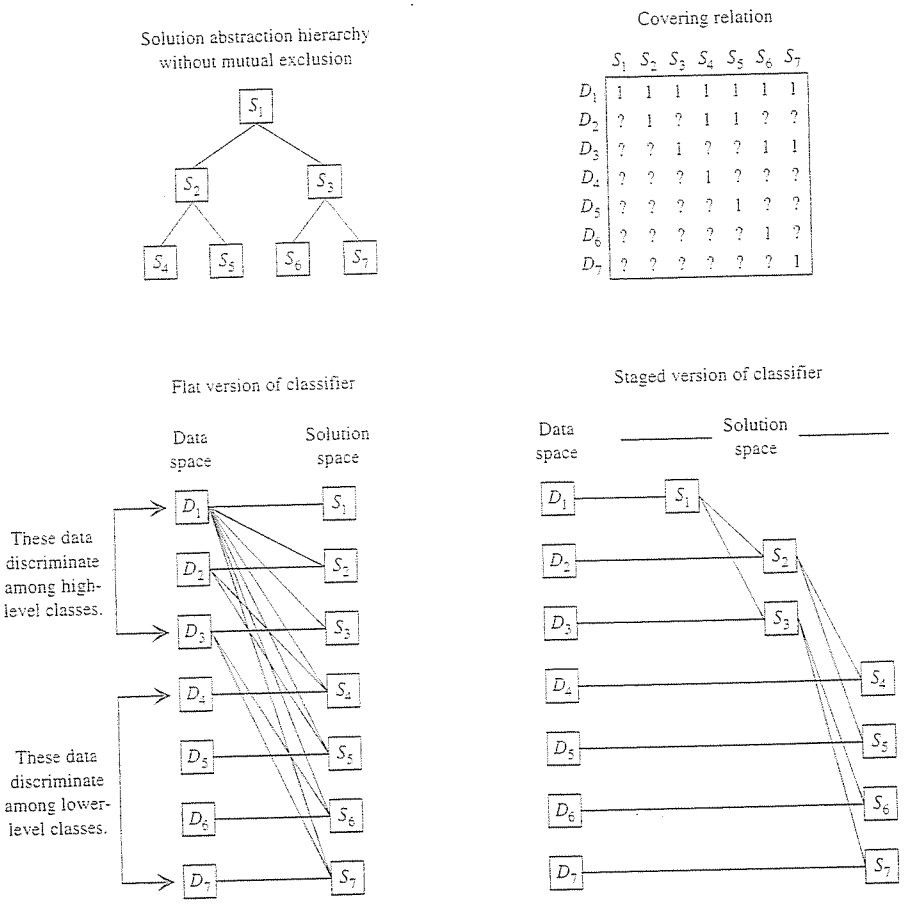


FIGURE 7.7. An example of data patterns for a hierarchical solution space. In a hierarchically organized solution space, the highest nodes designate the most abstract classes. The more specialized subclasses must satisfy all the criteria of their superclasses but can have additional criteria as well. In this example, the highest class, S_1, is characterized by a single data element, D_1. Subclasses S_2 and S_3, respectively, are differentiated by the data D_1 and D_2. In this way, the deeper nodes in the tree are differentiated by increasing amounts of data.

simple hierarchy of classes in which each successive subclass is distinguished from its superclass by requiring a match with an additional data element. By virtue of this arrangement, any data assignment that satisfies solution class S_5 must also satisfy S_2 and S_1. This nesting of superclasses can be reflected conveniently in the graphical notation by organizing the nodes and propagation in stages, corresponding to the levels of an abstraction hierarchy. The propagation rules need only be modified to keep propagating the values along all the connected arcs. Thus, in the example in Figure 7.7, the data vector (1 ? ? ? ? ? ?) is compatible with all the classes but can be summarized simply as S_1. More importantly, the data vector (0 ? ? ? ? ? ? ?) is inconsistent with all the classes and the data vector (1 0 ? ? ? ? ?) is consistent only with S_1, S_3, and all of S_3's subclasses.

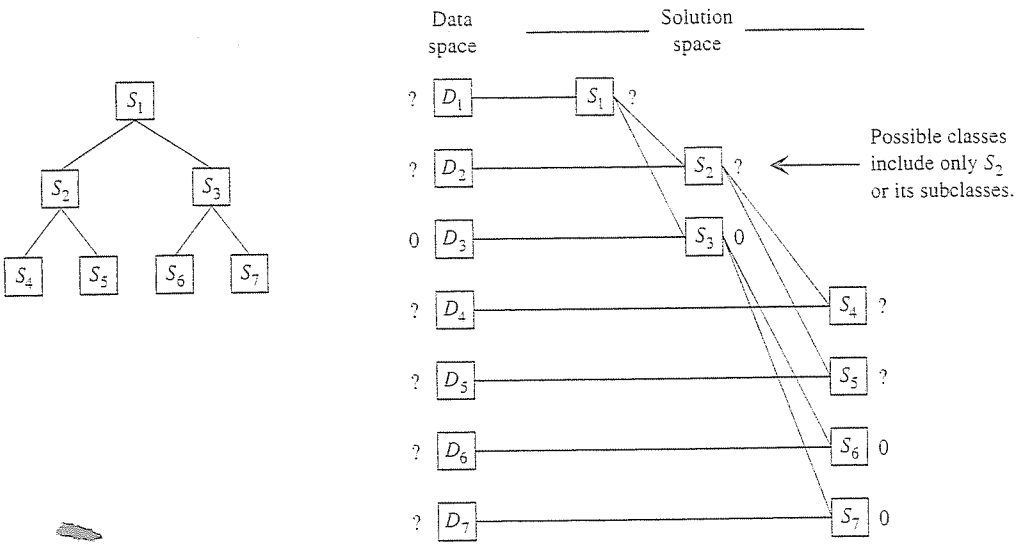


FIGURE 7.8. Hierarchical elimination in classification. The data vector (? ? 0 ? ? ? ?) is inconsistent with solution class S_3 and its subclasses. Assuming that all the members of S_1 belong either to S_2 or S_3, we can conclude that the only classes possibly consistent with the data are S_2 and its subclasses.

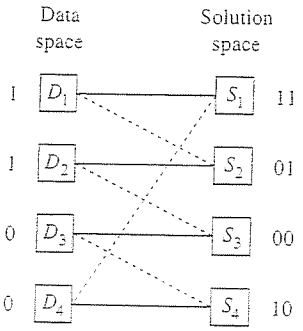
Pursuing the logic of hierarchical elimination a bit further, Figure 7.8 shows for the same classes as Figure 7.7 that the data vector (? ? 0 ? ? ? ?) is inconsistent with S_3 and all of its subclasses. Since S_2 and S_3 are the only subclasses of S_1, we would like to conclude that the classification must be S_2 or one of its subclasses. This hierarchical elimination relies not only on the topology of the solution hierarchy, but also on the requirement that *each solution classe designates the union of the designations of its subclasses*. In this case, since S_1 only has subclasses S_2 and S_3, if S_3 is ruled out only S_2 is left. This step would not be valid if there was an implicit set of "other" members of S_1 belonging to neither S_2 nor S_3. In the following we refer to class nodes satisfying this requirement in a taxonomy—class nodes that designate the union of the designations of their subclasses—as **encompassing class nodes**.

The links from the solution nodes to other solution nodes in Figure 7.8 imply that the nodes are all encompassing. This requirement on the interpretation of the graph could be built into the interpreter or indicated by additional annotations on graphs if an interpreter needed to reason with graphs with and without this meaning.

Reasoning by Elimination and Mutually Exclusive Solutions

In the basic conjunctive classification model, there is no way to "rule in" a solution. The only way to eliminate a solution is to observe a 0 where a 1 is required, or to observe a 1 where a 0 is required. In the following we consider a way to augment this with reasoning by elimination.

Consider a medical diagnostic case based on a classification model. Suppose simple tests can be performed that rule out all the diagnostic candidates but one. If we then conclude that the sole remaining candidate is the diagnostic solution, we have sanctioned reasoning by elimination. To paraphrase Sherlock Holmes, "Having eliminated the impossible, what is left must be the solution no matter how improbable."



Covering relation

	S_1	S_2	S_3	S_4
D_1	1	0	?	?
D_2	?	1	0	?
D_3	?	?	1	0
D_4	0	?	?	1

FIGURE 7.9. Mutually exclusive solutions. The classes S_1 and S_2 are mutually exclusive in this set. Every member of S_1 must have D_1 assigned to 1 and every member of S_2 must have D_1 assigned to 0.

In reasoning by elimination, negative evidence for one class is used as positive evidence for another. Mutual exclusion is a dual to this. With mutual exclusion, positive evidence for one class is used to rule out a second mutually exclusive class. In hierarchical networks such as that in Figure 7.7, evidence against a node like S_3 can be used to rule out all of its subclasses. If in addition we are given that S_2 and S_3 are mutually exclusive, then we could carry out additional reasoning by elimination. Mutual exclusion admits ruling out a class whenever a class mutually exclusive to the first class matches the data. For example, if S_4 matches the data then we can rule out S_3. Ruling in S_4 would admit S_2 by virtue of the hierarchy, which would then rule out S_3.

Some classes are mutually exclusive by virtue of their specific classification criteria, without adding any relations to the interpretation of the graph. In Figure 7.9 the solution classes S_1 and S_2 are mutually exclusive, as are S_2 and S_3, S_3 and S_4, and S_4 and S_1, respectively. Each mutually exclusive class places incompatible requirements on the data assignments.

Mutual exclusivity is not transitive. The data vector (1 0 1 0) in Figure 7.9 satisfies both S_1 and S_3, even though both S_1 and S_3 are mutually exclusive of S_2. In this model, whenever classes S_m and S_n are mutually exclusive, there must be some data value D_i such that $C(S_m, D_i) = 1$ and $C(S_n, D_i) = 0$. In models of classification with more flexible conditions relating data values to solution classes, the conditions for mutual exclusion are more general. There must be some combination of assigned values to data that is consistent with one class but not the other.

7.2.3 *Pragmatics in Classification Systems*

In this section we consider extensions to our basic classification model and some simple approaches for deciding what new data to acquire.

Generalizing Data Domains and Classification Predicates

Most of the example domains and knowledge systems in this chapter require extending the conjunctive classification model in two ways. First, the set of possible data values needs to be extended from the set {0, 1} to be a set of values appropriate for the domain and its instrumentation. Second, the way that conditions about data can be combined needs to be extended. In con-

junctive classification, a class is matched if the first condition is satisfied *and* the second is satisfied *and* the third is satisfied and so on. That is why it is called conjunctive classification. There are other ways to combine data, such as by allowing disjunction or exclusive ORs. A very general approach is to allow the conditions on data to be combined by any logical predicate. Since the conjunctive classification model we have been using is a special case of the more general model, the phenomena we have discussed in the preceding section are properties of the general model as well.

Some applications use other extensions of the classification method beyond this. The most common variation is the addition of some means for incorporating estimates of prior probabilities, as in the MORE, MYCIN, and PROSPECTOR applications. Appropriate models of uncertainty are discussed in Chapter 6. In a classification context, the classification predicates determine which solutions match which data. The role of the probability estimates in classification processes is to sharpen the results of a match, using statistical knowledge to increase overall selectivity and establishing that some data or some solutions are unlikely when compared with others.

Data Abstraction

All the conjunctive classification examples we have considered so far assume that the matching criteria for selecting classes are expressed directly in terms of the raw data. As a practical matter, it is much more typical to abstract the data first. As suggested in the introduction to hierarchical classification, a data abstraction step is itself a special case of classification in that it abstracts combinations of data to "abstract data classes." These classes, however, differ from hierarchical solution classes in that they do not correspond to solutions to the larger classification problem. Abstracted data and raw data may be combined in matches for the "second" part of the classification model.

Figure 7.10 gives an example of a model having both data abstraction and solution refinement. It is a close-up illustration of the inside connections of a horseshoe diagram. In this example, some of the raw data are abstracted in one or more stages and some are used directly in the classification model.

In the simplest case, raw data elements are used directly in the classification problem. In Figure 7.10, R_{10} and R_{11} become D_6 and D_7. In contrast, abstraction changes the form of the raw data. R_7 changes its form to AD_4 before being used as D_3. An example of this from a medical domain might be that the raw data "white blood cell count is 2,000" is abstracted to "low WBC." A raw data element might change its form to more than one abstraction, depending on the measured value. The raw datum R_8 is abstracted to either AD_5 or AD_6 before being used. An example from a medical domain would be the abstraction of the raw datum "temperature reading" to either "normal temperature" or "fever temperature."

Multiple raw data elements can also be combined during abstraction. In Figure 7.10, abstract datum AD_3 combines two raw data elements, R_5 and R_6. In a real domain, the abstract datum might be "high pressure" and the two data elements might correspond to two different points of measurement.

Raw data can be abstracted through multiple levels. R_1 and R_2 are combined to form abstracted data AD_2, which is combined with AD_3 to form AD_1, which is then used directly as D_1. The combinators for data elements are not limited to conjunction. The point of this is that there is substantial variety in the forms of data abstraction used in different models.

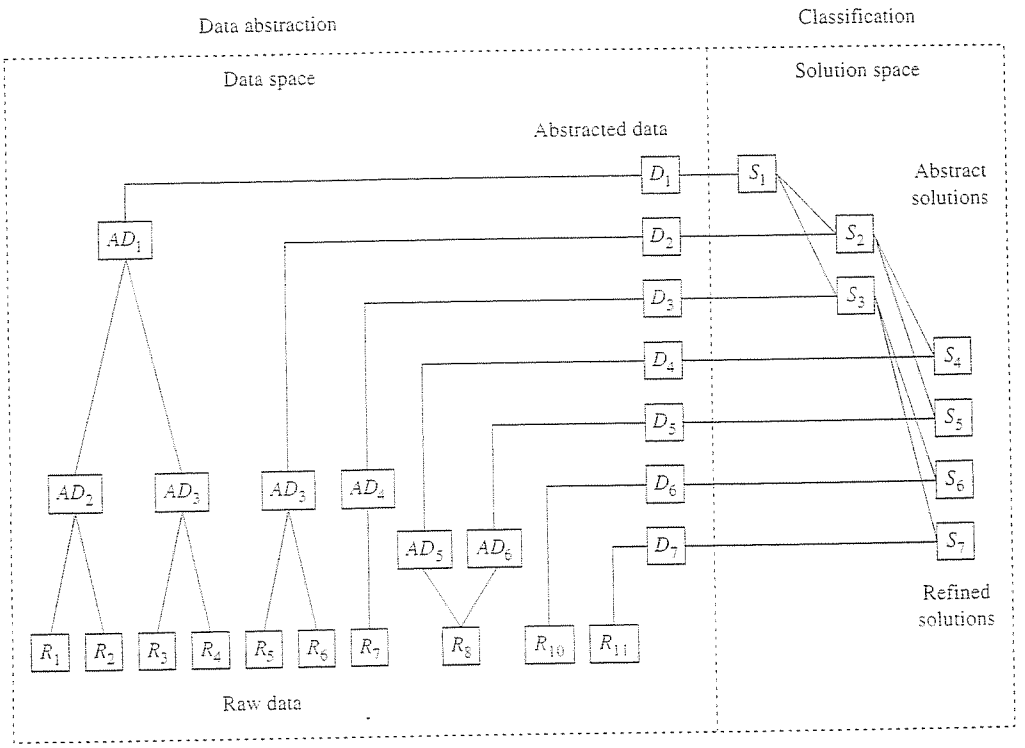


FIGURE 7.10. Adding data abstraction to the conjunctive classification model.

Although we have distinguished data-space hierarchies from solution-space hierarchies, we note that the distinction is sometimes in the mind of the beholder. Another way to draw a front-end classification model is in terms of a multistage classification diagram in which the raw and abstract data form the first stage and the abstract and refined solutions form the second stage. In this redrawing, the topology of the links remains the same but the meanings of the links differ from our prototypical case.

7.2.4 *Summary and Review*

In this section we introduced a simple model for classification, conjunctive classification. This initial model allows binary values for data and simple conjunctive relations among data to match data with solution classes. Using examples based on this simple model, we illustrated cases where data matched no solutions, one solution, or many solutions.

We extended the model to admit reasoning by elimination and mutual exclusion. Reasoning by elimination uses negative evidence against one class to support another class. We considered variations on this idea in hierarchical organizations, which highlighted cases where entire branches of a taxonomy could be pruned at once. Mutual exclusion uses positive evidence for one class as negative evidence for another class.