

**Figure 4.1** Key to diagrams of linked data structures.

varying the linked list’s length dynamically. We can change the order of the nodes. In fact, we can make arbitrary changes to a linked list’s structure simply by manipulating the links. (Such effects are simply not possible with arrays.)

In linked lists we have a choice between linking each node to one or both of its immediate neighbors. Thus we have two different data structures to study in this chapter:

- singly-linked lists (SLLs)
- doubly-linked lists (DLLs).

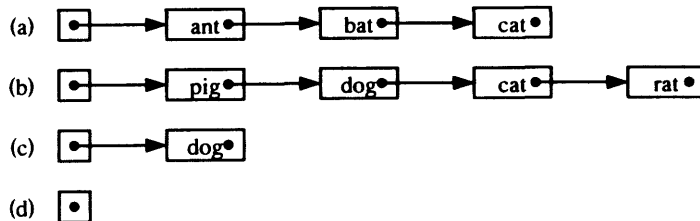
### 4.1.1 Singly-linked lists

A *singly-linked list* (or *SLL*) consists of a sequence of nodes with the following properties:

- Each SLL node contains an element, together with a link to its successor (or a null link if the node has no successor).
- The SLL has a *header*, which contains a link to the SLL’s first node (or a null link if the SLL is empty).

Figure 4.2 shows various SLLs. The SLL (a) consists of three nodes, which contain the words ‘ant’, ‘bat’, and ‘cat’, respectively; this SLL’s header contains a link to the node containing ‘ant’, which in turn is linked to the node containing ‘bat’, which in turn is linked to the node containing ‘cat’, which being the last node contains a null link. The SLLs (b) and (c) have different numbers of nodes and different elements. The SLL (d) is empty, so its header contains a null link.

In Java, we can represent SLL headers by objects of the class `SLL`, shown in Program 4.3, and SLL nodes by objects of the class `SLLNode`, shown in Program 4.4. Each



**Figure 4.2** Singly-linked lists.