Chapter

# 13 Scope and Resolution of Elements

Charts are the building blocks of a model. These blocks are not isolated entities, but are linked together by information that might flow between them and by elements they might share. In particular, some of the elements defined in one chart can be used in others. Clearly, however, in large projects there are many elements that need not be known outside a limited portion of the specification.

Hence issues of scope—dealing with the questions of where elements are defined, where they are recognized, and where they may be used—are important. This chapter discusses these issues, and the way we deal with them is strongly related to the hierarchy of charts, which was discussed in Chap. 12.

This chapter also introduces a new component of our languages, the *global definition set* (GDS), which contains information that is visible throughout the entire model.

## 13.1 Visibility of Elements and Information Hiding

Decomposing specifications into many charts raises issues of visibility and the scoping of elements. Consider Fig. 13.1. The activity MAIN has two subactivities, A and B, between which X flows, and each is described in a separate chart. Obviously, we want X to be recognized in both charts because it is part of their external interface. The X in both charts is thus the same X. On the other hand, we would like the two Ys appearing in these charts to be different when each is internal to the chart in which it appears. These two charts may actually have been prepared by different teams. In fact, the two Ys could be of very different types, say, a data-item in A and an event in B. Thus X represents the case of an element that has to be *visible* to several charts, and the Ys represent cases of elements that are to be *hidden* inside specific charts.

These notions of *visibility* and *information hiding* are important in any kind of structured development. Some elements are allowed to be known only in specific parts of the model, and others might be *global,* that is, known throughout it. Often, it is important to give subteams the freedom to name their elements as they wish, regardless of the possible existence of identical names elsewhere in the model, and to produce reports and carry out analysis on particular portions thereof. To accommodate these possibilities, we associate a *scope* with each element. The scope of an element is a set of charts in which the element is known and can be used. As in modern programming languages, we have a notion of where the element is *defined* and a set of scoping rules that determine where it is visible.

## 13.2 Defining, Referencing, and Resolving Elements

Each element in the model belongs to a specific chart. We say that it is *defined in* that chart. Graphical elements (boxes, arrows, and connectors) are defined in the chart in which they are drawn, besides the special case of external boxes. Textual elements (information elements and actions) are defined in the chart that is specified in the element's Data Dictionary entry. See Sec. 13.4.

Elements defined in one chart may be used in others. For example, we may define the data-item X of Fig. 13.1 in the higher-level chart MAIN by writing MAIN in the field Defined in Chart of its Data Dictionary entry, as shown in Fig. 13.2. Because X is used along a flow-

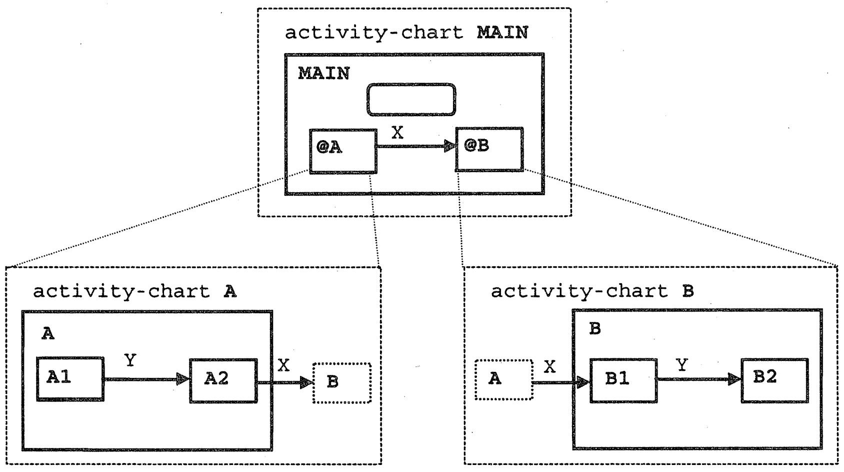


**Figure 13.1** Visibility vs. information hiding.

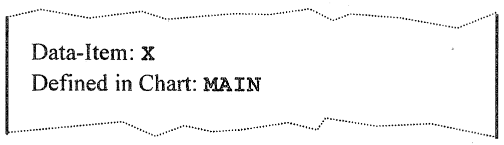

Data-Item: **X**
Defined in Chart: **MAIN**

Figure 13.2 An element defined in a chart.

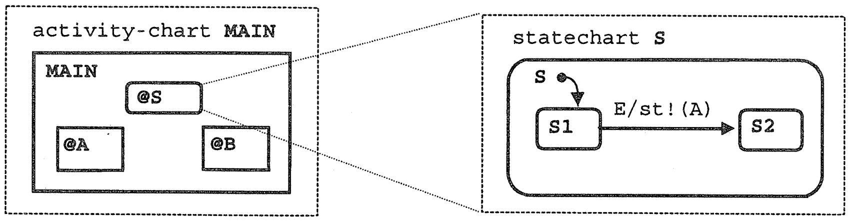


Figure 13.3 A reference activity.

line between the subactivities A and B, it also appears along flow-lines in the charts for A and B. In these two charts, in which X is used but not defined, we say that X is a *reference element.*

Another example of a reference element appears in Fig. 13.3. Here, the activity A is defined in the activity-chart MAIN, by virtue of its being drawn there. On the other hand, because it is started in the statechart S, which describes the control activity of MAIN, activity A is a reference element in S, where it is used but not defined.

Each reference element must be matched with, or *resolved to,* an element in some other chart. The latter is said to be the *resolution* of the former. In the aforementioned examples, the reference data-items X of Fig. 13.1 in both charts A and B are resolved to the data-item X defined in the chart MAIN, and, similarly, the reference activity A of Fig. 13.3 in the statechart S is resolved to the activity A in MAIN.

Often, it is useful to be able to refer to elements that have not yet been defined. In the terminology just introduced, this amounts to having a reference element that cannot be resolved to any element. Such a situation might occur in intermediate stages of the specification process. A simple example is the use of an external event as a trigger in a statechart before the activity-chart that defines that event is constructed. Another example appears in Fig. 13.4, which is similar to Fig. 13.3. The difference is that here the activity K, which is started in statechart S, has not yet been defined in MAIN. This could have been intentional (K is not ready yet), or it could indicate an error. Thus K is an *unresolved reference element* in chart S.

The specific rules for visibility and resolution differ for different types of elements. They are discussed in detail for graphical elements in Sec. 13.3 and for textual elements in Sec. 13.4.

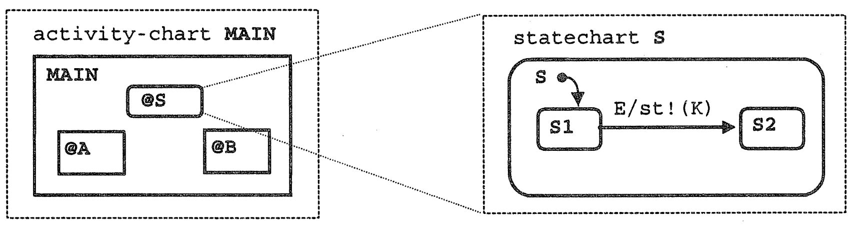


Figure 13.4 An unresolved reference activity.

Having scopes associated with elements makes it possible to use the same name for different elements, and there are rules that determine when this is allowed. Elements with the same name can be distinguished by attaching the chart name (i.e., the one in which they are defined) to their own name. The format is `chart-name:element-name`. However, this use is not always allowed, and the rules for referring to elements in this way are related to the scoping rules. Such practice is useful for testbenches (see Chap. 12), where the scoping rules do not hold, and any element of the model can be referred to freely.

The rules for uniqueness of names and for referencing are discussed in the following sections.

## 13.3 The Scope of Charts and Graphical Elements

Charts involve several kinds of graphical elements—boxes, arrows, and connectors. These elements, with the exception of external boxes (which are discussed later), are defined in the chart in which they are drawn. Arrows have no names and cannot be referred to in other charts. Also, the only connectors that have names are diagram connectors, and their naming and reference rules were discussed in Chap. 11. This leaves us with having to discuss the scoping and reference rules for charts and boxes only.

### 13.3.1 Referring to charts and box elements

Charts are global in the entire model. Their names are unique, even for different types of charts, and they are recognized everywhere. So far, we saw that charts are referred to in other charts in two ways: in the names of boxes (to point to offpage charts) and in the Data Dictionary (to specify that a module is described by an activity-chart). We shall see later that generic charts are referred to in a similar way. As for other elements of the model, references to charts are resolved to charts of appropriate type. If such charts do not exist yet in the model, we say, as for other element types, that the reference charts are *unresolved*.

The box elements of our languages are activities and data-stores, which are defined in activity-charts, states, which are defined in statecharts, and modules, which are defined in module-charts. As we have seen in earlier chapters, the box elements are named in the graphics, and the name of the box must be unique among its siblings boxes. When the name is not unique in the chart, the box can be referred to by its pathname, preceded by its ancestor(s), (e.g., `A.B.C`; see App. A.1). If there is a synonym for the box, defined in its Data Dictionary entity, then that synonym must be unique among the names and synonyms of the boxes defined in the same chart. A box can be referred to by its name or its synonym.

We now describe the rules for referencing a box element in a chart other than the one in which it is defined. Any cases that are not discussed, such as referencing a state in an activity-chart, are not allowed.

### 13.3.2 Referring to activities in statecharts

Activities can be referred to in statecharts in actions (e.g., `st!(A)`), in events (e.g., `sp(A)`), and in conditions (e.g., `ac(A)`). These actions, events, and conditions may appear as parts of labels along transitions, as parts of static reactions, and in the definitions of other elements in the Data Dictionary. In addition, activities may be referred to in a state's Data Dictionary entity in the field `Activities in State`. See Chap. 7.

As discussed in Chap. 7, in a (logical) statechart reference is allowed only to activities that are siblings of the control activity described by the statechart. This is the only way to refer to activities in a statechart. As an example, in Fig. 13.5, the activity `A` in the chart `MAIN` is referred to in the statechart `S2`, which belongs to the logical statechart `S` that controls the activity `MAIN`.

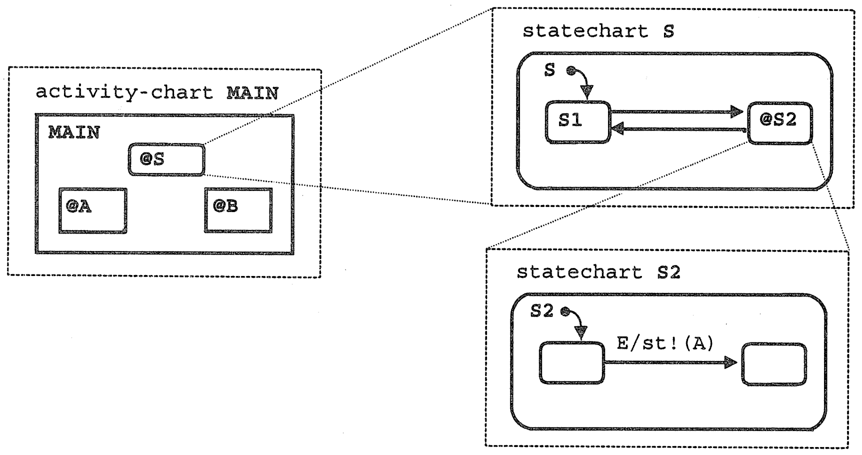


Figure 13.5 Referring to an activity in a statechart.

Notice that the activities that can be referred to in a given statechart SC must belong to a very particular activity-chart, namely, the parent activity-chart of SC, that is, the one containing the control activity described by SC. Therefore, there is no need to attach a chart name to the name of a referenced activity, and, indeed, such an attachment is not allowed.

If a statechart refers to an activity name that does not appear in the parent activity-chart, as in Fig. 13.4, that reference remains unresolved. This is true even if there is an activity with the same name elsewhere in the model.

### 13.3.3 Referring to states in statecharts

States can be referred to in statecharts in events (e.g., en(S)) and conditions (e.g., in(S)). These events and conditions can be used along transitions, as part of static reactions, and in Data Dictionary definitions of other elements. See Chap. 5.

The visibility rule is that a state can be referred to in any state that belongs to the same logical statechart. In other words, any state that is defined in a page that is a descendant of some statechart SC is visible to all charts that are descendants of SC. States defined in other charts that are not part of the logical statechart of SC are not visible.

States in the same page are referred to by name or pathname (if the name is nonunique in the page), while states in other pages are preceded by the appropriate chart name (i.e., chart-name:state-name). As an example, consider Fig. 13.6. In the statechart S2, the state OFF that appears in the label E[in(OFF)] is understood to be the state OFF in the orthogonal component S22, which appears in the same statechart, although there is a state named OFF in the chart S1, too. On the other hand, to refer to S2:ON in the label in chart S1, the state name is preceded by the chart name S2, and although the name Q is unique in the entire logical chart, the chart name is also added to it when it is used in another chart.

A reference to a state name that does not appear in any chart of the same logical statechart remains unresolved.

### 13.3.4 External activities or modules

External activities and modules are considered to be "real" elements, (e.g., they have their own entities in the Data Dictionary) only when they are defined explicitly in the Data Dictionary as environment activities or modules. An unnamed external box is just a graphical object, like a connector, that signifies some anonymous external source or target. A named external box that is not defined as an environment box serves as a reference to another box. Like other reference elements, an attempt is made to resolve such a

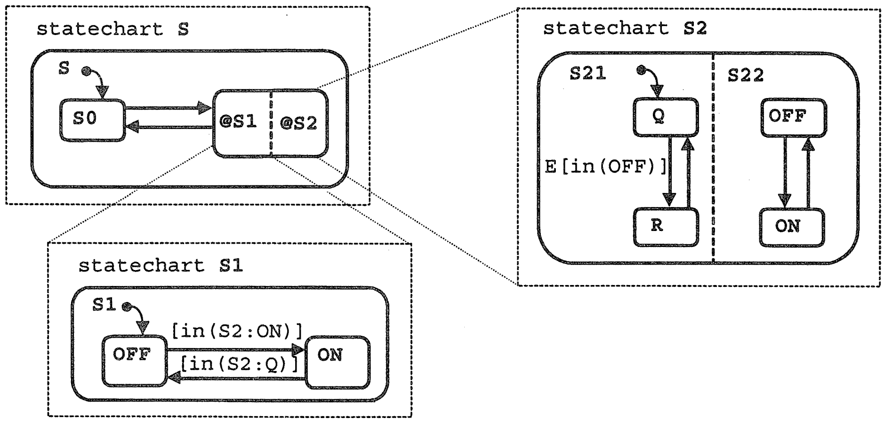


**Figure 13.6** Referring to states in a statechart.

box to a matching element—in this case, a box from the parent chart. The matching box in the parent chart has the same name, and it can be an internal box (i.e., a module, activity, or data-store) or an external box.

Consider the example in Fig. 13.7. Activity-chart `M1_AC` contains a number of external activities: `E1` is resolved to the environment module `E1`, and `M2` is resolved to the internal module `M2`, but `M31` does not match any module in `M` (although `M` contains a module named `M31`). This is because the matching boxes are allowed to be found only among the siblings of `M1` (e.g., `M3`) or the siblings of `M1`'s ancestors (e.g., `E1`). Similarly, in activity-chart `A`, the external activity `D` is resolved to the data-store `D` in `M1_AC`, and `E1` is resolved to `E1` in `M` "via" `E1` in `M1_AC`.

A named external box in a root chart (i.e., one with no parent), or a box to which no box in the parent can be matched, is considered to be an unresolved external box. For example, if the external module `E2` in the root chart `M` is not explicitly defined as an environment module, it is considered unresolved. Also, `K` in the activity-chart `A` and `M31` in `M1_AC` are unresolved external boxes, because no matching boxes for them are found.

### 13.3.5 Referring to modules and activities in activity-charts

Modules can be referred to in an activity-chart in the field `Implemented by` (respectively, `Resides in`) of the Data Dictionary entity of an activity (respectively, a data-store). See Fig. 10.6. Any module, from any module-chart, can be referred to in these fields. Recall, however, that the rules of Chap. 10 concerning the consistency of the hierarchies of modules and activities must be followed.

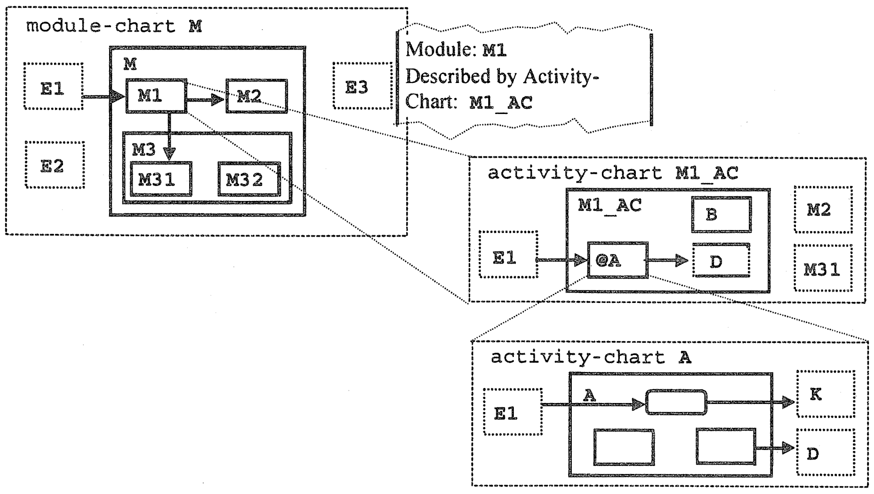


Figure 13.7 Resolution of external boxes.

Because module names are not unique, our languages allow module names to be referred to from different charts. In cases of possible ambiguity, the chart name should be attached to the module name.

Activity names and data-store names are entered in the related fields `Is Activity` and `Is Data-Store`, respectively. As explained in Sec. 10.4, an element name in these fields is meaningful only when the implementing module is specified. The activity or data-store entered must be from the activity-chart that describes the implementing module. See Fig. 13.8. Consequently, there is no need to specify the chart name of the referred to element, and, indeed, attaching this name is not allowed.

## 13.4 The Scope of Textual Elements

Textual elements (i.e., events, conditions, data-items, user-defined types, information-flows, and actions) are defined via the Data Dictionary. The chart in which the element is defined is specified by the modeler in the field `Defined in Chart`. See Fig. 13.2. This should be contrasted with graphical elements, for which the definition charts are determined by where they are drawn.

### 13.4.1 Visibility of textual elements

A textual element that is defined in a particular chart is recognized in and can be used in other charts. The visibility rules for textual elements are very similar to those employed in programming languages that support nesting and block structure. A textual element is clearly visible in the chart in which it is defined. It is also visible in all the descendant charts in the chart hierarchy defined in Chap. 12. An exception is when the element is *masked by* another textual element with the same name, as discussed shortly.

Let us take an example. The event OUT_OF_RANGE, defined in the activity-chart EWS_ACTIVITIES, is used in the statechart EWS_CONTROL on a transition; see, for example, Fig. 4.3. To use our terminology, the reference to OUT_OF_RANGE in EWS_CONTROL is resolved to the event OUT_OF_RANGE that is defined in EWS_ACTIVITIES. Because the statechart EWS_CONTROL is a subchart of the activity-chart EWS_ACTIVITIES (see Fig. 12.2), the textual elements defined in the latter are visible in the former and can therefore be used therein.

Figure 13.9 illustrates masking. The data-item X flows between activities A and B in the activity-chart MAIN. Assume that it is also defined there. The offpage chart C, which defines an internal activity of MAIN, uses an element with the same name, X (in the example, X is actually an event in C). According to the visibility rule, the data-item X of MAIN could have been used in the subchart C, but because an event X is defined in C, the data-item X is no longer recognized there. Moreover, the same applies to C's subchart C1, in which we may refer only to the event X of C and not to the data-item X of MAIN. In such a case, we say that the data-item MAIN:X is *masked by* the event C:X.

### 13.4.2 Naming textual elements

The name and synonym of a textual element are given in its Data Dictionary entity. Within a chart, all such names and synonyms must be unique. Hence, if an event named E has already been defined in

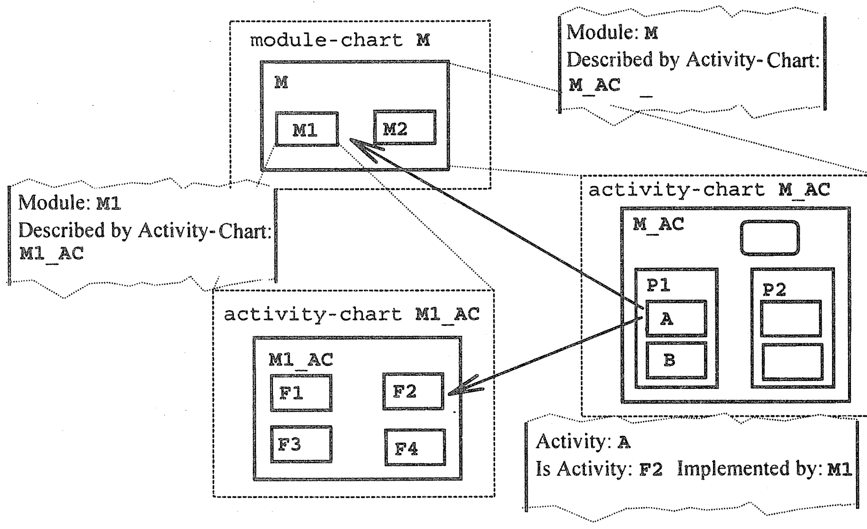


Figure 13.8 Referring to activities in Is Activity field.

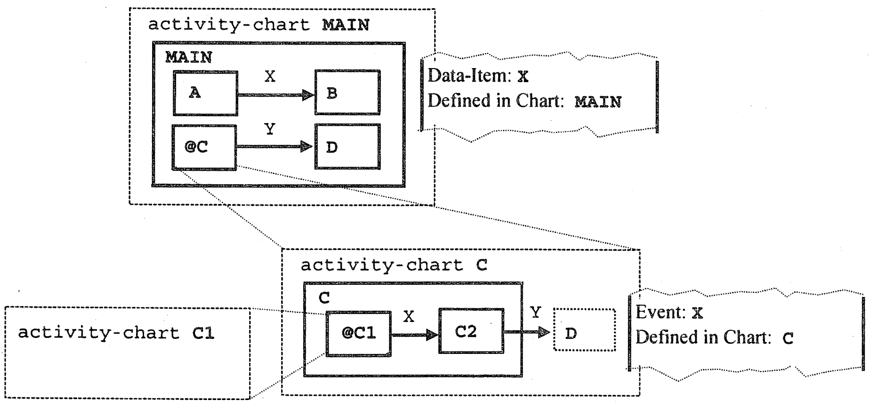


**Figure 13.9** Masking a textual element.

some chart A, the name E cannot be used to define, say, a condition in the same chart. It may be used, however, in some other (physical) chart, to name an event, a condition, or any other textual element. As for naming graphical elements, the name E can be used anywhere, even in the chart A itself.

The possibility of using the same name for different elements is convenient and useful, especially in big projects when different teams use the same names in different scopes. However, despite the presence of rules for resolving references and detecting masking situations, this option should be exercised with care, as it may cause confusion.

Special attention must be paid when the same element is used in several charts to ensure that the different occurrences are resolved to the same element. Because a textual element is visible only in the descendants of its defining chart, an element should be defined in a chart that is high enough in the hierarchy to be the common ancestor of all charts within which the element is to be referenced. For example, consider the event E in Fig. 13.10, which is generated in state S1. If we want E to cause a transition in state S2, it must be defined in the statechart S or in one of its ancestor charts, even if it is not used there, because elements that are defined in S1 are not visible in S2, and vice versa. When E is defined in S, both references to it in S1 and S2 are resolved to the definition in S, and the two are therefore understood to refer to the same element. This example should be contrasted with the case illustrated in Fig. 13.1, where we used the same name Y in the two charts A and B for two different elements flowing between subactivities. Because we want these elements to be distinct, we should define them in separate entities in the Data Dictionary in each of the two charts.

A textual element can be referred to in the same chart or in some other chart by its name or synonym. We do not allow the format chart-name:element-name for textual elements because the chart name would be either redundant (if the chart is the one containing the resolution) or illegal (if another chart is referenced, thus referring to an "invisible" element or one that is out of scope). For example, in Fig. 13.10 we are not allowed to replace the event E in the statechart S2 by S1:E because, according to the visibility rules, elements defined in S1 are not visible in S2. Note that this rule does not apply to testbenches, where all elements of the model are visible.

### 13.4.3 More about resolution of textual elements

Reference elements are always resolved to elements of the same type. Thus if we were to define a condition named E, not an event, in the statechart S of Fig. 13.10, the two references to E in S1 and S2 would not be resolved to this condition because they are used as events.

If a textual element is referred to without having been defined explicitly in the Data Dictionary and there is no corresponding element in the ancestor charts, the element remains unresolved. Typically, this happens in intermediate stages of the specification. Sometimes the type of an unresolved element is not clear from its usage. A good example is when an element appears as a label on a flow-line, in which case it can be an event, a condition, a data-item, or an information-flow. However, elements appearing in transition labels, for example, have uniquely determined types, as do those appearing in expressions that define other textual elements. (An exception is the case of an action that can possibly turn out to be an event, such as E in Fig. 13.10.)

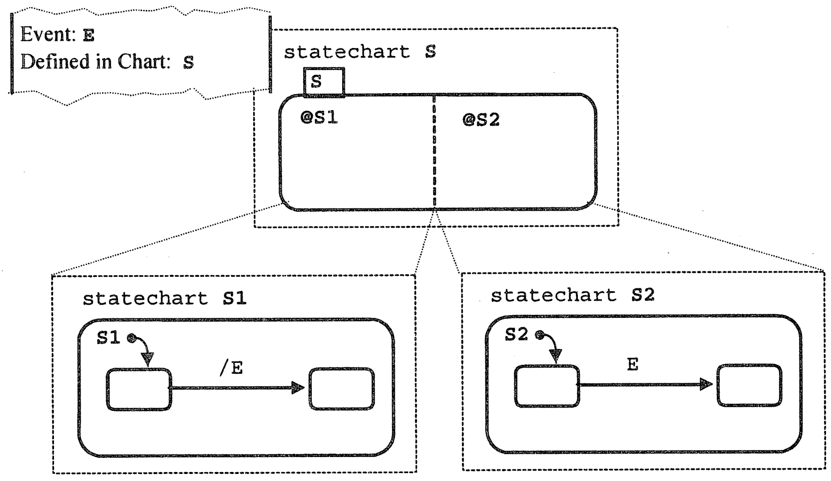


Figure 13.10 Connecting elements from different charts.

Being unresolved does not prevent elements from being visible, and hence from being used, in descendant charts. Thus elements from such descendant charts can be resolved to unresolved elements. For example, assume that in Fig. 13.1 we do not explicitly define the element X. It will nevertheless be considered a reference element in the three charts appearing in the figure. It will be unresolved in chart MAIN, but in the other two charts it will be resolved to an unresolved element X in MAIN. As in other cases, however, this kind of resolution will be carried out only if the types of the elements match. For example, in Fig. 13.11, E is not explicitly defined in chart MAIN, and therefore it is an unresolved reference element. Judging only from its usage in MAIN, it may be an information-flow, a condition, a data-item, or an event, but in this case it is considered to be an event because it is used as an event in the subchart S.

## 13.5 Global Definition Sets

The visibility rules imply that textual elements that have to be global to the entire model should be defined in the root of the chart hierarchy, which is the common ancestor of all charts in the model. The resolution scheme described earlier, which is based on the hierarchy of the functional components, is compatible with the functional decomposition method. In this method, every accessed data variable—event, condition, and data-item—is either local (i.e., it belongs to the functional component) or is part of the external interface (i.e., it appears on a flow-line and, as such, belongs to an ancestor functional component). Textual elements that are employed as abbreviations (i.e., information-flows and actions) are usually defined in the charts in which they are used. Therefore, the only "real" global information that has to be

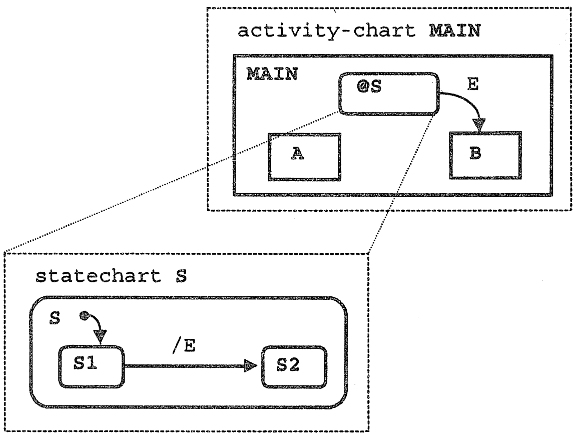
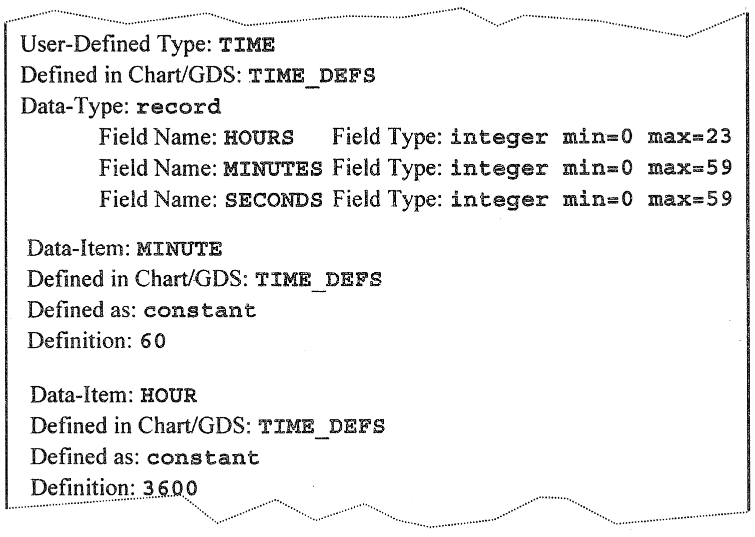


Figure 13.11 Compatible usage of textual elements.



User-Defined Type: `TIME`
Defined in Chart/GDS: `TIME_DEFS`
Data-Type: `record`

```
Field Name: HOURS    Field Type: integer min=0 max=23
Field Name: MINUTES  Field Type: integer min=0 max=59
Field Name: SECONDS  Field Type: integer min=0 max=59
```

Data-Item: `MINUTE`
Defined in Chart/GDS: `TIME_DEFS`
Defined as: `constant`
Definition: `60`

Data-Item: `HOUR`
Defined in Chart/GDS: `TIME_DEFS`
Defined as: `constant`
Definition: `3600`

**Figure 13.12** Elements defined in a global definition set.

shared throughout the entire model in an unstructured manner (and can even be moved between models) is that of constant definitions and user-defined types.

Our languages provide a special type of model component, the *global definition set* (GDS), for capturing such global definitions. This type of component is part of the Data Dictionary, and it is similar in many ways to a chart. There may be several GDSs in a model, each containing definitions of user-defined types as well as constant data-items and conditions. Figure 13.12 shows several Data Dictionary entities that belong to a GDS named `TIME_DEFS`. A GDS that contains definitions related to time, as in this example, is relevant to many application domains.

As mentioned, elements appearing in a GDS are visible in the entire model. For example, a data-item definition in any chart of a model that contains the GDS `TIME_DEFS` can be of type `TIME`. In particular, definitions in one GDS can use definitions in another GDS, but this should not be done in a circular fashion.

There are no hierarchical relationships among the GDSs in a model or between them and the charts of the model itself.

Global definition sets have a special role in the context of generic charts, as will be seen in Chap. 14.