occurrence indicators and themselves combined by group connectors. To demonstrate these facilities, let us now expand our example to include non-stanzaic types of verse. For the sake of demonstration, we will categorize poems as one of *stanzaic*, *couplets*, or *blank* (or *stichic*). A blank-verse poem consists simply of lines (we ignore the possibility of verse paragraphs for the moment)

> It will not have escaped the astute reader that the fact that verse paragraphs need not start on a line boundary seriously complicates the issue; see further section 2.5.2, *Concurrent Structures*.

so no additional elements need be defined for it. A couplet is defined as a line1 followed by a line2.

```
<!ELEMENT couplet O O (line1, line2) >
```

The elements line1 and line2 (which are distinguished to enable studies of rhyme scheme, for example) have exactly the same content model as the existing line element. They can therefore share the same declaration. In this situation, it is convenient to supply a *name group* as the first component of a single element declaration, rather than give a series of declarations differing only in the names used. A name group is a list of GIs connected by any group connector and enclosed in parentheses, as follows:

```
<!ELEMENT (line | line1 | line2) O O (#PCDATA) >
```

The declaration for the poem element can now be changed to include all three possibilities:

```
<!ELEMENT poem - O (title?, (stanza+ | couplet+ | line+) ) >
```

That is, a poem consists of an optional title, followed by one or several stanzas, or one or several couplets, or one or several lines. Note the difference between this definition and the following:

```
<!ELEMENT poem - O (title?, (stanza | couplet | line)+ ) >
```

The second version, by applying the occurrence indicator to the group rather than to each element within it, would allow for a single poem to contain a mixture of stanzas, couplets or blank verse.

Quite complex models can easily be built up in this way, to match the structural complexity of many types of text. As a further example, consider the case of stanzaic verse in which a refrain or chorus appears. A refrain may be composed of repetitions of the line element, or it may simply be text, not divided into verse lines. A refrain can appear at the start of a poem only, or as an optional addition following each stanza. This could be expressed by a content model such as the following:

```
 <!ELEMENT refrain - - (#PCDATA | line+)>
<!ELEMENT poem    - O (title?,
                       ( (line+)
                       | (refrain?, (stanza, refrain?)+ ) )) >
```

That is, a poem consists of an optional title, followed by either a sequence of lines, or an un-named group, which starts with an optional refrain, followed by one of more occurrences of another group, each member of which is composed of a stanza followed by an optional refrain. A sequence such as ‘refrain - stanza - stanza - refrain’ follows this pattern, as does the sequence ‘stanza - refrain - stanza - refrain’. The sequence ‘refrain - refrain - stanza - stanza’ does not, however, and neither does the sequence “stanza - refrain - refrain - stanza.” Among other conditions made explicit by this content model are the requirements that at least one stanza must appear in a poem, if it is not composed simply of lines, and that if there is both a title and a stanza they must appear in that order.

2.5 Complicating the Issue: More on Element Declarations

In the simple cases described so far, it has been assumed that one can identify the immediate constituents of every element defined in a textual structure. A poem consists of stanzas, and an anthology consists of poems. Stanzas do not float around unattached to poems or combined into some other unrelated element; a poem cannot contain an anthology. All the elements of a given document type may be arranged into a hierarchic structure, arranged like a family tree with a single ancestor at the top and many children (mostly the elements containing #PCDATA) at the bottom. This gross simplification turns out to be surprisingly effective for a large number of purposes. It is not however adequate for the full complexity of real textual structures. In particular, it does not cater for the case of more or less freely floating elements that can appear at almost any hierarchic level in the structure, and it does not cater for the case where different elements overlap or several different trees may be identified in the same document. To deal with the first case, SGML provides the *exception* mechanism; to deal with the second, SGML permits the definition of ‘concurrent’ document structures.

2.5.1 Exceptions to the Content Model

In most documents, there will be some elements that can occur at any level of its structure. Annotations, for example, might be attached to the whole of a poem, to a stanza, to a line of a stanza or to a single