Meeting Minutes 14 March 2001

Grouping Revisited

Authors:
Melanie Nelson (Physiome Sciences Inc.)
Warren Hedley (Bioengineering Institute, University of Auckland)
Contributors:
Poul Nielsen (Bioengineering Institute, University of Auckland)
Yi Ge (Physiome Sciences Inc.)
Kam Jim (Physiome Sciences Inc.)

1 Introduction

As Warren and Melanie closed in on a draft of the specification for CellML version 1.0, Physiome's software developers actually began implementing code that can write this version of CellML. Yi Ge was in charge of making Physiome's Cell Editor software write CellML. After reading the 2001-03-02 draft version of the CellML specification, he had two suggestions about how we could make CellML easier for him to implement, and more intuitive in general. Now that the draft specification has been made public, the CellML development team can turn its attention to these suggestions.

This progress report presents Yi's comments on the difficulty of defining hierarchies using the current grouping scheme, and possible new schemes that address these comments.

2 Summary of the Current Scheme

In the grouping system described in the 2001-03-02 draft of the CellML specification, software is expected to build up a hierarchy out of numerous **<group>** elements, each of which describe the relationship between a single parent and its children. This approach was adopted so that multiple relationships could be defined for a single group; re-use targeted each level in the hierarchy, rather than the entire hierarchy. An example of the old scheme of hierarchy definition is given in Figure 1.

Using the current grouping syntax, three **<group>** elements are required to define the two level hierarchy in the example. This is because the children of the two intermediate nodes (**sodium_current** and **calcium_current**) must be defined in separate **<group>** elements. However, the scheme does have the advantage of allowing one of the nodes (**cell_membrane**) to be shared by the geometric containment and logical encapsulation hierarchies.

3 Yi's Comments

Yi voiced the opinion that the assembly of a hierarchy from its individual levels was a costly and difficult exercise for implementors, and was complex, verbose, and counter-intuitive for document authors. Even if hierarchies defined in a model only partially overlapped, it would still be significantly less verbose to define each hierarchy in its entirety separately, than to define each level separately and associate relationships with the parts in common. Yi recognised that eventually multiple groups would need to be merged (for instance, if two subnetworks are to be merged, their grouping hierarchies will also need to be merged). However, he thought it would be easier to do this from a few nested group definitions that defined large parts of the total hierarchy, rather than from a large number of definitions that each defined one node in the hierarchy tree.

An additional problem with the current scheme is that the **role** attribute on the **<component_ref>** elements is not a very intuitive way to define a hierarchy.

```
<group>
  <relationship_ref relationship="is_contained_in" />
  <component_ref component="cell" role="major" />
  <component_ref component="cell_membrane" role="minor" />
</group>

<group>
  <relationship_ref relationship="is_encapsulated_by" />
  <relationship_ref relationship="is_contained_in" />
  <component_ref component="cell_membrane" role="major" />
  <component_ref component="sodium_current" role="minor" />
  <component_ref component="calcium_current" role="minor" />
</group>

<group>
  <relationship_ref relationship="is_encapsulated_by" />
  <component_ref component="sodium_current" role="major" />
  <component_ref component="sodium_channel_1" role="minor" />
  <component_ref component="sodium_channel_2" role="minor" />
</group>

<group>
  <relationship_ref relationship="is_encapsulated_by" />
  <component_ref component="calcium_current" role="major" />
  <component_ref component="L_type_calcium_channel" role="minor" />
  <component_ref component="T_type_calcium_channel" role="minor" />
</group>
```

FIGURE 1: The hierarchy definition scheme proposed in the 2001-03-02 draft of the CellML specification. This example defines a logical encapsulation hierarchy and a geometric containment hierarchy. Note that one of the **<group>** elements participates in both hierarchies.

4 A New Method

Using XML nesting to represent a hierarchy is more intuitive and less verbose than our current syntax. We cannot just nest the components themselves, since we must be able to support multiple hierarchies over a single network of components. However, we can nest the **<component_ref>** elements in the **<group>** element. This has the dual advantage of getting rid of the non-intuitive **role** attribute as well as making it possible to define several levels of a hierarchy within a single **<group>** element.

Figure 2 shows the same example used in Figure 1, defined using XML nesting to specify the groups.

```
<group>
  <relationship_ref relationship="containment" />
  <component_ref component="cell">
    <component_ref component="cell_membrane">
      <component_ref component="sodium_current" />
      <component_ref component="calcium_current" />
    </component_ref>
  </component_ref>
</group>

<group>
  <relationship_ref relationship="encapsulation" />
  <component_ref component="cell_membrane">
    <component_ref component="sodium_current">
      <component_ref component="sodium_channel_1" />
      <component_ref component="sodium_channel_2" />
    </component_ref>
    <component_ref component="calcium_current">
      <component_ref component="L_type_calcium_channel" />
      <component_ref component="T_type_calcium_channel" />
    </component_ref>
  </component_ref>
</group>
```

FIGURE 2: A group definition scheme using XML nesting. Note that the geometric containment group and the logical encapsulation group must be defined separately, since they only partially overlap.

The CellML-defined relationship types are now identified by the strings "encapsulation" and "containment". These were formerly identified by "is_encapsulated_by" and "is_contained_in", which were confusing, particularly when combined with the **role** attributes on the **<component_ref>** elements

In the event that a hierarchy consists of several discontinuous segments, multiple **<component_ref>** elements can be placed inside a single **<group>** element. The specification would again advise that to assemble these into a single hierarchy, an imaginary root component could act as a parent to the referenced components. An example of this possibility is shown in Figure 3.

Warren considered renaming the **<group>** element the **<hierarchy>** element. This is appropriate for the CellML-defined relationship types, but may be inappropriate for user-defined relationship types such as the "is_next_to" relationship type shown in examples in the 2001-03-02 draft specification. Grouping is a more general type of relationship.

```
<group>
  <relationship_ref relationship="encapsulation" />
  <component_ref component="cell_membrane">
    <component_ref component="sodium_current">
      <component_ref component="sodium_channel_1" />
      <component_ref component="sodium_channel_2" />
    </component_ref>
    <component_ref component="calcium_current">
      <component_ref component="L_type_calcium_channel" />
      <component_ref component="T_type_calcium_channel" />
    </component_ref>
  </component_ref>
  <component_ref component="cytosolic_calcium_buffers">
    <component_ref component="calmodulin" />
    <component_ref component="troponin" />
    <component_ref component="parvalbumin" />
  </component_ref>
</group>
```

FIGURE 3: A logical encapsulation hierarchy in which two components are at the top level. The **cell_membrane** and **cytosolic_calcium_buffer** components do not share an actual parent component in the logical encapsulation hierarchy, but can be considered to both be children of the imaginary root component that is assumed to be the parent of all unencapsulated components in the model. Note that there may be other components that are siblings of the **cell_membrane** and **cytosolic_calcium_buffer** components. If a component is not encapsulated and does not encapsulate other components, it will not appear in a **<group>** element.

5 Remaining Issues to be Resolved

There are still two issues with grouping that need to be resolved. These are discussed in the following subsections.

5.1 Allowing Multiple Group Elements

It is not immediately clear if there is any advantage to allowing hierarchies to be split up over multiple **<group>** elements as they are now. Keeping all of the nodes in a hierarchy in a single **<group>** element would make it easier for software to construct the grouping hierarchies when it reads in a CellML model. However, advanced software would still need to be able to merge hierarchies, since it would need to be able to pull in subnetworks stored in a database and integrate them. The subnetworks would typically be stored with their grouping information. Therefore, integrating subnetworks would require integrating grouping hierarchies.

Tentative recommendation: Allow multiple **<group>** elements for a given hierarchy, but recommend that a single **<group>** element be used per hierarchy whenever possible. Consider making the processor behaviour rules that specify how software is to assemble multiple group elements into a single hierarchy conformance level 2 rules.

5.2 Allowing Multiple Relationships within a Group

It is also not immediately clear whether or not a single **<group>** element should still be allowed to contain multiple **<relationship_ref>** elements. This would only be possible when the hierarchies corresponding to the referenced relationships are identical. In these cases, allowing multiple **<relationship_ref>** elements in a single **<group>** element could significantly reduce the amount of XML needed to define the groups. However, in cases where the hierarchies only partially overlap, it could complicate a model's definition.

Tentative recommendation: Allow multiple **<relationship_ref>** elements in a **<group>** element. All nodes defined in the **<group>** element must belong in all of the hierarchies corresponding to the referenced relationships.