Meeting Minutes 29 July 2003

Autumn A Cuellar

Table of Contents

Introduction

Am beginning to lose hope that we'll ever agree on a solution to this import problem...

sibling_connections Attribute

While at a conference on Waiheke, Andre, Matt, and I were discussing an example in which you are linking one metabolic pathway to another where you basically only need the output concentration of one species in one pathway to be the input concentration in a second pathway. Typically the pathway models that Catherine has described in CellML have been structurally flat, no encapsulation whatsoever. So in order to code up the example using the 1.1 described in the past two meeting minutes, you either import each component in the first pathway separately and re-connect them up, or you manually add an encapsulating component to the first pathway model through which the specie's concentration is passed, or we design some algorithm for software to be able to handle such situations.

So we went back to the attribute on the **`<import>`** element idea: an optional **`sibling_connections`** attribute, which, if given a value of `"true"`, tells a simulator that the value of the variable being imported relies on the owning component's siblings and the connections between them. Retaining sibling connections would be possible only for top-level components. Clean and simple solution, but only accounts for the use-case that we were working with.

So obviously Poul and David found problems with this solution immediately, namely that if you wanted another specie's concentration, you would need to create another instance of (in the use-case we presented) the entire model. Poul's, David's, and Matt's alternative solution is to list in one **`<import>`** element all the components between which you want to keep connections so all these components and their trees (children and descendants) would be imported and connections maintained.

I do not like this method if only because it feels wrong. It's not clean, nor does it seem particularly efficient. It seems that we are just trying too hard to solve all use-cases with one solution, and I'm not sure it's possible. I think we need to re-establish exactly what we intend with the import method because at the moment I'm kind of losing sight of why we are doing this. Andre pointed out that if we actually had software, editors that read in CellML models, it would probably be quite easy to pick and choose which components we wanted from other models and have the editor just spit out a new CellML file.

Extending the component and units Elements

The one thing we all seemed to agree on was to extend the **`<component>`** and **`<units>`** elements to include a **`component_ref`** and **`units_ref`** attribute, respectively, instead of using the **`<component_import>`** and **`<units_import>`** elements within the **`<import>`** element. The reasoning behind this was that we're really just re-defining components and units: they are treated the same as other components and units throughout the model, with the exception that an imported component need not be declared as an encapsulated component to be treated as one. We are also giving ourselves the option of later deciding to allow the imported component to be modified.