

Modular SBML



Martin Ginkel
Max-Planck-Institute for Dynamics of complex technical Systems
Magdeburg, Germany

5th July 2002

Overview

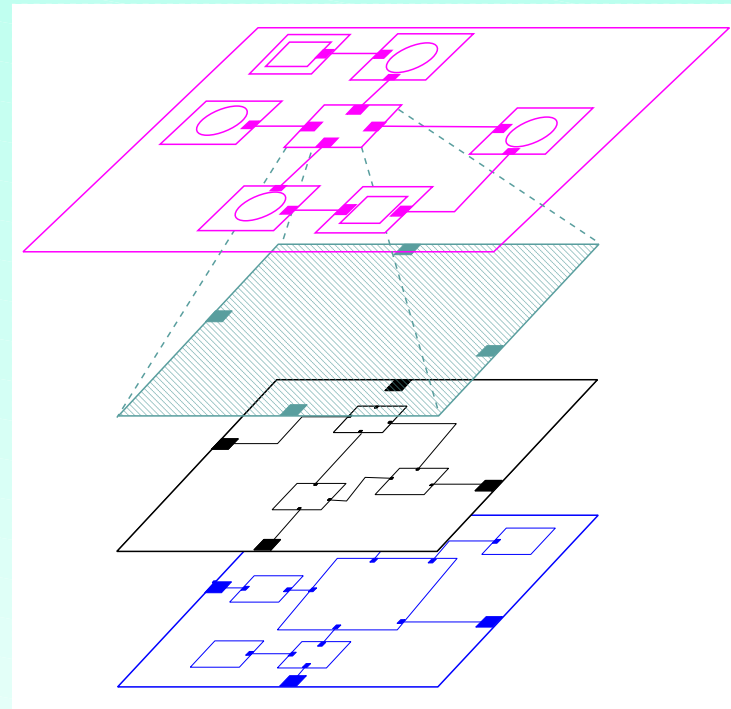
1. Why Modular Models?
2. What is needed?
3. How can this work in SBML?
4. Open Questions/Conclusions

Motivation: Why Modules?

- Systems Biology: Understanding the quantitative behaviour of complex cellular systems
- Models are complex networks with species and reactions
- Working in teams and division of tasks is necessary
- *Are flat and complex reaction maps the model of choice?*
- What about
 - abstractions from the reaction level
 - representations for higher order structures
 - reuse of (partial) models

Advantages of Modularity

1. Modular models are divided into comprehensible modules
 - unnecessary detail is hidden from the user
 - combining different partial models becomes easier
2. It is possible to separate interface and implementation of a module
 - Different implementations with the same interface \leadsto Exchangability
 - Example: Gene expression
 - PDE model for Polymerase movement
 - ODE model with time delay
 - transcription frequency = initiation frequency
 - Adjusting the detail level to the purpose of the model



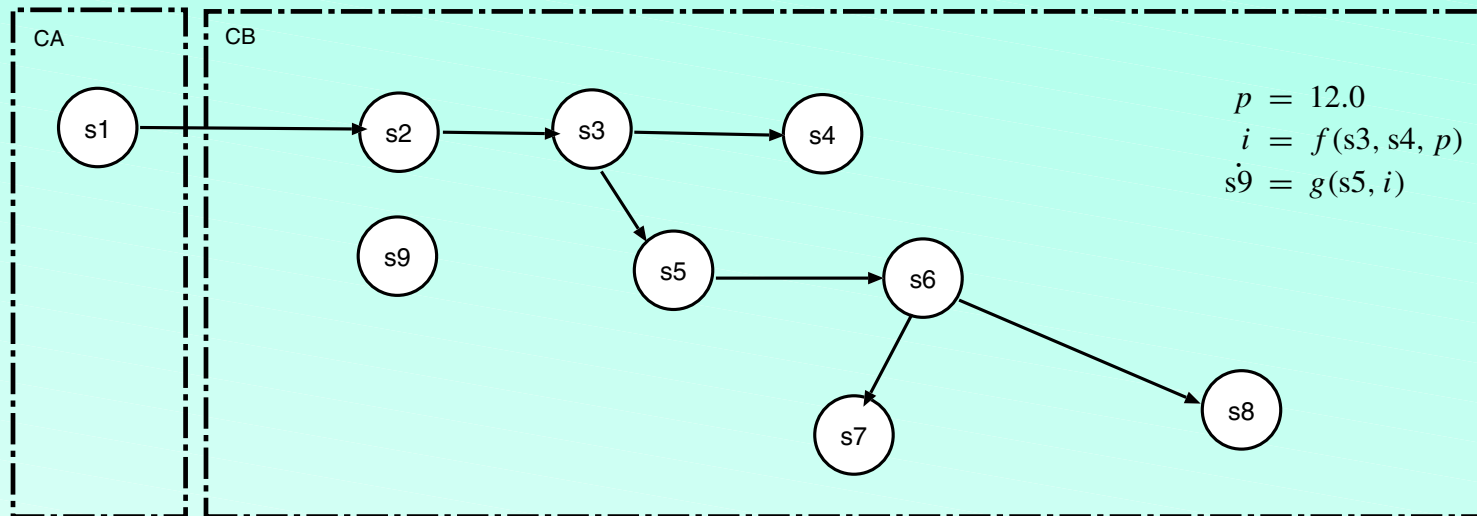
Advantages of Modularity

3. Debugging is easier with modularity
 - For debugging one can isolate the module of interest from the rest of the system and study the I/O behaviour in a test frame
 - It is possible to compare the behaviour of different implementations of a module
4. Established standardized models in libraries improve understanding and development of larger models very much
 - Abstraction of the detailed level possible
 - Not lots of reactions but named modules like e. g. „mapk-cascade“, which have to be connected and parametrized
 - reuse of modules becomes possible \rightsquigarrow modeling libraries

What is necessary for Modular Models?

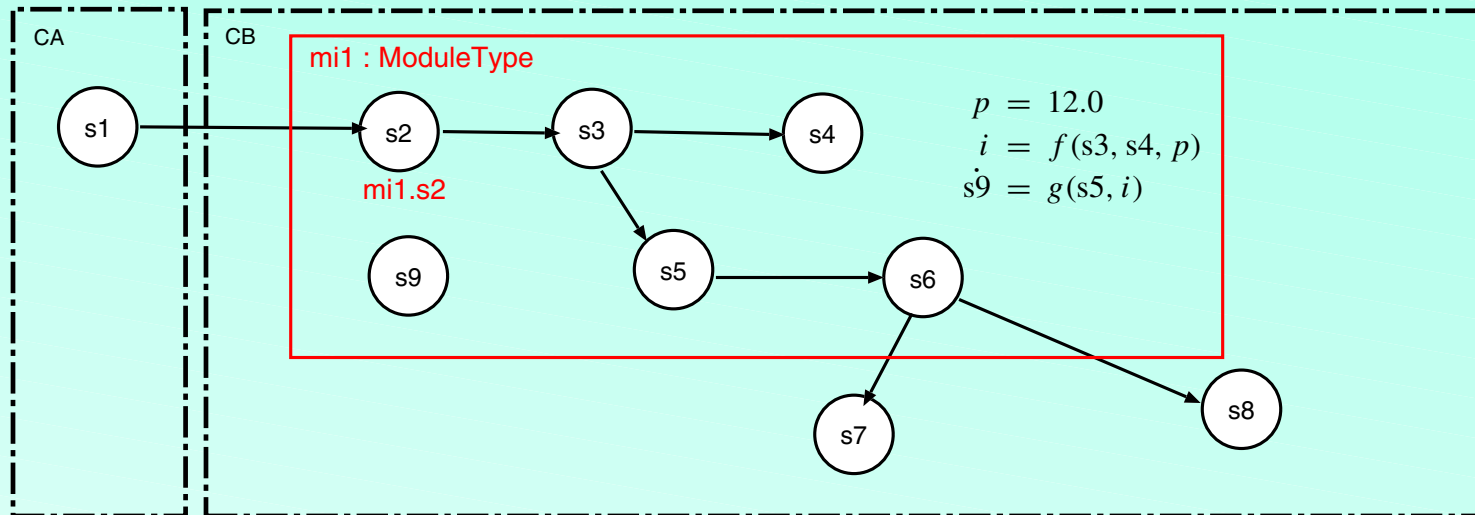
1. Modules: encapsulated logical (possibly physical) submodels with same set of elements as the SBML-Model
2. Namespaces: hierarchical names to access and specify parts of modules
3. Interface: Defined connection-elements to integrate a module into a larger model
4. Model Assembly: model-instantiation and connection
5. Parametrization: Adjusting initial and parameter values and compartments of model-instances

Status Quo



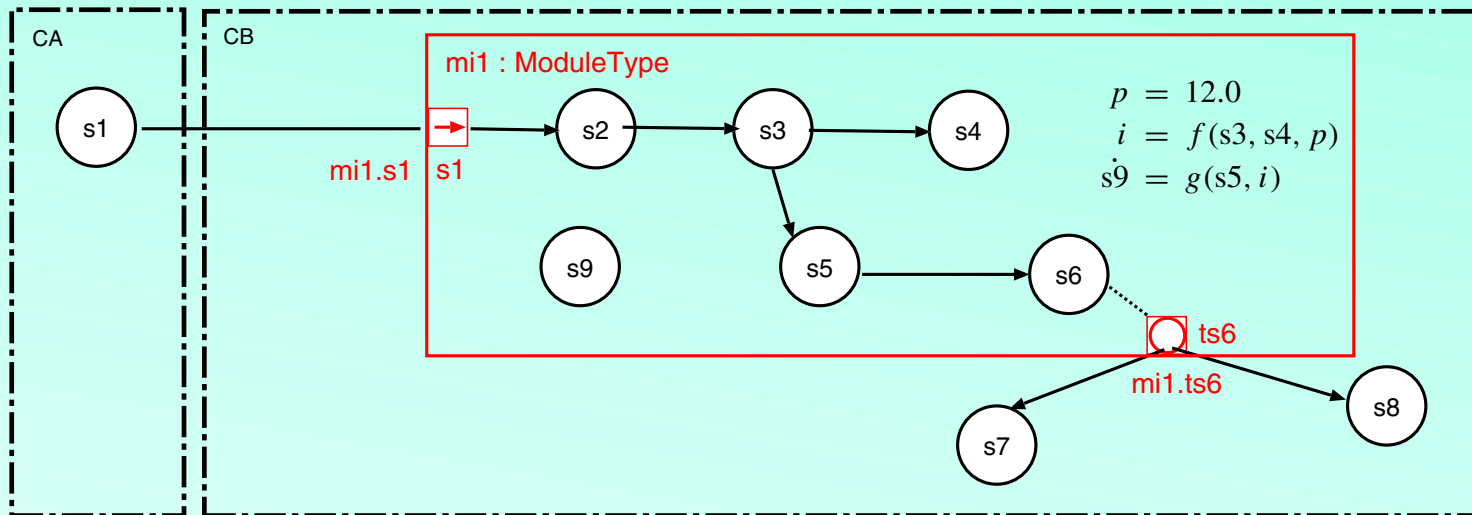
- species (S1. . . S9) attached to compartments (CA, CB)
- reactions connecting species with kinetic laws
- parameters (p, i)
- rules for parameters and species

Module separation and naming



- Module separated from the environment, interior parts can not refer to the outside
- Modules should be self-contained and can form *Hierarchies*
- Names: module-*Types* are named (unique per document) and get applied as *Instances* with an instance name (unique per parent-module)
- Names of elements inside a module are in a separate namespace, access from the outside through hierarchical names (Separator: .)

Interface definition

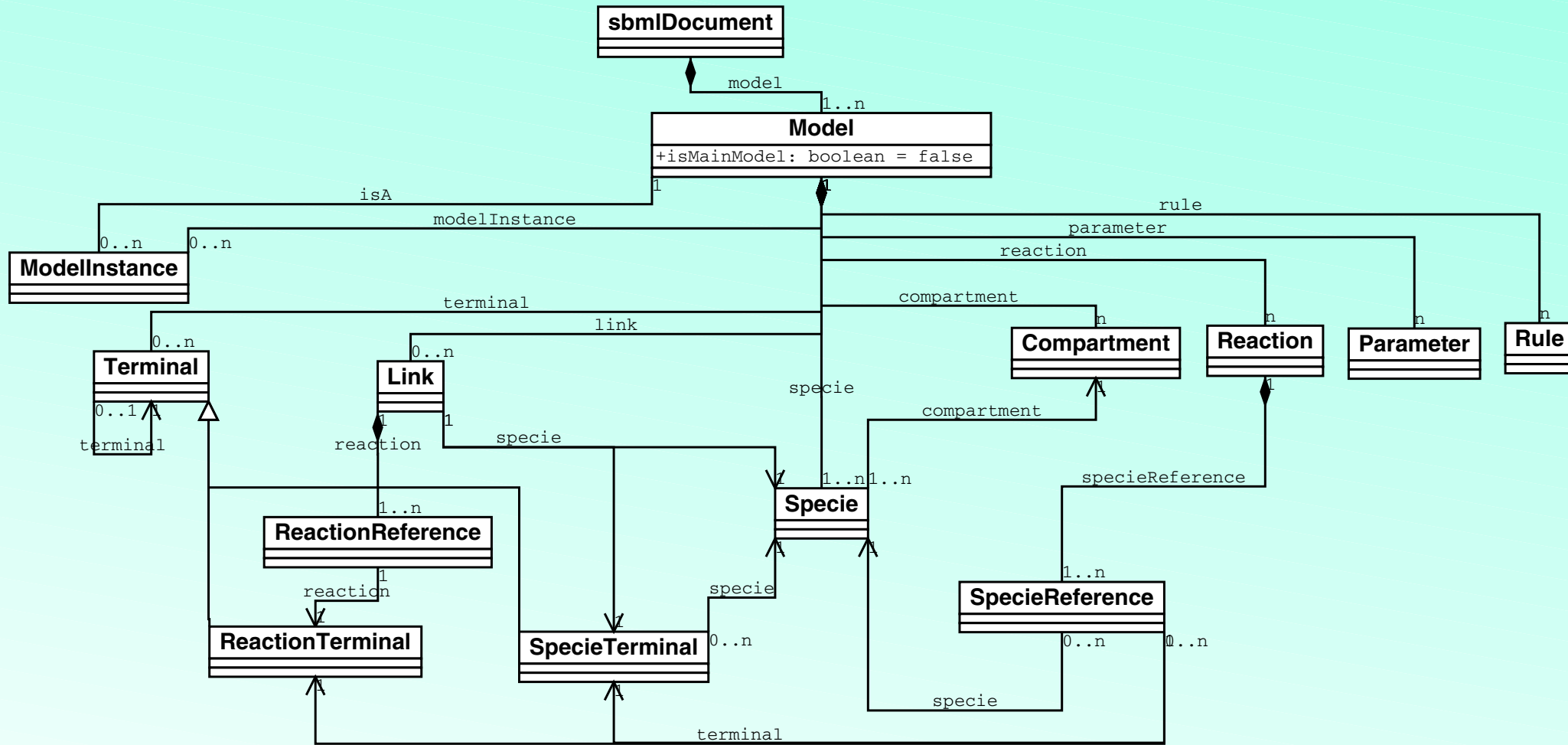


- Modules are connected with others through *Terminals*
- Specie-Terminals (mi1.ts6) make it possible to use a specie of the module
- Reaction-Terminals (mi1.s1) mimic a outside specie to inside reactions
- Terminals can be propagated through the module-hierarchy

Parametrizations

- Use of fixed modules instances only is unflexible!
- Necessary Specifications:
 - values of parameters (also in kinetic laws), initial amounts, volumes
 - compartment-assignment for species, substitution of compartments, outsides
- Possible Specifications:
 - Formulae for kinetic laws and rules
 - Module-types for inner module-instances (polymorph) (Not in Proposal)

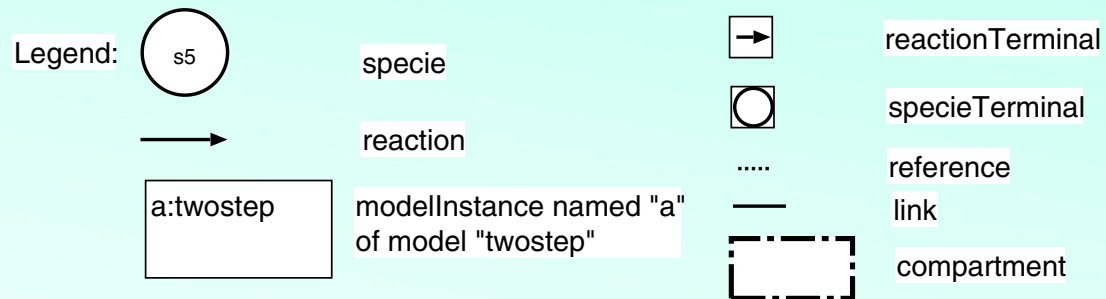
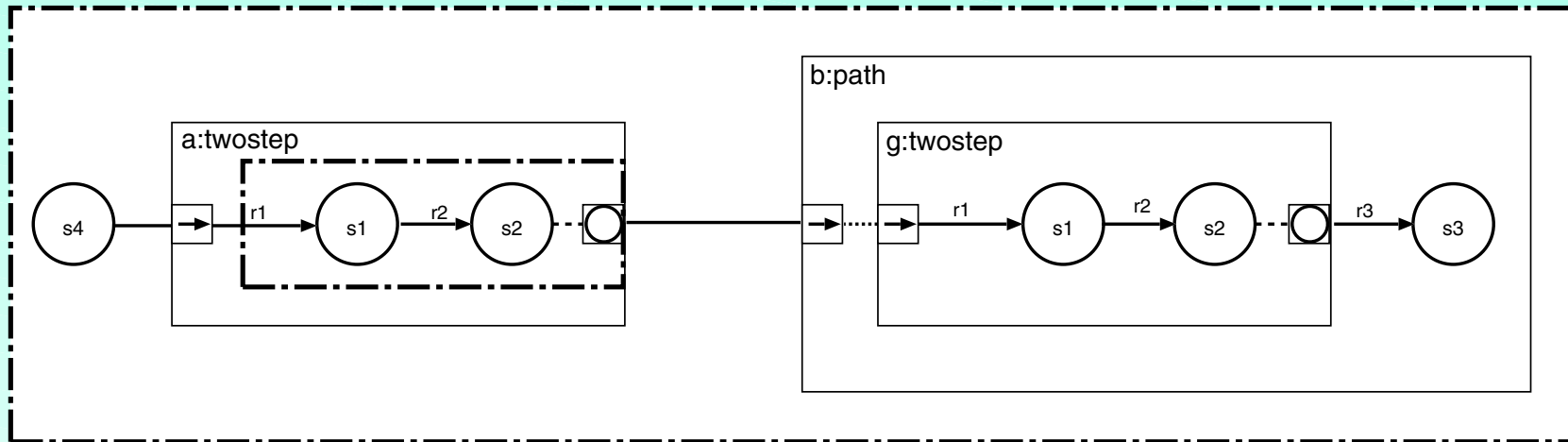
Possible Realization in SBML-X



Legend:

- ← Association
- ◁ Generalization
- ◆ Aggregation

Example Structure



Model in SBML

```
1  <sbml level="2" version="1" >
    <listOfModels>
3    <model name="twostep">
        ...
        <listOfTerminals>
            <reactionTerminal name="r1_t"/>
21    <specieTerminal name="s2_t" specie="s2"/>
        </listOfTerminals>
        <listOfReactions>
24    <reaction name="r1">
            <listOfReactants >
                <specieReference terminal="r1_t"/>
27    </listOfReactants>
        ...
```

Model in SBML

```
<listOfModelInstances>
  <modelInstance name="g" isa="twostep">
72   <listOfCompartmentSpecs>
      <compartmentSpec compartment="c1"
                          isSubstitutedBy="parent.c1"/>
75   </listOfCompartmentSpecs>
      <listOfSpeciesSpecs>
          <speciesSpec specie="s1" initialamount="100"/>
78          <speciesSpec specie="s2" initialamount="1"/>
      </listOfSpeciesSpecs>
      <listOfReactionSpecs>
81          <reactionSpec reaction="r1">
              <parameterSpec parameter="vm" value="300"/>
              </reactionSpec>
84      </listOfReactionSpecs>
      ...
```

Model in SBML

```
111 <model name="whole" isMainModel="true">
    ...
126     <listOfModelInstances>
        <modelInstance name="a" isa="twostep">
            <listOfCompartmentSpecs>
129                 <compartmentSpec compartment="c1" volume="0.1"
                    outside="parent.c1"/>
            </listOfCompartmentSpecs>
132     ...
147 <listOfLinks>
        <link specie="s4">
            <reactionReference reaction="a.r1_t"/>
150 </link>
        <link specie="a.s2_t">
            <reactionReference reaction="b.r1_t"/>
153 </link>
```

Open Questions

- units as a list of the sbml document (not in a model)
- distinction module vs. model
- separate namespace for (specie-) terminals
- additional specification possibilities
- overriding isA Attribute of modelinstance

Conclusions

- Modular modeling has several advantages for modelers
 - comprehensibility
 - maintainability
 - building of libraries
- Modular, hierarchical models are possible in SBML with named models and modelinstances
- terminals define interfaces to the models
- modelinstances can be specified and linked