

Systems Biology Graphical Notation: Process Description language Level 1

Version 2.0

Date: November 9, 2011

Disclaimer: This is a working draft of the SBGN Process Description Level 1 Version 2.0 specification. It is not a normative document.

Editors:

Stuart Moodie
Nicolas Le Novère
Emek Demir
Huaiyu Mi
Alice Viléger

EMBL European Bioinformatics Institute, UK
EMBL European Bioinformatics Institute, UK
Sloan-Kettering Institute, USA
University of Southern California, USA
London, UK

To discuss any aspect of SBGN, please send your messages to the mailing list sbgn-discuss@sbgn.org. To get subscribed to the mailing list or to contact us directly, please write to sbgn-editors@lists.sourceforge.net. Bug reports and specific comments about the specification should be entered in the issue tracker http://p.sf.net/sbgn/pd_tracker.



Contents

1	Introduction	1		
1.1	SBGN levels and versions	1	3.3.31	FluxArc
1.2	Developments, discussions, and notifications of updates	2	3.3.32	ConsumptionArc
1.3	Note on typographical convention	2	3.3.33	ProductionArc
			3.3.34	ModulationArc
			3.3.35	LogicArc
			3.3.36	EquivalenceArc
			3.3.37	CloneMarker
			3.3.38	SimpleCloneMarker
			3.3.39	LabelledClonerMarker
2	Concepts	3	3.4	Controlled vocabularies
2.1	Definitions and Nomenclature	4	3.4.1	Entity pool node material types
2.1.1	Language versus notation	4	3.4.2	Entity pool node conceptual types
2.1.2	What are the languages?	4	3.4.3	Macromolecule covalent modifications
2.1.3	Nomenclature	4	3.4.4	Physical characteristics
2.1.4	Graph, diagram or map?	5	3.5	Entity Pool Node Identity and Cloning
2.2	The Language Definition	5	3.6	Map and Submap Linking
			3.6.1	Compartment spanning
3	Concepts and Glyphs	6	4	Layout Rules for a Process Description
3.1	Introduction	6	4.1	Introduction
3.2	How to read the Language Specification	6	4.2	Requirements
3.3	Definitions	7	4.2.1	Node-node overlaps
3.3.1	SBGNElement	7	4.2.2	Node-edge crossing
3.3.2	Map	8	4.2.3	Node border-edge overlaps
3.3.3	SBGNGlyph	8	4.2.4	Edge-edge overlaps
3.3.4	AuxiliaryUnit	9	4.2.5	Node orientation
3.3.5	SBGNNode	10	4.2.6	Node-edge connection
3.3.6	SBGNArc	10	4.2.7	Node labels
3.3.7	EntityPoolNode	11	4.2.8	Edge labels
3.3.8	Empty Set	12	4.2.9	Compartments
3.3.9	StatelessEPN	13	4.3	Recommendations
3.3.10	Simple chemical	14	4.3.1	Node-edge crossing
3.3.11	UnspecifiedEntity	15	4.3.2	Labels
3.3.12	Perturbing Agent	16	4.3.3	Avoid edge crossings
3.3.13	StatefulEPN	17	4.3.4	Branching of <i>association</i> and <i>dissociation</i>
3.3.14	Macromolecule	18	4.3.5	Units of information
3.3.15	NucleicAcidFeature	19	4.4	Additional suggestions
3.3.16	Complex	20		
3.3.17	Subunit	22	5	Acknowledgments
3.3.18	ProcessNode	24	5.1	Level 1 Release 1.0
3.3.19	NonStoichiometricProcess	25	5.2	Level 1 Release 1.1
3.3.20	Phenotype	25	5.3	Level 1 Release 1.2
3.3.21	SubmapNode	26	5.4	Level 1 Release 1.3
3.3.22	LogicalOperator	27	5.5	Comprehensive list of acknowledgements
3.3.23	StoichiometricProcess	29	5.6	Financial Support
3.3.24	Compartment	35		
3.3.25	AttributeValue	37	A	Complete examples of Process Description Maps
3.3.26	StateVariable	38		
3.3.27	Annotation	40		
3.3.28	CrossReference	41		
3.3.29	SubmapTerminal	42		
3.3.30	Tag	43		

B Reference card	73	C.5 State and transformation of compartments	76
C Issues postponed to future levels	75	D Revision History	77
C.1 Multicompartment entities	75	D.1 Version 1.0 to Version 1.1	77
C.2 Logical combination of state variable values	75	D.2 Version 1.1 to Version 1.2	78
C.3 Non-chemical entity nodes	75	D.3 Version 1.2 to Version 1.3	78
C.4 Generics	76		

Chapter 1

Introduction

The goal of the Systems Biology Graphical Notation (SBGN) is to standardize the graphical/visual representation of biochemical and cellular processes. SBGN defines comprehensive sets of symbols with precise semantics, together with detailed syntactic rules defining their use. It also describes the manner in which such graphical information should be interpreted. For a general description of SBGN, one can read:

Nicolas Le Novère, Michael Hucka, Huaiyu Mi, Stuart Moodie, Falk Schreiber, Anatoly Sorokin, Emek Demir, Katja Wegner, Mirit I Aladjem, Sarala M Wimalaratne, Frank T Bergman, Ralph Gauges, Peter Ghazal, Hideya Kawaji, Lu Li, Yukiko Matsuoka, Alice Villéger, Sarah E Boyd, Laurence Calzone, Melanie Courtot, Ugur Dogrusoz, Tom C Freeman, Akira Funahashi, Samik Ghosh, Akiya Jouraku, Sohyoung Kim, Fedor Kolpakov, Augustin Luna, Sven Sahle, Esther Schmidt, Steven Watterson, Guanming Wu, Igor Goryanin, Douglas B Kell, Chris Sander, Herbert Sauro, Jacky L Snoep, Kurt Kohn & Hiroaki Kitano. The Systems Biology Graphical Notation. *Nature Biotechnology* **27**, 735 - 741 (2009). <http://dx.doi.org/10.1038/nbt.1558>

This document defines the *Process Description* visual language of SBGN. Process Descriptions are one of three views of a biological process offered by SBGN. It is the product of many hours of discussion and development by many individuals and groups.

1.1 SBGN levels and versions

It was clear at the outset of SBGN development that it would be impossible to design a perfect and complete notation right from the beginning. Apart from the prescience this would require (which, sadly, none of the authors possess), it also would likely need a vast language that most newcomers would shun as being too complex. Thus, the SBGN community followed an idea used in the development of other standards, i.e. stratify language development into levels.

A *level* of one of the SBGN languages represents a set of features deemed to fit together cohesively, constituting a usable set of functionality that the user community agrees is sufficient for a reasonable set of tasks and goals. Within *levels*, *versions* represent small evolution of a language, that may involve new glyphs, refined semantics, but no fundamental change of the way maps are to be generated and interpreted. Capabilities and features that cannot be agreed upon and are judged insufficiently critical to require inclusion in a given level, are postponed to a higher level or version. In this way, the development of SBGN languages is envisioned to proceed in stages, with each higher levels adding richness compared to the levels below it.

1.2 Developments, discussions, and notifications of updates

The SBGN website (<http://sbgn.org/>) is a portal for all things related to SBGN. It provides a web forum interface to the SBGN discussion list (sbgn-discuss@caltech.edu) and information about how anyone may subscribe to it. The easiest and best way to get involved in SBGN discussions is to join the mailing list and participate.

Face-to-face meetings of the SBGN community are announced on the website as well as the mailing list. Although no set schedule currently exists for workshops and other meetings, we envision holding at least one public workshop per year. As with other similar efforts, the workshops are likely to be held as satellite workshops of larger conferences, enabling attendees to use their international travel time and money more efficiently.

Notifications of updates to the SBGN specification are also broadcast on the mailing list and announced on the SBGN website.

1.3 Note on typographical convention

The concept represented by a glyph is written using a normal font, while a *glyph* means the SBGN visual representation of the concept. For instance “a biological process is encoded by the SBGN PD *process*”.

Chapter 2

50

Concepts

51

To set the stage for what follows, we first give a brief overview of some of the concepts in the Process Description language with the help of an example shown in Figure 2.1.

52

53

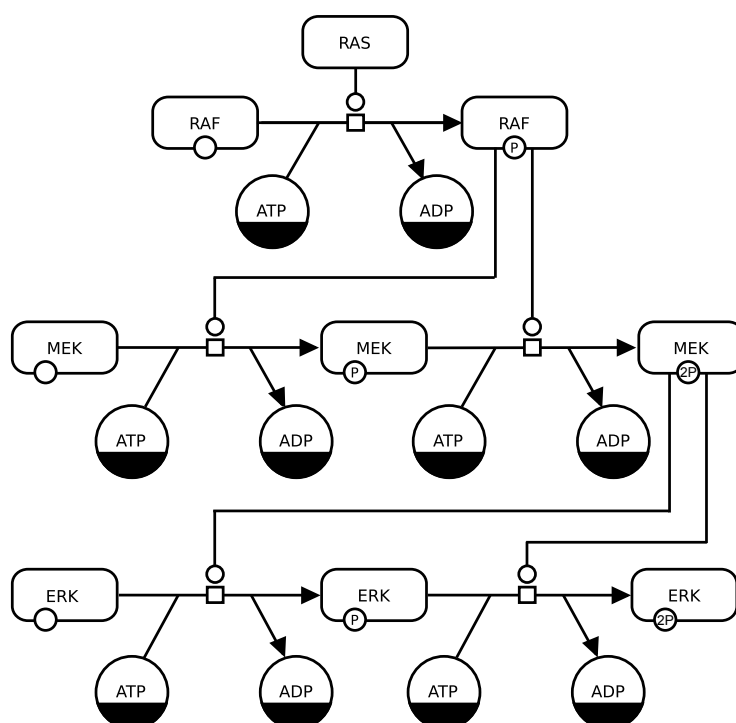


Figure 2.1: This example of a Process Description uses two kinds of entity pool nodes: one for pools of different macromolecules (Section 3.3.14) and another for pools of simple chemicals (Section 3.3.10). Most macromolecule nodes in this map are adorned with state variables (Section 3.3.26) representing phosphorylation states. This map uses one type of process node, the process node (Section 3.3.23), and three kind of connecting arc, consumption (Section 3.3.32), production (Section 3.3.33) and catalysis (Section 3.3.34). Finally, some entity pool nodes have dark bands along their bottoms; these are clone markers (Section 3.3.37) indicating that the same pool nodes appear multiple times in the map.

The map in Figure 2.1 is a simple map for part of a mitogen-activated protein kinase (MAPK) cascade. The larger nodes in the figure (some of which are in the shape of rounded rectangles and others in the shape of circles) represent biological materials—things like macromolecules and simple chemicals. The biological materials are altered via processes, which are indicated in Process Description language by lines with arrows and other decorations. In this particular map, all of the processes happen to be the same: processes catalyzed by biochemical entities. The directions of the arrows indicate the direction of the processes; for example, unphosphorylated RAF kinase processes to phosphory-

54

55

56

57

58

59

60

lated RAF kinase via a process catalyzed by RAS. Although ATP and ADP are shown as incidental to the phosphorylations on this particular graph, they are involved in the same process as the proteins getting phosphorylated. The small circles on the nodes for RAF and other entity pools represent state variables (in this case, phosphorylation sites).

The essence of the Process Descriptions is *change*: it shows how different entities in the system process from one form to another. The entities themselves can be many different things. In the example of Figure 2.1, they are either pools of macromolecules or pools of simple chemicals, but as will become clear later in this chapter, they can be other conceptual and material constructs as well. Note also that we speak of *entity pools* rather than individuals; this is because in biochemical network models, one does not focus on single molecules, but rather collections of molecules of the same kind. The molecules in a given pool are considered indistinguishable from each other. The way in which one type of entity is transformed into another is conveyed by a *process node* and links between entity pool nodes and process nodes indicate an influence by the entities on the processes. In the case of Figure 2.1, those links describe consumption Section 3.3.32, production Section 3.3.33 and catalysis Section 3.3.34, but others are possible. Finally, nodes in Process Descriptions are usually not repeated; if they do need to be repeated, they are marked with *clone markers*—specific modifications to the appearance of the node (Section 3.3.37). The details of this and other aspects of Process Description notation are explained in the following chapters.

2.1 Definitions and Nomenclature

2.1.1 Language versus notation

SBGN specifications propose symbols, ways to organise them, but also semantic rules to analyse the resulting representations. SBGN "drawings" can be translated into English, but also into computer readable formats. Those specifications really propose true languages. SBGN is therefore made up of three languages.

2.1.2 What are the languages?

PD is a language that permits the description of all the processes taking place in a biological system. The ensemble of all these processes constitute a Description. **ER** is a language that permits the description of all the relations involving the entities of a biological system. The ensemble of all these relations constitute a Relationship. **AF** is a language that permits the description of the flow of activity in a biological system.

2.1.3 Nomenclature

The three languages of SBGN should be referred to as:

- the Process Description language.
- the Entity Relationship language.
- the Activity Flow language.

Abbreviated as:

- the PD language.
- the ER language.
- the AF language.

A specific representation of a biological system in one of the SBGN languages should be referred to as:

- a Process Description map.
- an Entity Relationship map.
- an Activity Flow map.

Abbreviated as:

- a PD map.
- an ER map.
- an AF map.

The corpus of all SBGN representations should be referred to as:

- Process Descriptions.
- Entity Relationships.
- Activity Flows.

The capitalization is important. PD, ER and AF are names of languages. As such they must be capitalized in English. This is not the case of the accompanying noun (language or map).

2.1.4 Graph, diagram or map?

A graph is a very technical term that belongs to mathematics and is uncommon in biology. Diagram is a concept that encompasses more than just graph. Examples are Venn diagrams for instance. Therefore, we recommend using the term map for SBGN representations. Those representations effectively permit users to travel and orient themselves in a biological system. Map is also the term most frequently used by the different communities, whether in metabolism, signaling or genomics.

2.2 The Language Definition

Language defined as a set of classes that represent the key concepts of the language. Done in UML. The top level classes are roughly equivalent to the grapheme, which is the basic language unit. The grapheme can then be represented by one or more glyph. For example in English the grapheme of 'a' can be represented by the glyphs *a*, *a* or *a*. In this specification we refer to a concept that organises the language as an *abstract class* and a top-level concept, i.e., a grapheme, as a *concrete class* or both generically as just a *class*. The graphical representation of the concrete class is the *glyph*.

Chapter 3

Concepts and Glyphs

3.1 Introduction

Although ultimately defined by its glyphs, SBGN Process Description Level 1 these glyphs represent concepts that are related to varying degrees and ultimately can be organised hierarchically. This provides us with a useful way of organising the glyphs and thinking about Process Description. Therefore in this chapter we describe the conceptual structure of SBGN Process Description Level 1 and place each glyph within this structure. For each glyph or concept we describe its physical appearance and define syntax and other usage rules. An overview of this hierarchy is provided in figure 3.1.

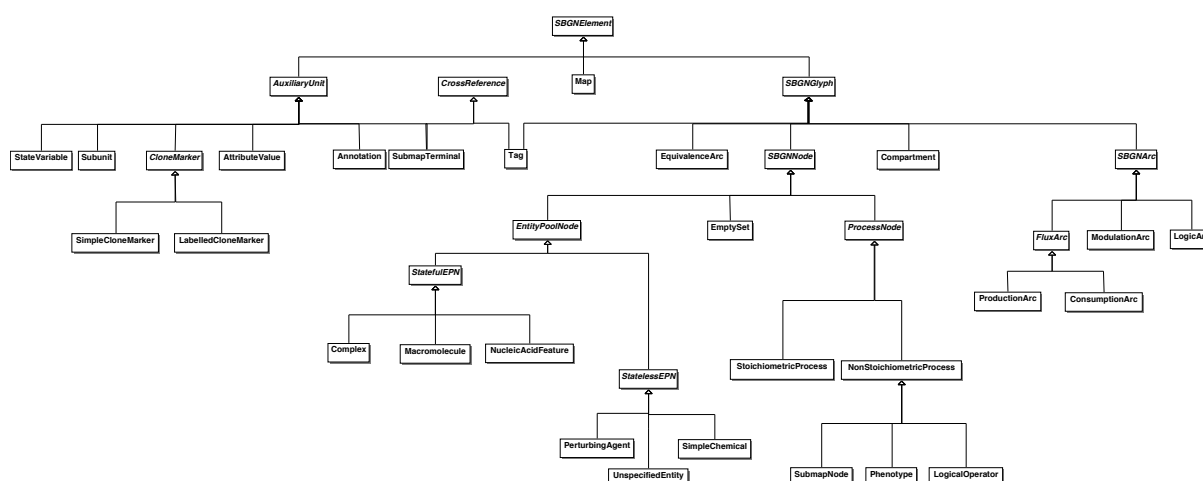


Figure 3.1: Organisation of the node glyphs within SBGN Process Description language. All UML classes (boxes) correspond to Process Description node glyphs except those with italicised names, which are organisational groupings. They correspond to the groupings used elsewhere in this document.

3.2 How to read the Language Specification

- Organised alphabetically.
- use UML to describe the language elements.
- Some language elements describe concepts and rules, some describe elements that are represented by glyphs, other both.
- Class represent concepts.

- Classes have attributes which are required (R), optional (O), or can be a collection of a number of values. In the latter case the permitted range is specified as:
 - * any number
 - 1..* 1 or more
- Associations are indicated by solid lines. If there is a black diamond then this means the class the diamond is closest to contains the other class. This means the contained class cannot exist without the parent class, i.e., if the containing class instance is destroyed then the contained is too.
- open diamonds are a referencing association, the referenced class can exist without the referencing class.
- dependency indicates that there is an indirect association between these classes.
- direction containment or referencing means that one side sees the other, but the other does not see the referencing class.
- cardinality on the edges of the class refers to the distant class and indicates the following:
 - 1 mandatory that one instance is associated.
 - * optional - any number of instances permitted in association
 - 1..* mandatory that at least 1 is associated
 - 0..1 Optional. instance is optional.
- class defines all possible cases
- instance is a single case of the class with attributes and associates set. In SBGN case a glyph is an instance of a class.
- Idea is that UML defines language elements and then the notation section specifies the graphical representation. Equivalent to the grammar of a language and the letters used to write it.
- aim is to minimise duplication of rules as much as possible to improve maintainability. Compromised between that and readability and ease of understanding spec.

3.3 Definitions

3.3.1 SBGNElement

All the glyphs in SBGN Process Description Level 1 inherit from SBGNElement. This is an abstract or conceptual class that helps organise Process Description conceptually. SBGNElement (figure 3.2) has a single attribute id that is an identifying attribute. This means that all SBGN elements defined here, which ultimately extend SBGNBase, can all be uniquely identified from each other. This makes sense if you think that a glyph drawn on a map is distinct from another glyph drawn on the map. The id attribute reflects this and is not shown explicitly in a Process Description map.

Generalisation

None

Attributes

id: identifier (**R**) uniquely identifies all SBGN elements in the same namespace.

Changes from Previous Version

Not defined in the previous version.

3.3.2 Map

187

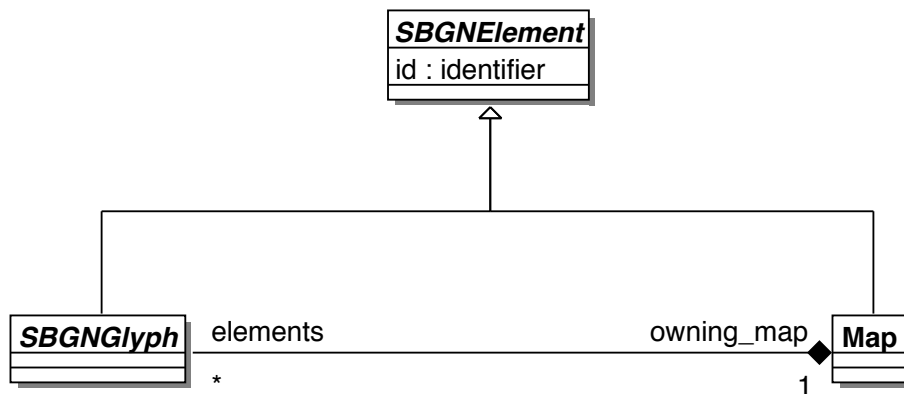


Figure 3.2: UML definition of Map and SBGNElement.

The Map (figure 3.2) is a container that holds all the glyphs drawn in a Process Description map. A Map may contain a SubmapNode (see section 3.3.21) in which case some of the detail is summarised in another Process Description map. In this case the former is referred to as the “Main Map” and the latter the “Submap”. A Main Map shares the same namespace as its Submap and uses the CrossReference (see section 3.3.28) identify the interface between both maps. The details of Main Map and Submap linking semantics are described in more detail in section 3.6.

Generalisation

- SBGNElement (see section 3.3.1)

Attributes

No additional attributes

Associations

elements:SBGNGlyph (*) The collection of glyphs held by the map.

Rules and Constraints

- A map is valid if it is empty (although not very useful).
- All instances of SBGNGlyph (see section 3.3.3) must be unique (see section 3.5).

Notation

The map is the canvas upon which the Process Description language is drawn. It’s only visible feature is its colour. It can take any pattern or colour (or be transparent for that matter), but as SBGN is ‘colour blind’ this does not convey any meaning in itself.

Changes from Previous Version

Not defined explicitly in previous versions.

3.3.3 SBGNGlyph

Glyphs are the fundamental building blocks of the Process Description language and are the only elements that can be drawn directly on a map (Map).

Generalisation

- SBGNElement (see section 3.3.1)

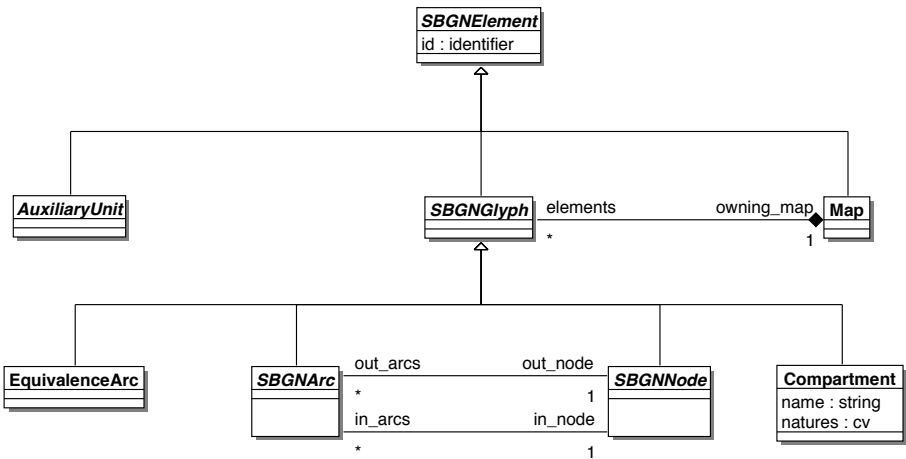


Figure 3.3: UML definition of the Auxiliary Unit and its subclasses.

Attributes	214
No additional attributes.	215
Associations	216
owning_map:Map (1) The map that contains this class.	217
Rules and Constraints	218
No additional rules and constraints.	219
Changes from Previous Version	220
Not defined in previous version.	221
3.3.4 AuxiliaryUnit	222

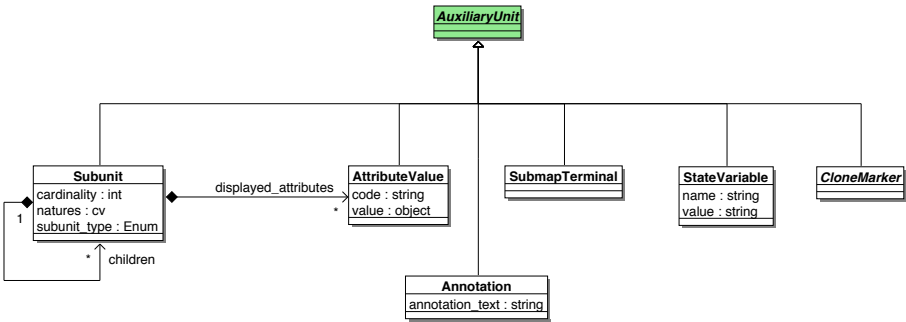


Figure 3.4: UML definition of the Auxiliary Unit and its subclasses.

The AuxiliaryUnit (figure 3.4) represents symbols that may be used to adorn glyphs. In doing so they change the meaning of the glyph and/or provide additional information about it.	223
	224
Generalisation	225
• SBGNElement (see section 3.3.1)	226
Attributes	227
No additional attributes.	228

Associations

229

No additional associations.

230

Rules and Constraints

231

No additional rules and constraints.

232

Changes from Previous Version

233

Not defined in previous version.

234

3.3.5 SBGNNode

235

The SBGNNode (figure 3.3) represents the nodes in the graph structure that is the core representation within Process Description language. The nodes are connected to glyphs descended from SBGNArc for form a direct graph.

236

237

238

Generalisation

239

- SBGNGlyph (see section 3.3.3)

240

Attributes

241

No additional attributes.

242

Associations

243

out_arcs:SBGNArc (*) The arcs that are leaving this node.

244

in_arcs:SBGNArc (*) The arcs that are entering this node.

245

Rules and Constraints

246

No additional rules and constraints.

247

Changes from Previous Version

248

Not defined in the previous version.

249

3.3.6 SBGNArc

250

The SBGNArc (figure 3.3) represents the arcs (also know as edges) in the graph structure that is the core representation within Process Description language. The arc is connected to two nodes descended from SBGNNode, one at each end. As the arc has a direction (directed arc) these nodes are by convention designated the *out node* to indicate the arc is leaving the node and *in node* to indicate that it is entering the node.

251

252

253

254

255

Generalisation

256

- SBGNGlyph (see section 3.3.3)

257

Attributes

258

No additional attributes.

259

Associations

260

out_node:SBGNNode (1) The node that this arc is leaving.

261

in_node:SBGNNode (1) The node that this arc is entering.

262

Rules and Constraints

263

No additional rules and constraints.

264

Changes from Previous Version

265

Not defined in the previous version.

266

3.3.7 EntityPoolNode

An entity pool is a population of entities that cannot be distinguished from each other, when it comes to the SBGN Process Description Level 1 map. For instance all the molecular entities that fulfill the same role in a given process form an entity pool. As a result, an entity pool can represent different granularity levels, such as all the proteins, all the instances of a given protein, only certain forms of a given protein. To belong to a different compartment is sufficient to belong to different entity pools. Calcium ions in the endoplasmic reticulum and calcium ions in the cytosol belong to different entity pools when it comes to representing calcium release from the endoplasmic reticulum.

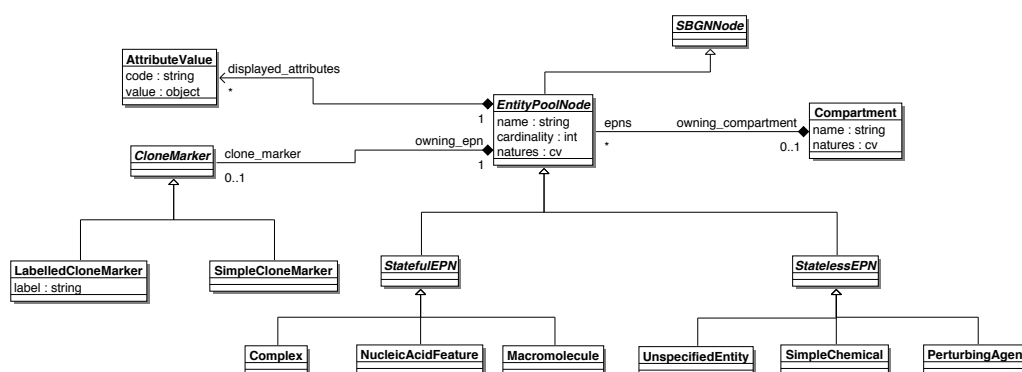


Figure 3.5: UML definition of the entity pool node and its descendant glyphs.

The EntityPoolNode (figure 3.5) is the common ancestor of all these glyphs and defines attributes that are common to all its descendants. It must belong to a compartment (c.f. section 3.3.24) and can contain a clone marker if it is cloned. Note that not all EPNs can be cloned.

Generalisation

- SBGNNode (see section 3.3.5)

Attributes

label: string (R) The name that identifies the entity in the Process Description map. EPNs with the same label should be from the same entity. the string cannot be empty and must start and end with a non-space character. Any Unicode character is acceptable.

cardinality: int (R) The number of copies of the entity. Must be a positive non-zero integer.

natures: cv (O) The nature of the entity pool node as defined by a controlled vocabulary. Zero, one or more values may be set, but each one must belong to a different controlled vocabulary.

Associations

owning_compartment:Compartment (0..1) The compartment that this EPN belongs too.

clone_marker:CloneMarker (0..1) The clone marker decorator. See section 3.3.37 for its use.

displayed_attributes:AttributeValue (*) One or more decorators used to display attribute values.

Logical Identity

[Logical Key:

- owning_compartment
- name
- cardinality
- natures

Rules and Constraints

- If cardinality > 1 then the descendant glyph must be displayed as a multimer.

- If the EPN is drawn directly on a *Map* then `owning_compartment` is assigned to an invisible default compartment. *nature* can only use the material type (section 3.4.1), conceptual type (section 3.4.2) or physical characteristics (section 3.4.4) controlled vocabularies.
- The appropriate subclass of `CloneMarker` must be used to distinguish logically identical instances of this class.

Notation

Although there is no direct graphical representation of this class this class does affect the appearance of the `AttributeValue` and its associated glyph the *Unit of Information*. The *Unit of Information* can be used to present the cardinality and natures attributes. These used the following codes to indicate which attribute is being presented”

pc container physical characteristic

mt entity pool material type

ct entity pool conceptual type

N multimer cardinality

Changes from Previous Version

Not defined in the previous version.

3.3.8 Empty Set

It is useful to have the ability to represent the creation of an entity or a state from an unspecified source, that is, from something that one does not need or wish to make precise. For instance, in a model where the production of a protein is represented, it may not be desirable to represent all of the amino acids, sugars and other metabolites used, or the energy involved in the protein's creation. Similarly, we may not wish to bother representing the details of the destruction or decomposition of some biochemical species into a large number of more primitive entities, preferring instead to simply say that the species “disappears into a sink”. Yet another example is that one may need to represent an input (respectively, output) into (resp. from) a compartment without explicitly representing a transport process from a source (resp. to a target).

For these and other situations, SBGN defines a single glyph to handle these situations representing the involvement of an external pool of entities. The symbol used in SBGN is borrowed from the mathematical symbol for “empty set”, but it is important to note that it does not actually represent a true absence of everything or a physical void—it represents the absence of the corresponding structures in the model, that is, the fact that the external pool is conceptually outside the scope of the map.

A frequently asked question is, why bother having an explicit symbol at all? The reason is that one cannot simply use an arc that does not terminate on a node, because the dangling end could be mistaken to be pointing to another node in the map. This is specially true if the map is rescaled, causing the spacing of elements in the map to change. The availability and use of an explicit symbol for sources and sinks is critical.

The definition of the *Empty Set* is shown in figure 3.6. The empty set is not an EPN as it does not represent a single pool of entities and does not share any of the other attributes of an EPN, nor does it belong to a particular compartment.

Generalisation

- `SBGNNode` (see section 3.3.5)

Attributes

No additional attributes.

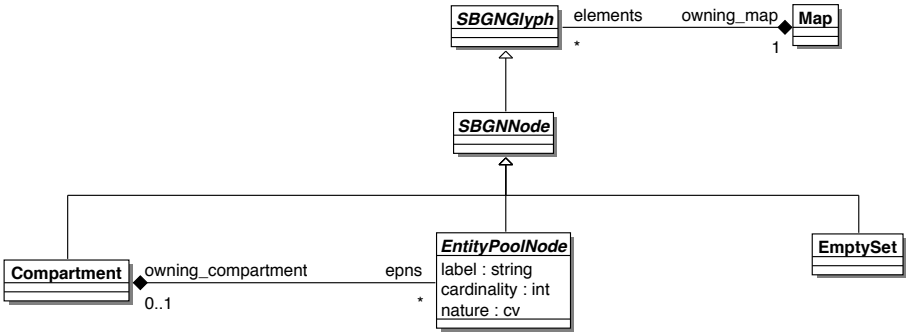


Figure 3.6: The UML definition of the EmptySet and its context in relation to other elements of the Process Description language.

Associations

No additional associations.

Rules and Constraints

- All instances of *Empty Set* can be regarded as identical therefore not special decoration is used to indicate replication on the map.

Notation

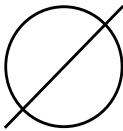


Figure 3.7: The empty set glyph.

Glyph: *Empty Set*

SBO Term: SBO:0000291 ! empty set

Auxiliary Items: None

Container: Represented by the mathematical symbol for “empty set”, that is, a circle crossed by a bar linking the upper-right and lower-left corners of an invisible square drawn around the circle (\emptyset). Figure 3.7 illustrates this. The symbol should be linked to one and only one edge in a map.

Label: None

Changes from Previous Version

The EmptySet and *Empty Set* glyph has replaced the *Source* and *Sink* glyphs. This symbols used remains the same, but the underlying concept has changed. The *Source* and *Sink* glyphs where types of EPN.

3.3.9 StatelessEPN

The StatelessEPN (figure 3.8) represents a pool where the entities do not change ‘state’. In other-words the entities do not undergo any physical change that is useful to record in a Process Description map. Therefore these glyphs cannot be assigned a state-variable.

Generalisation

- EntityPoolNode (see section 3.3.7)

Attributes

No additional attributes.

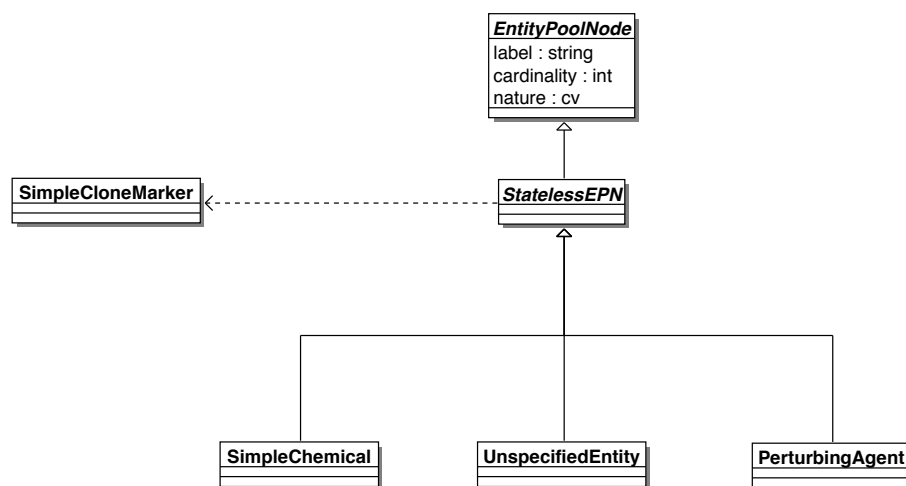


Figure 3.8: UML definition of the stateless entity pool node and its descendant glyphs.

Associations

No additional associations.

Rules and Constraints

- if a clone marker is used it must be of type SimpleCloneMarker.

Changes from Previous Version

Not defined in the previous version.

3.3.10 Simple chemical

A simple chemical in SBGN is defined as the opposite of a macromolecule (Section 3.3.14): it is a chemical compound that *not* formed by the covalent linking of pseudo-identical residues. Examples of simple chemicals are an atom, a monoatomic ion, a salt, a radical, a solid metal, a crystal, etc. The complex can be represented by a monomeric glyph (*Simple chemical monomer*) and a multimeric glyph (*Simple chemical multimer*).

Generalisation

- StatelessEPN (see section 3.3.9)

Attributes

No additional attributes.

Associations

No additional associations.

Rules and Constraints

No additional rules and constraints.

Notation

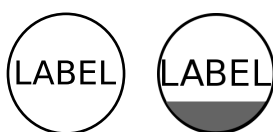


Figure 3.9: The Process Description glyph for simple chemical.

Glyph: *Simple chemical monomer*

SBO Term: SBO:0000247 ! simple chemical

Container: A *simple chemical* is represented by a circular container, as depicted in Figure 3.9. To avoid confusion with the Unspecified Entity (3.3.11), this glyph must remain a circle and cannot be deformed into an eclipse.

Label: The identification of the *simple chemical* is carried by an unbordered box containing a string of characters. The characters may be distributed on several lines to improve readability, although this is not mandatory. The label box has to be attached to the center of the circular container. The label is permitted to spill outside the container.

Auxiliary Items:

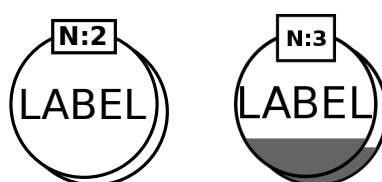


Figure 3.10: The Process Description glyph for multimer.

Glyph: *Simple chemical multimer*

SBO Term: SBO:0000421 ! multimer of simple chemicals

Container: A *simple chemical multimer* is represented by two identical containers shifted horizontally and vertically and stacked one on top of the other. Figure 3.10 illustrates the glyph.

Label: A *multimer* has no identity on its own. However, the first of the monomers carries an identifying label. The label is placed in an unbordered box containing a string of characters. The characters can be distributed on several lines to improve readability, although this is not mandatory. The label box must be attached to the center of the top monomer's container. The label may spill outside of the container.

Auxiliary Items:

Changes from Previous Version

The glyphs used for the SimpleChemical have been changed. Previously the glyph was a circle.

3.3.11 UnspecifiedEntity

The simplest type of EPN is the UnspecifiedEntity: one whose type is unknown or simply not relevant to the purposes of the map. This arises, for example, when the existence of the entity has been inferred indirectly, or when the entity is merely a construct introduced for the needs of a map, without direct biological relevance. These are examples of situations where the UnspecifiedEntity is appropriate. (Conversely, for cases where the identity of the entities composing the pool is known, there exist other, more specific glyphs described elsewhere in the specification.)

Generalisation

- StatelessEPN (see section 3.3.9)

Attributes

No additional attributes.

Associations

No additional associations.

Rules and Constraints

- The UnspecifiedEntity cannot have cardinality > 1. This means there is no multimer glyph.

Notation**Glyph: *Unspecified entity***

SBO Term: SBO:0000285 ! material entity of unspecified nature

Container: An *unspecified entity* is represented by an elliptic container, as shown in 3.11. Note that this must remain an ellipse to avoid confusion with the Simple Chemical glyph, which is a circle (c.f. 3.3.10).

Label: An *unspecified entity* is identified by a label placed in an unbordered box containing a string of characters. The characters can be distributed on several lines to improve readability, although this is not mandatory. The label box must be attached to the center of the container. The label may spill outside of the container.



Figure 3.11: *The Process Description glyph for unspecified entity.*

Changes from Previous Version

No changes from the previous version.

3.3.12 Perturbing Agent

Biochemical networks can be affected by external influences. Those influences can be the effect of well-defined physical perturbing agents, such as a light pulse or a change in temperature; they can also be more complex and not well-defined phenomena, for instance the outcome of a biological process, an experimental setup, or a mutation. For these situations, SBGN provides the *perturbing agent* glyph. It is an EPN, and represents the amount to perturbing agent applied to a process.

Generalisation

- StatelessEPN (see section 3.3.9)

Attributes

No additional attributes.

Associations

No additional attributes.

Rules and Constraints

- The PerturbingAgent cannot have cardinality > 1. This means there is no multimer glyph.

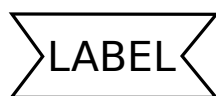
Notation

Figure 3.12: *The Process Description glyph for perturbing agent.*

Glyph: *Perturbing agent*

SBO Term: SBO:0000405 ! perturbing agent

Container: A *perturbing agent* is represented by a modified hexagon having two opposite concave faces, as illustrated in Figure 3.12.

Label: A *perturbing agent* is identified by a label placed in an unbordered box containing a string of characters. The characters can be distributed on several lines to improve readability, although this is not mandatory. The label box must be attached to the center of the *perturbing agent* container. The label may spill outside of the container.

Changes from Previous Version

No changes from pervious version.

3.3.13 StatefulEPN

Stateful entity pools can undergo physical changes, for example chemical modification or conformational change, which we wish to record in a Process Description map. This information is captured via the StateVariable, as can be seen in figure 3.13). Replicated StatefulEPSs are indicated by the *Labelled Clone Marker* decoration. Note that not all glyphs that are descendants of StatefulEPN can be cloned at all.

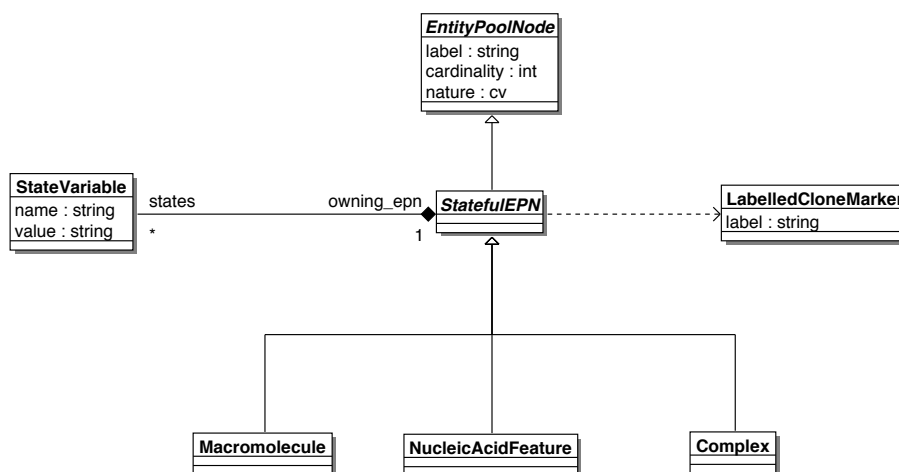


Figure 3.13: UML definition of the stateful entity pool node and its descendant glyphs and its association with state variables.

Generalisation

- EntityPoolNode (see section 3.3.7)

Attributes

No additional attributes.

Associations

states:StateVariable (*) The state variables associated with this EPN.

Rules and Constraints

- Two or more state variables with the same name are permitted.
- State variables with no name set are permitted.
- A LabelledCloneMaker must be used to indicate cloning.
- StatefulEPNs that are identical and so decorated with a LabelledCloneMarker must use the same label to indicate that they are part of the same 'clone'.

Changes from Previous Version

Not defined in the previous version.

3.3.14 Macromolecule

Many biological processes involve *macromolecules*: biochemical substances that are built up from the covalent linking of pseudo-identical units. Examples of macromolecules include proteins, nucleic acids (RNA, DNA), and polysaccharides (glycogen, cellulose, starch, etc.). Attempting to define a separate glyph for all of these different molecules would lead to an explosion of symbols in SBGN, so instead, SBGN Process Description Level 1 defines only one glyph for all macromolecules. The same glyph is to be used for a protein, a nucleic acid, a complex sugar, and so on. The exact nature of a particular macromolecule in a map is then clarified using its label and decorations, as will become clear below. (Future levels of SBGN may subclass the *macromolecule* and introduce different glyphs to differentiate between types of macromolecules). This has two associated glyphs. One where

Generalisation

- StatefulEPN (see section 3.3.13)

Attributes

No additional attributes.

Associations

No additional associations.

Rules and Constraints

No additional rules and constraints.

Notation

There are two glyphs associated with Macromolecule. The first *Macromolecule monomer* is used when cardinality = 1 and the second *Macromolecule multimer* is used when cardinality > 1.

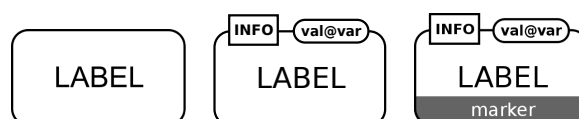


Figure 3.14: The Process Description glyph for macromolecule, shown plain and unadorned on the left, and with with an additional state variable and a unit of information in the right and the cloned form on the right.

Glyph: *Macromolecule monomer*

SBO Term: SBO:0000245 ! macromolecule

Container: A macromolecule is represented by a rectangular container with rounded corners, as illustrated in Figure 3.14.

Label: A *macromolecule* is identified by a label placed in an unbordered box containing a string of characters. The characters can be distributed on several lines to improve readability, although this is not mandatory. The label box must be attached to the center of the container. The label may spill outside of the container.



Figure 3.15: The Process Description glyph for macromolecule multimer, shown plain and unadorned on the left, and with with an additional state variable and a unit of information on the right.

Glyph: Macromolecule multimer

SBO Term: SBO:0000420 ! multimer of macromolecules

Container: A *multimer* is represented by two identical containers shifted horizontally and vertically and stacked one on top of the other. Figure 3.15 illustrates the glyph.

Label: As monomer

Changes from Previous Version

No changes from the previous version.

3.3.15 NucleicAcidFeature

The NucleicAcidFeature represents a fragment of a macromolecule carrying genetic information. A common use for this construct is to represent a gene or transcript. The label of this EPN and its nature are often important for making the purpose clear to the reader of a map.

Generalisation

- StatefulEPN (see section 3.3.13)

Attributes

No additional attributes.

Associations

No additional associations.

Rules and Constraints

No additional rules and constraints.

Notation

The NucleicAcidFeature has two associated glyphs. The first *Nucleic acid feature monomer* is used when cardinality = 1 and the second, *Nucleic acid feature multimer* is used when cardinality > 1.

Glyph: Nucleic acid feature monomer This glyphs represents a monomeric macromolecule.

SBO Term: SBO:0000354 ! informational molecule segment

Container: A *nucleic acid feature* is represented by a rectangular container whose bottom half has rounded corners, as shown in Figure 3.16. This design reminds that we are fundamentally dealing with a unit of information, but this information is carried by a macromolecule.

Label: The identity of a particular *Nucleic acid feature* is established by a label placed in an unordered box containing a string of characters. The characters may be distributed on several lines to improve readability, although this is not mandatory. The label box must be attached to the center of the container. The label may spill outside of the container.

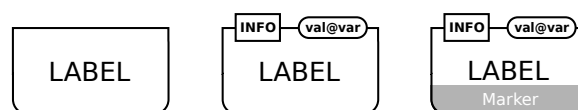


Figure 3.16: The Process Description glyph for nucleic acid feature monomer, shown plain and unadorned on the left and with an additional state variable and a unit of information in the middle and the cloned form on the right.

Glyph: Nucleic acid feature multimer This glyphs represents a multimeric macromolecule.

SBO Term: SBO:0000419 ! multimer of informational molecule segments

Container: A Nucleic acid feature multimer is represented by two identical containers shifted horizontally and vertically and stacked one on top of the other. Figure 3.10 illustrates the glyph.

Label: As monomer glyph.

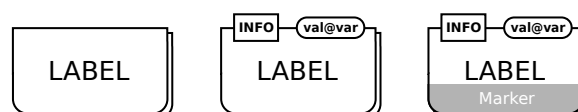


Figure 3.17: The Process Description glyph for nucleic acid feature multimer, shown plain and unadorned on the left and with an additional state variable and a unit of information in the middle and the cloned form on the right.

Changes from Previous Version

No changes from the previous version.

3.3.16 Complex

A Complex represents a biochemical entity composed of other biochemical entities, whether macromolecules, simple chemicals, multimers, or other complexes (figure 3.18). The Complex can describe its composition by the set of Subunits it contains (see figure 3.3.17). This description is entirely optional and is their to assist the user with a visual shorthand about the composition of the complex.

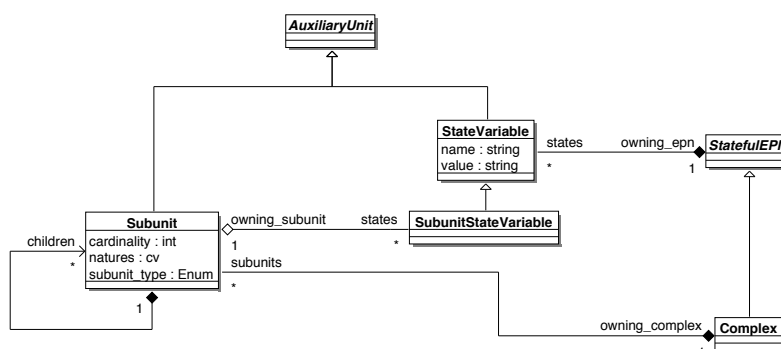


Figure 3.18: The UML definition of the Complex and its associated subunits. In particular this describes organisation of the state variables that belong to both the subunit, but also the complex.

Generalisations

- EntityPoolNode (see section 3.3.7)

Attributes

No additional attributes

Associations

subunits:Subunit (*) The subunits that describe the composition of this complex.

Special Rules and Constraints

- Once a set of subunits are defined then they are always defined.
- The set of subunits in the Complex does not identify it. One or more Complexes that contain the same set of subunits, but have different labels are **not** identical.

Notation

The Complex is represented by two glyphs, the *Complex Monomer* which represents a Complex where the cardinality is one and the *Complex Multimer* where the cardinality is greater than that.

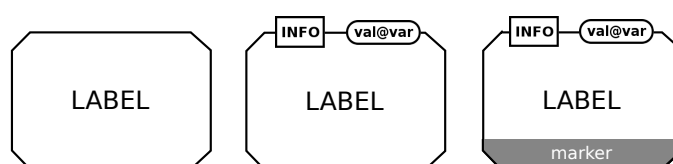


Figure 3.19: The complex glyph.

Complex Monomer

SBO Term: SBO:0000253 ! non-covalent complex

Auxiliary Items: A *complex* can carry state variables (see Section 3.3.26). The state of a complex is defined by the set of the all its state variable and all the state variables of all its components. A *complex* can also carry one or several *units of information* (see Section 3.3.25). A *complex* may carry a *clone marker* (see Section 3.3.37).

Cloning: Labeled Clone Marker

Container: A *complex* possesses its own container box surrounding the juxtaposed container boxes of its components. This container box is a rectangle with cut-corners (an octagonal box with sides of two different lengths). The size of the cut-corners are adjusted so that there is no overlap between the container and the components. The container boxes of the components must not overlap.

Label: The identification of a *named complex* is carried by an unbordered box containing a string of characters. The characters may be distributed on several lines to improve readability, although this is not mandatory. Ideally the label box should be attached to the midway between the border of the complex's container box and the border of the components' container boxes. However, if the Complex contains Subunit glyphs then the label may be positions to optimise the clarity and avoid overlapping.

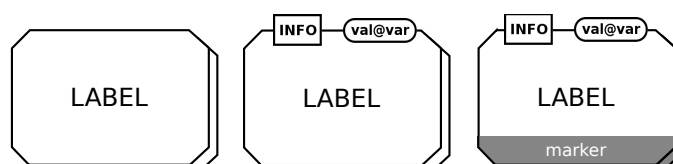


Figure 3.20: The Complex Multimer glyph.

Complex Multimer

SBO Term: SBO:0000418 ! multimer of complexes

Auxiliary Items: As monomer.

Cloning: Labeled Clone Marker

Container: A *Complex Multimer* is represented by two identical *Complex* containers shifted horizontally and vertically and stacked one on top of the other. Figure 3.20 illustrates the glyph.

Label: As monomer

Examples of complex EPNs In this section, we provide examples of Entity Pool Node representations drawn using the SBGN Process Description Level 1 glyphs described above.

Figure 3.21 represents calcium/calmodulin kinase II, with phosphorylation on the sites threonine 286 and 306, as well as catalytic and autoinhibitory domains. Note the use of *units of information* and *state variables*.

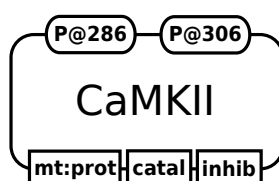


Figure 3.21: An example representation of calcium/calmodulin kinase II.

Figure 3.22 represents the glutamate receptor in the open state, with both phosphorylation and glycosylation. The entity carries two functional domains, the ligand-binding domain and the ion pore, and its chemical nature is precided.

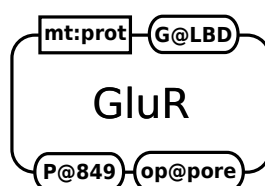


Figure 3.22: An example of a glutamate receptor in the open state.

Layout Rules and Guidelines

- The subunits inside the complex must not overlap.
- The subunits should sit above the clone marker so that they are not obscured by it.
- The label should not be obscured by subunits or obscure them.

Changes from Previous Version

- Clarified that complex must have a label and the label identifies the complex irrespective of its subunit composition.
- The label positioning does not need to be at the centre of the Complex glyph.

3.3.17 Subunit

A complex can optionally be decorated with subunit symbols (Subunit see figure 3.18) to describe the composition of the complex. The symbols available are equivalent to those used by the EPN glyphs including the *complex*. Therefore it is possible to describe complexes within complexes. Subunits may contain labels.

Table 3.1: Mapping between the subunit types and the glyphs used to represent it. These are essentially the EPN glyphs described in this document.

Subunit type	Monomer Glyph	Multimer Glyph
SimpleChemical	<i>Simple Chemical Monomer</i>	<i>Simple Chemical Multimer</i>
UnspecifiedEntity	<i>Unspecified Entity</i>	None
PerturbingAgent	<i>Perturbing Agent</i>	None
Macromolecule	<i>Macromolecule Monomer</i>	<i>Macromolecule Multimer</i>
NucleicAcidFeature	<i>Nucleic Acid Feature Monomer</i>	<i>Nucleic Acid Feature Multimer</i>
Complex	<i>Complex Monomer</i>	<i>Complex Multimer</i>

Generalisation

- EntityPoolNode (see section 3.3.7)

Attributes

cardinality: int (**R**) The number of copies of the subunit.

name: string (**O**) The name of the subunit.

subunit_type: enum (**R**) The type of the subunit. It can have one of the following values that correspond to the equivalent EPN class: SimpleChemical, UnspecifiedEntity, PerturbingAgent, Macromolecule, NucleicAcidFeature, Complex.

Associations

owning_complex:Complex (1) The complex that owns the subunit.

states:SubunitStateVariable (*) The state variables assigned to this subunit.

children:Subunit (*) Subunits that are contained by this subunit.

Rules and Constraints

- Two or more state variables with the same name are permitted.
- State variables with no name set are permitted.
- Subunits can also contain subunits. There is no limit on such nesting. The namespace rules below apply.
- Each state variable actually belongs to the Complex.
- The subunit defines a namespace for its state variables, e.g. subunit “A” assigned a state variable “P@Ser202” and a subunit “B” assigned the same state variable can be distinguished as A:P@Ser202 and B:P@Ser202.
- If the subunit is of type Complex then children can contain one or more Subunit instances.
- If the subunit has a cardinality > 1 then this should be displayed by the AttributeValue (see section 3.3.25).
- If nature contains one or more instances then these must be displayed via a AttributeValue.

Notation

The example in figure Figure 3.23 illustrates the use of subunits in a complex shows an equivalent complex without subunits. This is an import point. For every *Complex* drawn with subunits it will always be possible to drawn an equivalent version that does not use contains subunits.

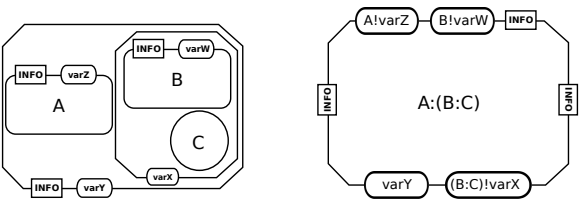


Figure 3.23: Both these complex glyphs are equivalent. The one on the left is described using sub-unit decorators, the one on the right describes the same thing without them.

Changes from Previous Version

3.3.18 ProcessNode

The Process (figure 3.24) represents a process that transforms one or more entity pools into one or more entity pools, that are identical or different. A process may be used to represent or summarise more than one known process. SBGN Process Description Level 1 defines a generic *process* (Section 3.3.23), as well as five more specific ones: the *omitted process* (Section 3.3.23), the *uncertain process* (Section 3.3.23), the *association* (Section 3.3.23), the *dissociation* (Section 3.3.23), and the *phenotype* (Section 3.3.20).

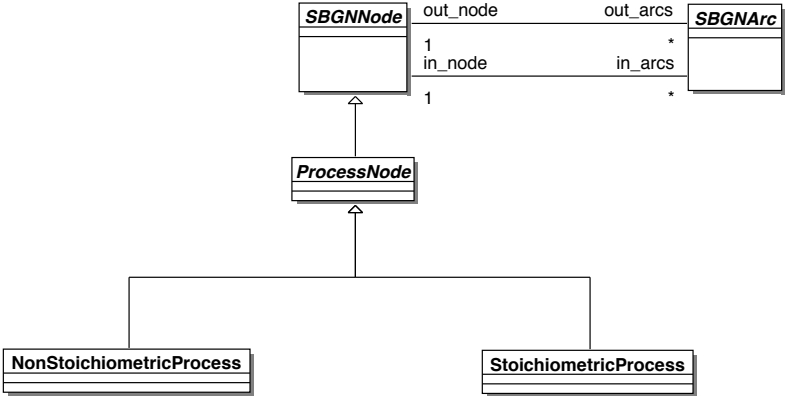


Figure 3.24: The UML definition of the Process and its associated subclasses. Note that the Process extends SBGNNode so all its descendants can potentially be nodes in a directed graph.

Generalisation

- SBGNNode (see section 3.3.5)

Attributes

No additional attributes.

Associations

No additional associations.

Rules and Constraints

No additional rules and constraints.

Changes from Previous Version

- This was not explicitly defined in the previous version, but this version did define a glyph called *Process*. To avoid ambiguity this glyph has now been renamed *Stoichiometric Process* (see section 3.3.23).

- Previous specifications stated that processed could be duplicated when all associated EPNs were cloned. This behaviour has been changed the current status where all processes are unique in a Process Description map.

3.3.19 NonStoichiometricProcess

The NonStoichiometricProcess (figure 3.25) is a type of process. It does not necessarily result in a measurable change of entity pools, nor does it necessarily have a defined start and end point. In many cases the process is not well defined. This may because it is not well understood or because the detail is not important or is being summarised.

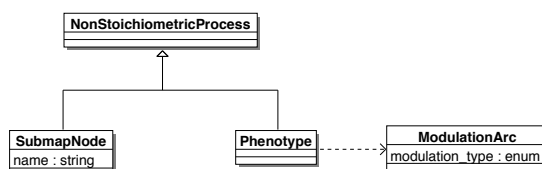


Figure 3.25: The UML definition of the NonStoichiometricProcess and its associated subclasses.

Generalisation

- ProcessNode (see section 3.3.18)

Attributes

No additional attributes.

Associations

No additional associations.

Rules and Constraints

No additional rules and constraints.

Changes from Previous Version

Not defined in the previous version.

3.3.20 Phenotype

A biochemical network can generate phenotypes or affect biological processes. Such processes can take place at different levels and are independent of the biochemical network itself. To represent these processes in a map, SBGN defines the Phenotype (figure 3.25).

Generalisation

- NonStoichiometricProcess (see section 3.3.19)

Attributes

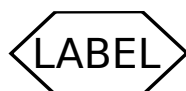
No additional attributes.

Associations

No additional associations.

Rules and Constraints

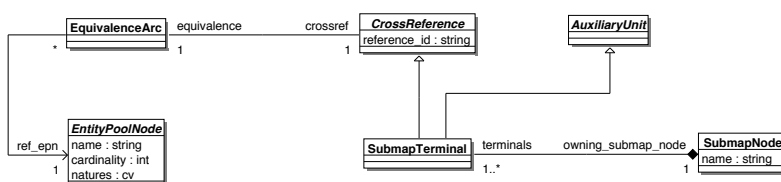
- The Phenotype can only be modulated.
- The Phenotype must be connected to at least one modulating arc.

Notation**Glyph:** *Phenotype***SBO Term:** SBO:0000358 ! phenotype**Container:** A *phenotype* is represented by an elongated hexagon, as illustrated in Figure 3.26.**Label:** A *phenotype* is identified by a label placed in an unbordered box containing a string of characters. The characters can be distributed on several lines to improve readability, although this is not mandatory. The label box must be attached to the center of the *phenotype* container. The label may spill outside of the container.**Figure 3.26:** The Process Description glyph for phenotype.**Changes from Previous Version**

Clarified that the Phenotype cannot be cloned as it is now a subclass of Process, which is always unique.

3.3.21 SubmapNode

The SubmapNode (figure 3.27) is a placeholder for another process and is used when one wishes to hide the detail of this process from the Process Description map, but make it available to the reader as a separate related map. The Submap is not equivalent to an OmittedProcess (section 3.3.23). The Submap allows the detail of section of the Process Description map to be exported to another Process Description map and replaced by the SubmapNode, which acts as a place-holder. This is described in section 3.3.2 and the semantics of submap linking is defined in section 3.6.

**Figure 3.27:** The UML definition of the SubmapNode and its relationship to its submap, tags etc.**Generalisation**

- NonStoichiometricProcess (see section 3.3.19)

Attributes

name: string (**R**) The name of the submap that is being summarised. Note that this name ideally will indicate the function or the processes that are being summarised.

Associations

terminals:SubmapTerminal (1..*) The terminals provide a reference between the EPNs in the Main Map and those in the submap, which are identified by a Tag.

Rules and Constraints

- All instances of SubmapTerminal (see section 3.3.29) held by this class must be unique.

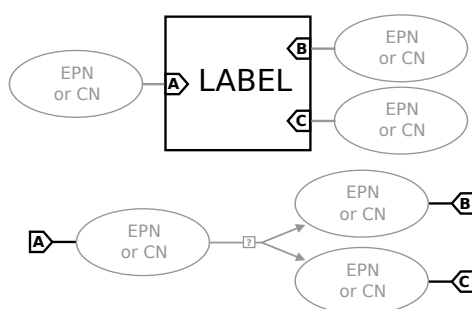
Notation**Glyph: Submap Node****SBO Term:** SBO:0000395 ! encapsulating process**Container:** The *submap* is represented as a square box to remind the viewer that it is fundamentally a process.**Label:** The identification of the *submap* is carried by an unbordered box containing a string of characters. The characters may be distributed on several lines to improve readability, although this is not mandatory. The label box has to be attached to the center of the container box.**Auxiliary Items:** A *submap* carries labeled terminals. When the *submap* is represented folded, those terminals are linked to external EPNs (Section 3.3.7). In the unfolded view, exposing the internal structure of the *submap*, a set of *tags* point to the corresponding internal EPNs Section 3.3.7.

Figure 3.28: The Process Description glyph for submap. (Upper part) folded submap. (Lower part) content of the submap.

Changes from Previous Version

This glyph was called *Submap* in previous version of the Process Description specification. This is confusing when talking about the Submap itself so this glyph is now referred to as the SubmapNode to distinguish it.

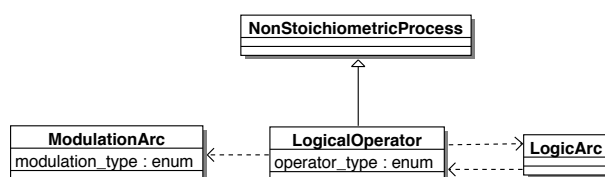
3.3.22 LogicalOperator

Figure 3.29: The UML definition of the LogicalOperator.

The LogicalOperator (figure 3.29) performs a Boolean operation on one or more inputs to give a binary output. The input must be a Boolean value, and are obtained from the LogicArc (see section 3.3.35) connected to the LogicalOperator. The output is a two-value quantity, 0 for False and positive non-zero for True. This is required because the output of the LogicalOperator must be connected to either a LogicArc or a ModulationArc (see section 3.3.34) both of which require their out node to provide a quality. The behaviour of the logical operator for each type of operator_type is shown in the following table:

AND	All inputs must be True for output to be True, otherwise output is false.
OR	At least one input must be True for output to be True. If all inputs are False then output is False.
NOT	Only one input is permitted and the output is the inversion of the input. Therefore True gives False and False gives True.

Generalisation

- NonStoichiometricProcess (see section 3.3.19)

Attributes

operator_type: enum (R) The operator type must be one of the following enumerations: AND, OR, NOT.

Associations

No additional associations.

Rules and Constraints

- in_arc can only contain one or more instances of LogicArc.
- out_arc can only contain one or more instances of LogicArc or ModulationArc.
- if operator_type is AND or OR, then in_arc must contain two or more arcs.
- if operator_type is NOT then in_arc must contain only one arc.
- out_arc can contain only one arc.

Notation

Glyph: And

SBO Term: SBO:0000173 ! and.

Origin: More than one EPN (section 3.3.7) or logical operator (section 3.3.22).

Target: One modulation (section 3.3.34), stimulation (section 3.3.34), catalysis (section 3.3.34), inhibition (section 3.3.34) or necessary stimulation (section 3.3.34) arc.

Node: And is represented by a circle carrying the word “AND”.

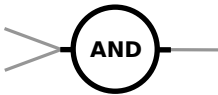


Figure 3.30: The Process Description glyph for and. Only two inputs are represented, but more would be allowed.

Glyph: Or

SBO Term: SBO:0000174 ! or.

Origin: More than one EPN (section 3.3.7) or logical operator (section 3.3.22).

Target: One modulation (section 3.3.34), stimulation (section 3.3.34), catalysis (section 3.3.34), inhibition (section 3.3.34) or necessary stimulation (section 3.3.34) arc.

Node: Or is represented by a circle carrying the word “OR”.

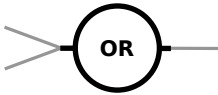


Figure 3.31: The Process Description glyph for or. Only two inputs are represented, but more would be allowed.

Glyph: *Not*

SBO Term: SBO:0000238 ! not.

Origin: One *EPN* (section 3.3.7) or *logical operator* (section 3.3.22).

Target: One modulation (section 3.3.34), stimulation (section 3.3.34), catalysis (section 3.3.34), inhibition (section 3.3.34) or necessary stimulation (section 3.3.34) arc.

Node: *Not* is represented by a circle carrying the word “NOT”.

773

774

775

776

777

778



Figure 3.32: *The Process Description glyph for not.*

Changes from Previous Version

Although the LogicOperator was not explicitly defined in the previous version the semantics and glyphs are unchanged.

779

780

781

3.3.23 StoichiometricProcess

782

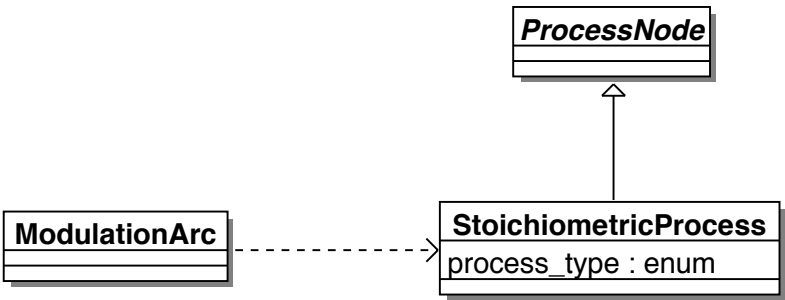


Figure 3.33: *The UML definition of the StiochiometricProcess. The class interacts with subclasses of FluxArc and ModulationArc.*

A process that produces a measurable change in the quantities of entity pools consumed and produced. In Process Description language this is represented by the StoichiometricProcess. This type of process can be though of as having a basal rate of governing the change in EPN quantities, that is changed by other influences described by instances of ModulationArc (see section 3.3.34). StoichiometricProcess can be one of several different types, which indicate the amount that is known about the process or in some cases the nature of the process, for example association and dissociation. The permitted values for process_type are described in the following table:

783

784

785

786

787

788

789

generic	A generic stoichiometric process that transforms a set of entity pools into another set of entity pools.	
omitted	Omitted processes are processes that are known to exist, but are omitted from the map for the sake of clarity or parsimony. A single <i>omitted process</i> can represent any number of actual processes. The <i>omitted process</i> is different from a <i>submap</i> . While a <i>submap</i> references to an explicit content, that is hidden in the main map, the <i>omitted process</i> does not “hide” anything within the context of the map, and cannot be “unfolded”.	
uncertain	Uncertain processes are processes that may not exist. A single <i>uncertain process</i> can represent any number of actual processes.	790
association	The association between one or more <i>EPNs</i> represents the non-covalent binding of the biological objects represented by those <i>EPNs</i> into a larger complex.	
dissociation	The dissociation of an <i>EPN</i> into one or more <i>EPNs</i> represents the rupture of a non-covalent binding between the biological entities represented by those <i>EPNs</i> .	

The *StoichiometricProcess* describes a process that transforms a given set of biochemical entities—macromolecules, simple chemicals or unspecified entities—into another set of biochemical entities. Such a transformation might imply modification of covalent bonds (conversion), modification of the relative position of constituents (conformational process) or movement from one compartment to another (translocation).

A cardinality label may be associated with *consumption* (Section 3.3.32) or *production* (Section 3.3.33) arcs to indicate the stoichiometry of the process. This label becomes a requirement when the exact composition of the number of copies of the inputs or outputs to a reaction are ambiguous in the map.

A process is regarded as reversible if both ‘sides’ of the process are connected to *production* arcs (see section ??). The semantics of *modulation* is the same as for irreversible processes, i.e. the amount of entity in the modulation pool affects the rate of the process.

Generalisation

- *ProcessNode* (see section 3.3.18)

Attributes

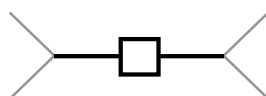
process_type: enum (**R**) This must be one of the following enumerations: generic, omitted, uncertain, association, dissociation.

Associations

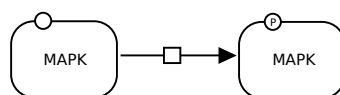
No additional associations.

Rules and Constraints

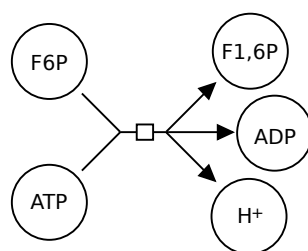
- The *in_arc* must contain one or more *FluxArc* of the same type, i.e., all *FluxArcs* be of class *ConsumptionArc* or all of class *ProductionArc*.
- In addition the *in_arc* may contain zero, one or more instances of *ModulationArc*.
- The *out_arc* must contain one or more instances of *ProductionArc*.
- If *process_type* is association then the *in_arc* must only contain arcs of type *ConsumptionArc* and *out_arc* can only contain one *ProductionArc*.
- If *process_type* is dissociation then the *in_arc* can contain only one *ConsumptionArc* instance.
- If a *StoichiometricProcess* only contains in and out arcs of class *ProductionArc* then it is regarded as reversible.

Notation**Glyph: *Process*****SBO Term:** SBO:0000375 ! process**Origin:** One or several *consumption* arcs (Section 3.3.32) or one or several *production* arcs (Section 3.3.33).**Target:** One or several *production* arcs (Section 3.3.33).**Node:** A process is represented by a square box linked to two connectors, small arcs attached to the centers of opposite sides. The consumption (Section 3.3.32) and production (Section 3.3.33) arcs are linked to the extremities of those connectors. The modulatory arcs (section 3.3.34) point to the other two sides of the box. A *process* connected to *production* arcs on opposite sides is a reversible process.**Figure 3.34:** *The Process Description glyph for process.*

The example in Figure 3.35 illustrates the use of a *process* node to represent the phosphorylation of a protein in a Process Description.

**Figure 3.35:** *Phosphorylation of the protein MAP kinase.*

The example in Figure 3.36 illustrates the use of a *process* node to represent a reaction between two reactants that generates three products.

**Figure 3.36:** *Reaction between ATP and fructose-6-phosphate to produce fructose-1,6-biphosphate, ADP and a proton.*

The example in Figure 3.37 illustrates the use of a *process* node to represent a translocation. The large round-cornered rectangle represents a compartment border (see Section 3.3.24).

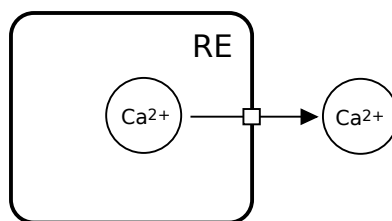


Figure 3.37: Translocation of calcium ion out of the endoplasmic reticulum. Note that the process does not have to be located on the boundary of the compartment. A process is not attached to any compartment.

The example in Figure 3.38 illustrates the use of a *process* node to represent the reversible opening and closing of an ionic channel in a Process Description.

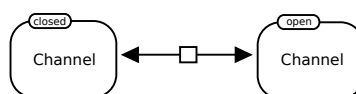


Figure 3.38: Reversible opening and closing of an ionic channel.

When such a reversible process is asymmetrically modulated, it must be represented by two different processes in a Process Description. Figure 3.39 illustrates the use of two *process* nodes to represent the reversible activation of a G-protein coupled receptor. In the absence of any effector, an equilibrium exists between the inactive and active forms. The agonist stabilises the active form, while the inverse agonist stabilises the inactive form.

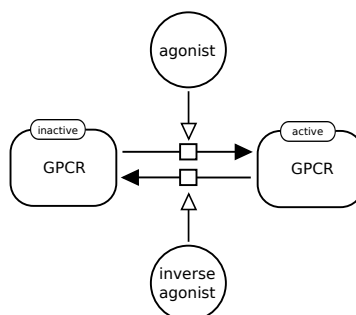


Figure 3.39: The reversible activation of a G-protein coupled receptor.

The example in Figure 3.40 presents the conversion of two galactoses into a lactose. Galactoses are represented by only one *simple chemical*, the cardinality being carried by the *consumption* arc.

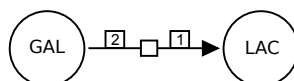


Figure 3.40: Conversion of two galactoses into a lactose.

Glyph: *Omitted process*

SBO Term: SBO:0000397 - omitted process.

Origin: One or several *consumption* arcs (Section 3.3.32) or one or several *production* arcs (Section 3.3.33).

Target: One or several *production* arcs (Section 3.3.33). 849

Node: An *omitted process* is represented by a *process* in which the square box contains a two parallel 850
slanted lines oriented northwest-to-southeast and separated by an empty space. 851

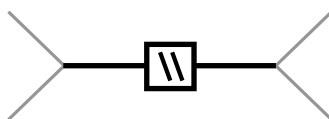


Figure 3.41: The Process Description glyph for omitted process.

Glyph: *Uncertain process* 852

SBO Term: SBO:0000396 ! uncertain process. 853

Origin: One or several *consumption* arcs (Section 3.3.32) or one or several *production* arcs (Sec- 854
tion 3.3.33). 855

Target: One or several *production* arcs (Section 3.3.33). 856

Node: An *uncertain process* is represented by a *process* which square box contains a question mark. 857

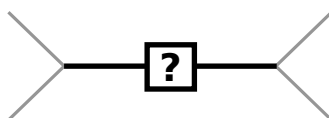


Figure 3.42: The Process Description glyph for an uncertain process.

Glyph: *Association* 858

SBO Term: SBO:0000177 ! non-covalent binding. 859

Origin: One or more *consumption* arcs (Section 3.3.32). 860

Target: One *production* arc (Section 3.3.33). 861

Node: An *association* between several entities is represented by a filled disc linked to two connectors, 862
small arcs attached on point separated by 180 degrees. The consumption (Section 3.3.32) and 863
production (Section 3.3.33) arcs are linked to the extremities of those connectors. 864

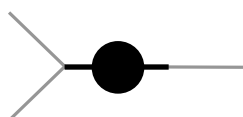


Figure 3.43: The Process Description glyph for association.

The example in Figure 3.44 illustrates the association of cyclin and CDC2 kinase into the Matura- 865
tion Promoting Factor. 866

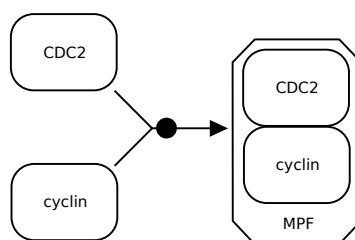


Figure 3.44: Association of cyclin and CDC2 kinase into the Maturation Promoting Factor.

Figure 3.45 gives an example illustrating the association of a pentameric macromolecule (a nicotinic acetylcholine receptor) with a simple chemical (the local anesthetic chlorpromazin) in an unnamed complex.

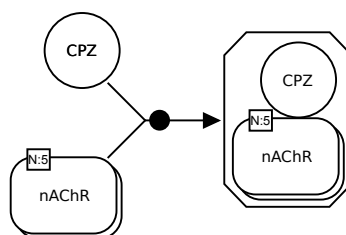


Figure 3.45: The association of a pentameric macromolecule with a simple chemical in an unnamed complex.

An association does not necessarily result in the formation of a *complex*; it can also produce a *multimer*, or a *macromolecule* (although the latter case is semantically borderline). 3.46 gives an example of this, using the formation of hemoglobin.

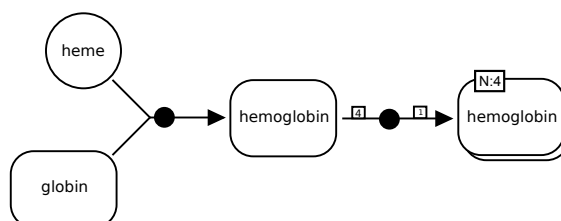


Figure 3.46: Formation of hemoglobin.

Glyph: *Dissociation*

SBO Term: SBO:0000180 ! dissociation.

Origin: One *consumption* arc (Section 3.3.32).

Target: One or more *production* arc (Section 3.3.33).

Note: A *dissociation* between several entities is represented by two concentric circles. A simple empty disc could be, in some cases, confused with the *catalysis* (section Section 3.3.34). Moreover, the existence of two circles reminds the dissociation, by contrast with the filled disc of the *association* (Section 3.3.23).

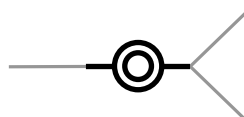


Figure 3.47: The Process Description glyph for dissociation.

The example in Figure 3.48 illustrates the dissociation of the small and large ribosomal subunits from a messenger RNA.

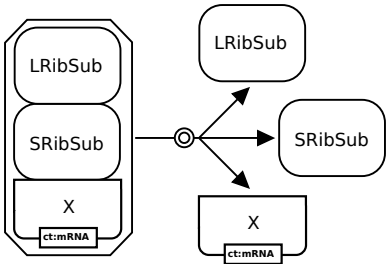


Figure 3.48: Dissociation of the small and large ribosomal subunits from a messenger RNA.

Changes from Previous Version

Although the NonStoichiometricProcess was not explicitly defined in the previous version the semantics and glyphs are unchanged.

3.3.24 Compartment

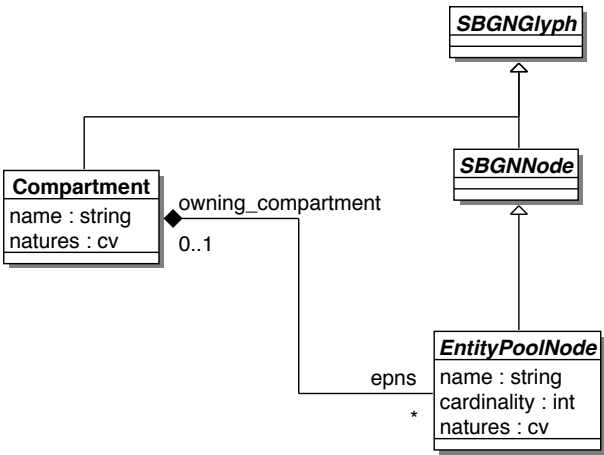


Figure 3.49: The UML definition of the Compartment showing how it containment of EntityPoolNode.

The Compartment is a logical or physical structure that contains entity pool nodes. An EntityPoolNode (see section 3.3.7) can only belong to one compartment. Therefore, the “same” biochemical species located in two different compartments are in fact two different pools.

Generalisation

- SBGNGlyph (see section 3.3.3)

Attributes

name: string (R) The name of the compartment.
natures: cv(*) A set of controlled vocabularies that describes a characteristic of the compartment.
Zero, one or more values may be set, but each one must belong to a different controlled vocabulary.

Associations

epns:EntityPoolNode (*) The EntityPoolNodes contained by this compartment.

Rules and Constraints

- name must not be used by another instance of Container contained by the same instance of Map.

- epns must contain a unique set of EntityPoolNodes. See section 3.5 for the definition of Entity-PoolNode uniqueness.

Notation

Glyph: *Compartment*

SBO Term: SBO:0000290 ! physical compartment

Container: A compartment is represented by a surface enclosed in a continuous border or located between continuous borders. These borders should be noticeably thicker than the borders of the EPNs. A compartment can take **any** geometry. A compartment must always be entirely enclosed.

Label: The identification of the compartment is carried by an unbordered box containing a string of characters. The characters can be distributed on several lines to improve readability, although this is not mandatory. The label box can be attached anywhere in the container box. Note that the label can spill-over from the container box.

Auxiliary Items: A *compartment* can carry a certain number of *units of information*, that will add information for instance about the physical environment, such as pH, temperature or voltage, see Section 3.3.25. The center of the bounding box of a *unit of information* is located on the mid-line of the border of the compartment.

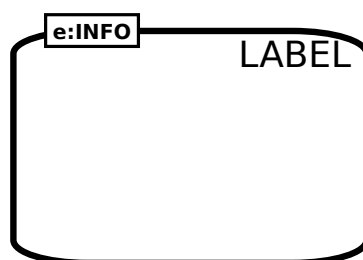


Figure 3.50: *The Process Description glyph for compartment.*

To allow more aesthetically pleasing and understandable maps, compartments are allowed to overlap each other visually, but it must be kept in mind that this does not mean the top compartment contains part of the bottom compartment. Figure 3.51 shows two semantically equivalent placement of compartments:

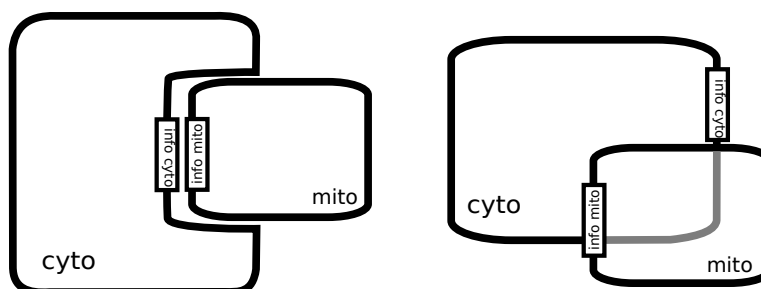


Figure 3.51: *Overlapped compartments are permitted, but the overlap does not imply containment.*

Overlapped (hidden) part of the compartment should not contain any object which could be covered by an overlapping compartment. Figure 3.52 illustrates the problem using an incorrect map.

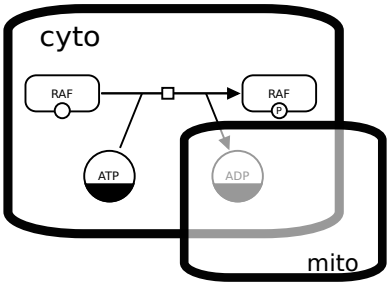


Figure 3.52: Example of an *incorrect* map. Overlapped compartments must not obscure other objects.

Changes from Previous Version

No changes from previous the version.

3.3.25 AttributeValue

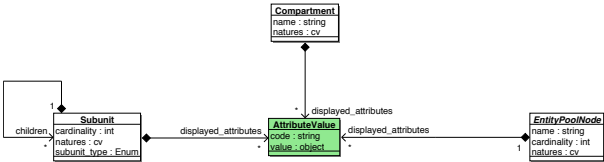


Figure 3.53: The UML definition of the AttributeValue and its usage by other classes.

The AttributeValue is used to present the values of certain attributes held by other SBGN elements. It is typically contained and owned by the class containing the attribute (or its descendants). It contains two values, one is a code to indicate the attribute that is defined and the other is the value itself. The code and the presentation format of the value are defined by the SBGN element that contains the AttributeValue, currently Compartment (see section 3.3.24), EntityPoolNode (see section 3.3.7), and Subunit (see section 3.3.17).

Generalisation

- AuxiliaryUnit (see section 3.3.4)

Attributes

code: string (R) The code indicating the attribute that is being presented.
value: object (R) The value of the attribute. The format of the value is determined by the class holding the attribute.

Associations

No additional associations.

Rules and Constraints

No additional rules and constraints.

Notation

For historical reasons the AttributeValue is represented graphically by the glyph *Unit of Information*.

Glyph: *Unit of information* When representing biological entities, it is often necessary to convey some abstract information about the entity's function that cannot (or does not need to) be easily related to its structure. The *unit of information* is a decoration that can be used in this situation to add information to a glyph. Some example uses include: characterizing a logical part of an entity such as a functional domain (a binding domain, a catalytic site, a promoter, etc.), or the information encoded in the entity (an exon, an open reading frame, etc.). A *unit of information* can also convey information about the physical environment, or the specific type of biological entity it is decorating.

SBO Term: Not applicable.

Container: A unit of information is represented by a rectangle. The long side of the rectangle should be oriented parallel to the border of the *EPN* being annotated by the *unit of information*. The center of the bounding box of a *state of information* should be located on the mid-line of the border of the *EPN*.

Label: A *unit of information* is identified by a label placed in an unbordered box containing a string of characters. The characters can be distributed on several lines to improve readability, although this is not mandatory. The label box must be attached to the center of the container. The label may spill outside of the container.

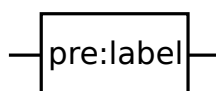


Figure 3.54: The Process Description glyph for unit of information.

Changes from Previous Version

There was no definition of the *AttributeValue* in the previous version of this specification. However, the *Unit of Information* did exist although its semantics have been changed. It no longer can hold arbitrary annotation but must display an attribute value and observe the constraints set out by the definition of the class owning the attribute.

Since the use of the *Unit of Information* has been deprecated, it is recommended that *Annotation* (see section 3.3.27) and the *Annotation* glyph is used instead.

3.3.26 StateVariable

Many biological entities such as molecules can exist in different *states*, meaning different physical or informational configurations. These states can arise for a variety of reasons. For example, macromolecules can be subject to post-synthesis modifications, wherein residues of the macromolecules (amino acids, nucleosides, or glucid residues) are modified through covalent linkage to other chemicals. Other examples of states are alternative conformations as in the closed/open/desensitized conformations of a transmembrane channel, and the active/inactive forms of an enzyme.

SBGN provides a means of associating one or more *StateVariable* with an entity (figure 3.55); each such variable can be used to represent a dimension along which the state of the overall entity can vary. When an entity can exist in different states, the state of the whole entity (i.e., the SBGN object) can be described by the set of *StateVariable* instances it contains. Note that this is true for the *Complex* (see section 3.3.16) where any state variables are associated visually with subunits (c.f. *Subunit* (see section 3.3.17)), but actually belong to the complex (figure 3.55).

Generalisation

- *AuxiliaryUnit* (see section 3.3.4)

Attributes

name: string (**O**) The name of the state variable. This value is optional.

value: string (**R**) The value of the state variable. This is optional, but cannot be an empty string.

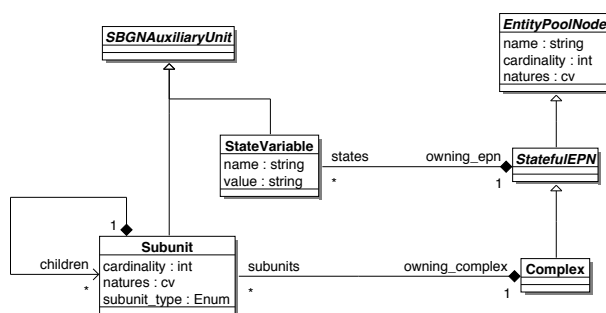


Figure 3.55: The UML definition of the StateVariable showing its relationship to StatefulEPN, Complex and Subunit.

Associations

owning_epn:StatefulEPN (1) The stateful EPN that owns the state variable.

Rules and Constraints

- The symbols in the

Notation

Glyph: State variable

SBO Term: Not applicable.

Container: A *state variable* is represented by a “stadium” container, that is two hemicircles of same radius joined by parallel segments, as shown in Figure 3.56. The parallel segment axis should be tangent to the border of the glyph of the *EPN* being modified by the *state variable*. The center of the bounding box of a *state variable* should be located on the mid-line of the border of the *EPN*. In previous versions of this specification the *state variable* was represented by an ellipse. This symbols is now **deprecated** in favour of the stadium symbol described above. New Process Description maps should not use the ellipse symbol.

Label: An unbordered box containing a string indicating the contents of the StateVariable. The style of labeling of *State Variables* encouraged by SBGN Process Description Level 1 is to combine a prefix representing the value of the variable with a suffix representing the variable’s name. Prefix and suffix should be separated by the symbol ‘@’, X@Y thus meaning *value X AT variable Y*. If name is undefined then only the value should be displayed and the ‘@’ character omitted. If both the name and value are undefined then the label should be empty (i.e., an empty string). The label of a *state variable* should, if possible, be displayed within the boundary of the glyph. In earlier versions of the SBGN specification it was permitted to separate the name and value into two unlabelled boxes and display the name box outside the *state variable* glyph. This is now **deprecated** and new Process Description maps should not use this notation.

Auxiliary Items: A *state variable* does not carry any auxiliary items.

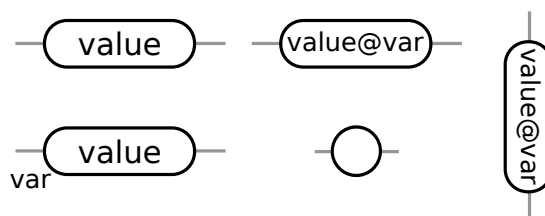


Figure 3.56: Examples of the Process Description glyph for state variable.

A *state variable* does not necessarily have to be Boolean-valued. For example, an ion channel can possess several conductance states; a receptor can be inactive, active and desensitized; and so on.

As another example, a *state variable* “ubiquitin” could also carry numerical values corresponding to the number of ubiquitin molecules present in the tail. However, in all cases, a *state variable* on an EPN can only take *one* defined value. Further, an EPN’s *state variable* should always be displayed and always set to a value. An “empty” *state variable* is a *state variable* that is set to the value “unset”, it is not a *state variable* with no value. Note that the value “unset” is *not* synonymous to “any value” or “unknown value”.

Changes from Previous Version

The StateVariable class was not explicitly defined in previous versions of the specification, however the *state variable* was. Some aspects of its notation have been deprecated and these are detailed above (section 3.3.26).

3.3.27 Annotation

In SBGN Process Description Level 1 there are cases where the language does not capture everything the author wishes to convey. This may be additional experimental detail or descriptions of mechanisms that cannot be described full by the Process Description language. In this case the language provides the Annotation. This contains text and is associated with a particular glyph in a map. Importantly, it is purely “decoration” and does alter the meaning the map.

Generalisation

- AuxiliaryUnit (see section 3.3.4)

Attributes

annotation_text: string (**R**) The text of the annotation. The text is mandatory and cannot be empty or just spaces.

Associations

annotated_glyph:SBGNGlyph (1) The instance of SBGNGlyph that is being annotated¹.

Rules and Constraints

No additional rules and constraints.

Notation

Glyph: *Annotation*

SBO Term: SBO:NEW

Container: An *annotation* is represented by a rectangular container with a folded corner, as illustrated in Figure 3.57. This container is linked to the annotated element via a callout (see figure 3.58. The callout should overlap with the object it is annotating.

Label: An *annotation* contains information placed in an unbordered box containing a string of characters. The characters can be distributed on several lines to improve readability, although this is not mandatory. The label box must be attached to the center of the container. The label may spill outside of the container.

Auxiliary Items: An *annotation* does not carry any auxiliary unit.

¹Note that as a result of this association only glyphs and **not** auxiliary items may be annotated by instances of Annotation

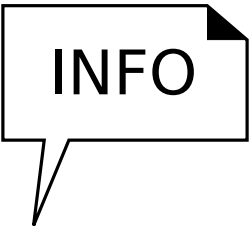


Figure 3.57: The Process Description glyph for annotation.

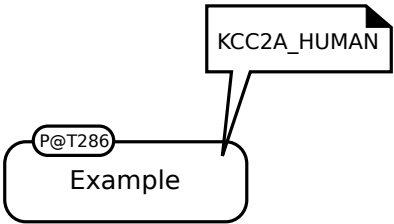


Figure 3.58: Example of annotations adding information to the description of the transphosphorylation of CaMKII. Note that three different types of links are used between annotation nodes and annotated elements. However, it is recommended to use a consistent scheme within a map.

Changes from Previous Version

1049

This is a new language element and an not previous versions of the Process Description language.

1050

3.3.28 CrossReference

1051

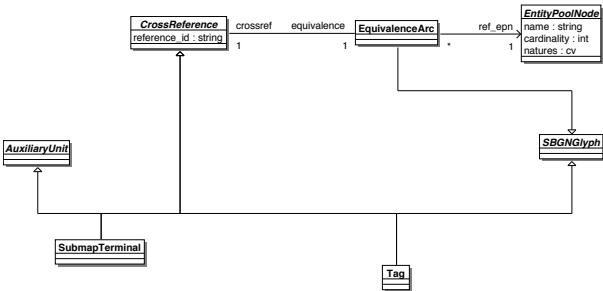


Figure 3.59: The UML definition of the Crossreference showing its subclasses Tag and SubmapTerminal and its association with other elements in the Process Description language.

CrossReference handles links or relationships between elements of a map and su-map. At present there is only one reference glyph, tag, which can be used in a map referred to by a submap (Section 3.3.21) or as an auxiliary unit on the submap. The clone marker can also provide additional reference mechanisms and is discussed below (Section 3.3.37).

1052
1053
1054
1055

Generalisation

1056

None

1057

Attributes

1058

reference_id: string (R) a string that identifies the cross-reference. The string cannot start and end in white space and cannot be empty.

1059
1060

Associations1061

equivalence:EquivalenceArc (1) The equivalence arc that links this class to the referenced element.1062

Rules and Constraints1063

- Two or more instances of CrossReference with the same reference_id value are pointing to the same element.10641065
 - The above rules applies within a Process Description map’s namespace (see section 3.6).1066

Changes from Previous Version1067

Not defined in the previous version.1068

3.3.29 SubmapTerminal1069

A SubmapTerminal (figure 3.27) is a named reference that is part of a SubmapNode (see section 3.3.21).1070
It provides the reference that is the link to a tag in the submap that the SubmapNode refers to.1071

Generalisation1072

- AuxiliaryUnit (see section 3.3.4)1073
 - CrossReference (see section 3.3.28)1074

Attributes1075

No additional attributes.1076

Associations1077

No additional associations.1078

Rules and Constraints1079

No additional rules and constraints.1080

Notation1081

Glyph: Submap Terminal1082

SBO Term: Not applicable.1083

Container: A tag is represented by a rectangle fused to an empty arrowhead. The flat edge opposite the arrowhead should be aligned to the edge of the Submap glyph and the connecting should connect to the middle of this face (see figure 3.60).108410851086

Label: A tag is identified by a label placed in an unbordered box containing a string of characters.1087
The characters can be distributed on several lines to improve readability, although this is not mandatory. The label box must be attached to the center of the container. The label may spill outside of the container.108810891090

Auxiliary Items: A tag does not carry any auxiliary items.1091

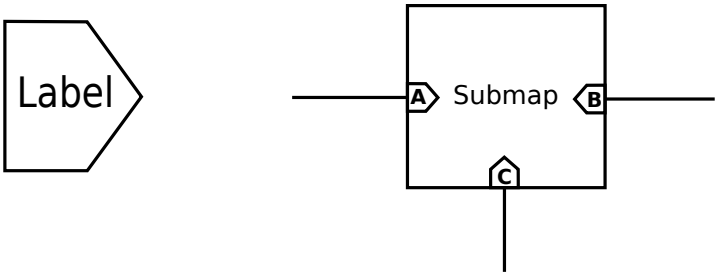


Figure 3.60: The Process Description glyph for Submap Terminal. This shows the basic glyph and its correct usage within a Submap glyph.

Changes from Previous Version

Clarified that the tag does not link a Compartment, but only instances of EntityPoolNode.

3.3.30 Tag

A Tag is a named handle, or reference, to another EntityPoolNode. *Tags* are used to identify those elements in *submaps* (Section 3.3.21).

Generalisation

- SBGNGlyph (see section 3.3.3)
- CrossReference (see section 3.3.28)

Attributes

No additional attributes.

Associations

No additional associations.

Rules and Constraints

- All values of `reference_id` must be unique within an instance of Map.

Notation

Glyph: *Tag*

SBO Term: Not applicable.

Container: A *tag* is represented by a rectangle fused to an empty arrowhead, as illustrated in Figure 3.61. The symbol should be linked to one and only one edge (i.e., it should reference only one EPN or compartment).

Label: A *tag* is identified by a label placed in an unbordered box containing a string of characters. The characters can be distributed on several lines to improve readability, although this is not mandatory. The label box must be attached to the center of the container. The label may spill outside of the container.

Auxiliary Items: A *tag* does not carry any auxiliary items.



Figure 3.61: *The Process Description glyph for tag.*

Changes from Previous Version

Clarified that the tag does not link a Compartment, but only instances of EntityPoolNode.

3.3.31 FluxArc

The FluxArc permits a quantity of entities to flow through the arc and in doing so connects a stoichiometric process (StoichiometricProcess (see section 3.3.23)) and an EPN (EntityPoolNode (see section 3.3.7)). The arc has a stoichiometry which affects the flow through the arc. So for example a stoichiometry of $2n$ permits twice as many entities through as one of n .

Generalisation

- SBGNArc (see section 3.3.6)

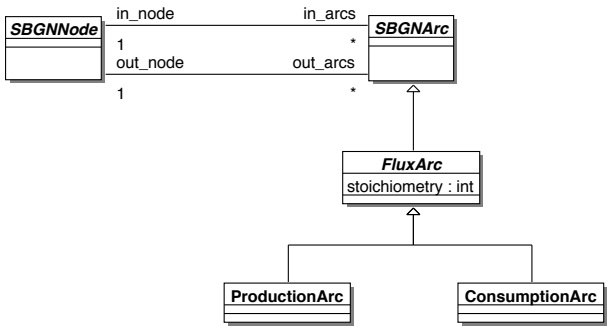


Figure 3.62: The UML definition of the FluxArc and its subclasses.

Attributes	1126
stoichiometry: int (R) The stoichiometry of this FluxArc (see section 3.3.31). This must be a non-zero positive integer.	1127
No additional attributes.	1128
	1129
Associations	1130
No additional associations.	1131
	1132
Rules and Constraints	1132
• If the stoichiometry > 1 then the stoichiometry must be displayed. Figure 3.63 illustrates the use of consumption/production arc stoichiometry labels to represent the stoichiometry of a process.	1133
	1134
	1135
• All process nodes (with the exception of <i>phenotype</i>) must have a LHS and RHS.	1136
• The EntityPoolNodes that make up the LHS of the process should be consistent with the RHS, i.e. the process should constitute a balanced biochemical reaction.	1137
	1138
• Once the stoichiometry of a FluxArc is displayed in a map then all other FluxArcs should display their stoichiometry label.	1139
	1140
• If the stoichiometry is undefined or unknown this should be indicated by the use of a question mark (“?”).	1141
	1142
• If more than one set of stoichiometries can be applied to the flux arcs of the process then the stoichiometry of the flux arcs must be displayed.	1143
	1144
Notation	1145
Stoichiometry Label The stoichiometry label is part of the <i>consumption arc</i> and <i>production arc</i> glyphs see below (sections 3.3.32 and 3.3.33). However, as their use is common to all subclasses of FluxArc their presentation is described here.	1146
	1147
	1148

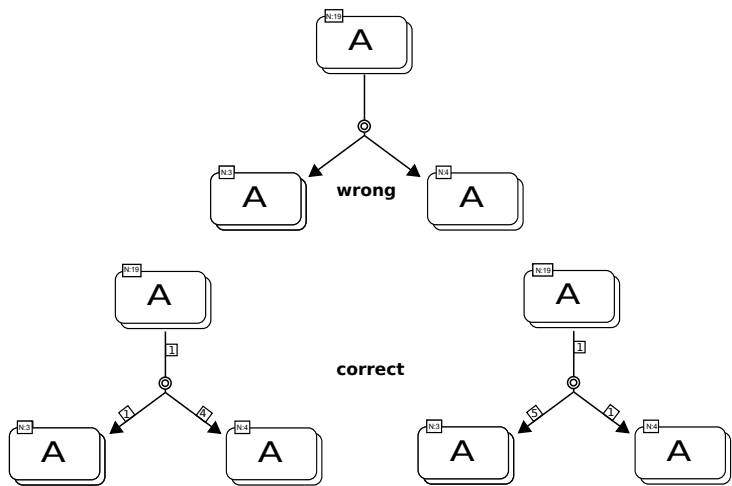


Figure 3.63: The figure illustrates why for the stoichiometry label is required to clarify potentially ambiguous stoichiometry. In the top example there is more than one possible solution, which can only be made clear using the stoichiometry labels in the bottom examples.

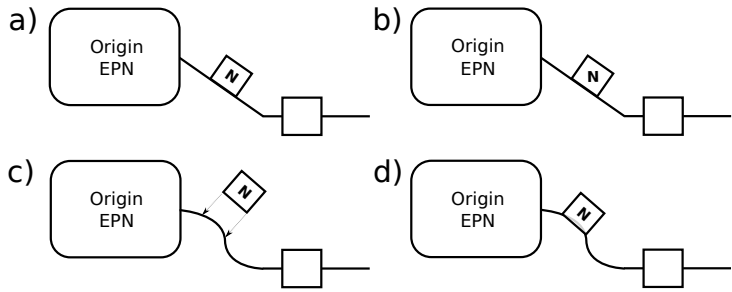


Figure 3.64: Examples of stoichiometry label layout. In figure (a) the label is aligned with the stoichiometry box, while in (b) the label is aligned with the orientation of the map: these are both simple cases where the arc is a straight line. In cases where the arc is curved, the corners at the base of the label are anchored to point on the arc (c) and the label is drawn over the arc (d). Note that in (d) the covered part of the arc is shown for clarity, but normally the box is opaque and so the arc is not visible.

The label is a node that must be drawn above the flux arc. This node is attached to the arc where it intersects the arc with its bottom corners (see figure 3.64.).

SBO Term: None

Container: A rectangle with a draw edge.

Label: A number that should remain within the container and be of a normal font, i.e., not bold or italic.

Changes from Previous Version

Not defined explicitly in the previous version.

3.3.32 ConsumptionArc

Generalisation

- FluxArc (see section 3.3.31)

Attributes

No additional attributes.

Associations

No additional associations.

Rules and Constraints

- The `in_node` must be an instance of `EntityPoolNode` (see section 3.3.7).
- The `out_node` must be an instance of `StoichiometricProcess` (see section 3.3.23).

Notation

Glyph: *Consumption* *Consumption* is the arc used to represent the fact that an entity pool is consumed by a process, but is not produced by the process.

SBO Term: SBO:0000394 ! consumption.

Origin: Any *EPN* (Section 3.3.7).

Target: Any *process node* (Section 3.3.18).

End point: No particular symbol is used to represent a consumption.

A cardinality label may be associated with *consumption* (Section 3.3.32) or *production* (Section 3.3.33) arcs, indicating the stoichiometry of a process. This label is a number enclosed in a rectangle with one of the long sides adjacent to the consumption arc. The cardinality is required to eliminate ambiguity when the exact composition, or the number of copies, of the inputs or outputs to a reaction are ambiguous from the map. An example is a multimer of 6 subunits dissociating into 2 monomers and 2 dimers. Without stoichiometry labels another result, such as 4 monomers and 1 dimer could be inferred. Once assigned to one arc connecting to a process node, cardinality should be represented on all *consumption* and *production* arcs connected to that process node to avoid misinterpretation.

Omitted cardinality on one edge only should not be treated as cardinality of 1, but as an unspecified cardinality. In most cases, the exact value may be derived from the context, but unless cardinality is explicitly shown, it should be considered as unspecified. In the case where the stoichiometry of some part of the process is not known, or undefined, a question mark (?) should be used within the cardinality label of the corresponding arcs.



Figure 3.65: *The Process Description glyph for consumption.*

Changes from Previous Version

No changes from previous version.

3.3.33 ProductionArc**Generalisation**

- `FluxArc` (see section 3.3.31)

Attributes

No additional attributes.

Associations

No additional associations.

Rules and Constraints

- The `in_node` must be an instance of `StoichiometricProcess` (see section 3.3.23).
- The `out_node` must be an instance of `EntityPoolNode` (see section 3.3.7).

Notation

Glyph: *Production* *Production* is the arc used to represent the fact that an entity pool is produced by a process. In the case of a reversible process, the *production* arc also acts as a *consumption* arc.

SBO Term: SBO:0000393 ! production.

Origin: Any *process node* (Section 3.3.18).

Target: Any *EPN* (Section 3.3.7).

End point: The target extremity of a *production* carries a filled arrowhead.

A cardinality label can be associated with a *production* arc indicating the stoichiometry of a process.

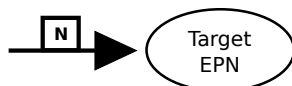


Figure 3.66: The Process Description glyph for production.

Changes from Previous Version

No changes from previous versions. **Do we need to put a Reversible arc to handle the reversible side of the consumption arc?**

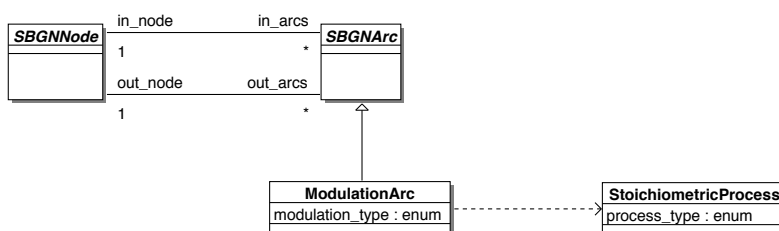
3.3.34 ModulationArc

Figure 3.67: The UML definition of the ModulationArc. The class interacts with subclasses of StoichiometricProcess.

The ModulationArc (figure 3.67) affects the flux of a process represented by the target process. Such a modulation can affect the process **positively or negatively**, or even both ways depending on the conditions, for instance the concentration of the intervening participants. The permitted values for process_type are described in the following table:

modulation	A general modulation where the exact nature of the modulation is not specified or not known. Modulation can be used when one does not know the precise direction of the effect.	
stimulation	A stimulation affects positively the flux of a process represented by the target process. This stimulation can be for instance a catalysis or a positive allosteric regulation. Note that <i>catalysis</i> exists independently in SBGN, see Section 3.3.34.	
catalysis	A particular case of stimulation, where the effector affects positively the flux of a process represented by the target process. The positive effect on the process is due to the lowering of the activation energy of a reaction.	1216
inhibition	An inhibition negatively affects the flux of a process represented by the target process. This inhibition can be for instance a competitive inhibition or an allosteric inhibition.	
necessary_stim	A necessary stimulation, is one that is necessary for a process to take place. A process modulated by a necessary stimulation can only occur when this necessary stimulation is active.	

As discussed in Chapter 2, it is implied, but not defined explicitly that the process has a rate at which it converts its LHS EPNs to its RHS EPNs (and vice-versa in the case of a reversible process). This concept is important in understanding how the Process Description language describes process modulation.

1. A *process* with no modulations has an underlying “basal rate” which describes the rate at which it converts inputs to outputs.
2. A *modulation* changes the basal rate in an unspecified fashion.
3. A *stimulation* is a modulation that increases the basal rate.
4. An *inhibition* is a modulation that decreases the basal rate.
5. The above types of modulation, when assigned to the same process, are combined and have a multiplicative effect on the basal rate of the process.
6. Modulators that do not interact with each other in the above manner, should be drawn as modulating different process nodes. Their effect is therefore additive.

Generalisation

- EntityPoolNode (see section 3.3.7)

Attributes

No additional attributes.

Associations

states:StateVariable (*) The state variables associated with this EPN.

Rules and Constraints

- At most one *necessary stimulation* can be assigned to a process node. Two *necessary stimulations* would imply an implicit AND or OR operator. For clarity only one *necessary stimulation* can be assigned to a *process*, and such combinations must be explicitly expressed using *logical operators*.

- At most one *catalysis* can be assigned to a *process*. Modulation by a catalysis arc implies that the exact biochemical mechanism underlying the process is known. In this context two *catalysis* cannot be assigned to the same process node as they represent independent reactions. Other EPNs can be assigned to the same process as a catalysis, such as modulators, stimulators, and inhibitors, and will have a multiplicative modulation on the reaction rate defined by the catalysis.

Notation

The ModulationArc is represented by a number of glyphs depending on its modulation_type. The table below defines what glyph is used for each type.

Type	Glyph
modulation	<i>Modulation</i>
stimulation	<i>Stimulation</i>
inhibition	<i>Inhibition</i>
necessary_stim	<i>Necessary Stimulation</i>

Glyph: *Modulation*

SBO Term: SBO:0000168 ! control.

Origin: Any *EPN* (Section 3.3.7) or any *logical operator* (Section 3.3.22).

Target: Any *process node* (Section 3.3.18).

End point: The target extremity of a *modulation* carries an empty diamond.



Figure 3.68: The Process Description glyph for modulation.

Figure 3.69 represents the effect of nicotine on the process between closed and open states of a nicotinic acetylcholine receptor. High concentrations of nicotine open the receptor while low concentrations can desensitize it without opening.

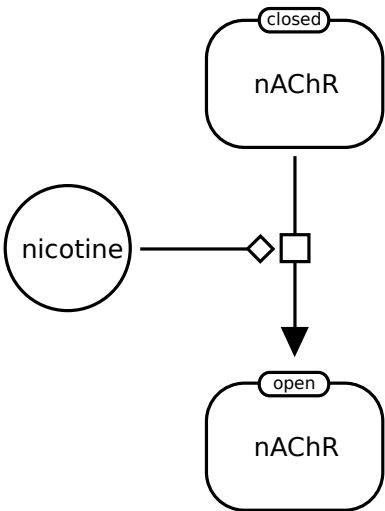
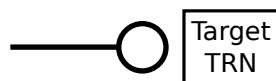
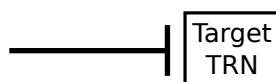


Figure 3.69: Modulation of nicotinic receptor opening by nicotine.

Glyph: Stimulation**SBO Term:** SBO:0000170 ! stimulation.**Origin:** Any *EPN* (Section 3.3.7) or any *logical operator* (Section 3.3.22).**Target:** Any *process node* (Section 3.3.18).**End point:** The target extremity of a *stimulation* carries an empty arrowhead.**Figure 3.70:** The Process Description glyph for stimulation.**Glyph: Catalysis****SBO Term:** SBO:0000172 ! catalysis.**Origin:** Any *EPN* (Section 3.3.7) or any *logical operator* (Section 3.3.22).**Target:** Any *process node* (Section 3.3.18).**Node:** The target extremity of a *catalysis* carries an empty circle.**Figure 3.71:** The Process Description glyph for catalysis.**Glyph: Inhibition****SBO Term:** SBO:0000169 ! inhibition.**Origin:** Any *EPN* (Section 3.3.7) or any *logical operator* (Section 3.3.22).**Target:** Any *process node* (Section 3.3.18).**Node:** The target extremity of an *inhibition* carries a bar perpendicular to the arc.**Figure 3.72:** The Process Description glyph for inhibition.**Glyph: Necessary stimulation****SBO Term:** SBO:0000171 ! necessary stimulation.**Origin:** Any *EPN* (Section 3.3.7) or any *logical operator* (Section 3.3.22).**Target:** Any *process node* (Section 3.3.18).**Node:** The target extremity of a *necessary stimulation* carries an open arrow (to remind that it is a *stimulation*) coming after a larger vertical bar.**Figure 3.73:** The Process Description glyph for Necessary Stimulation.

Examples The example in Figure 3.74 below describes the transcription of a gene X, that is the creation of a messenger RNA X triggered by the gene X. The creation of the protein X is then triggered by the mRNA X. (Note that the same example could be represented using the gene as reactant and

product, although it is semantically different.)

1283

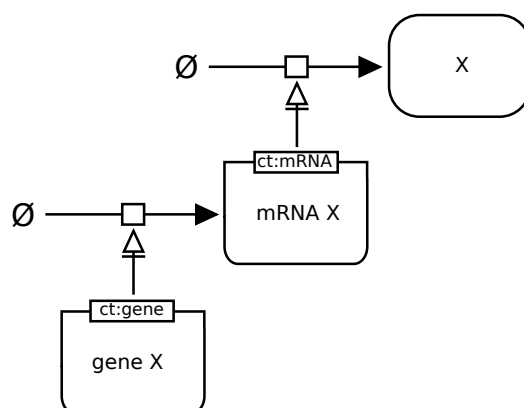


Figure 3.74: *The creation of a messenger RNA X triggered by the gene X.*

The example in Figure 3.75 below describes the transport of calcium ions out of the endoplasmic reticulum. Without IP3 receptor, there is not calcium flux, therefore, one cannot use a *stimulation*. The Necessary Stimulation instead represents this absolute stimulation.

1284

1285

1286

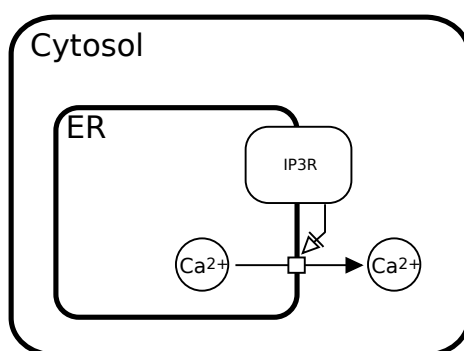


Figure 3.75: *The transport of calcium ions out of the endoplasmic reticulum into the cytosol. Note that IP3R crosses both compartment boundaries. This is allowed, but the Macromolecule should only belong to one of the compartments see section C.1 for more discussion of this issue.*

Changes from Previous Version

1287

The definition of ModulationArc did not exist in the previous version but there has been no changes to the glyphs and glyph semantics in this version.

1288

1289

3.3.35 LogicArc

1290

The LogicArc (figure 3.76) takes a quantity from either a LogicalOperator (see section 3.3.22) or an EntityPoolNode (see section 3.3.7) and converts it into a Boolean output, which serves as an input for a LogicalOperator (see section 3.3.22). How this is done is not defined, but one could imagine that when a threshold value of the quantity is exceeded the output is True.

1291

1292

1293

1294

Generalisation

1295

- SBGNArc (see section 3.3.6)

1296

Attributes

1297

No additional attributes.

1298

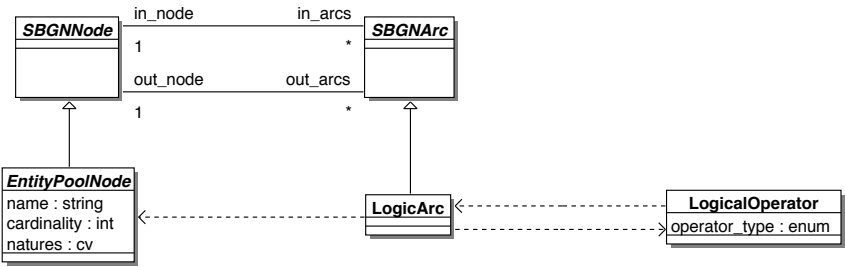


Figure 3.76: The UML definition of the LogicArc and its context.

Associations

No additional associations.

Rules and Constraints

- The `in_node` must be an instance of `EntityPoolNode` or `LogicalOperator`.
- The `out_node` must be an instance of `LogicalOperator`.

Notation

Glyph: Logic arc *Logic arc* is used to represent the fact that an entity influences the outcome of a logic operator.

SBO Term: SBO:0000398 ! logical relationship.

Origin: Any *EPN* (Section 3.3.7) or *logical operator* (Section 3.3.22).

Target: Any *logical operator* (Section 3.3.22).

End point: No particular symbol is used to represent a logic arc.

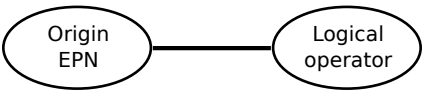


Figure 3.77: The Process Description glyph for logic arc.

Changes from Previous Version

No changes from the previous version.

3.3.36 EquivalenceArc

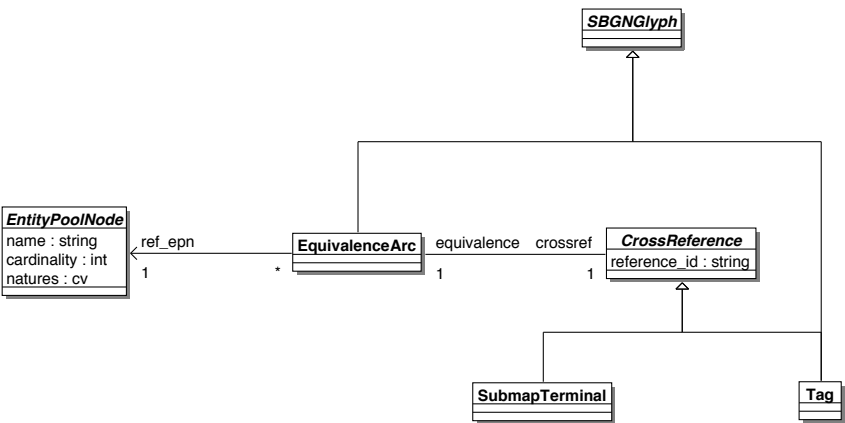


Figure 3.78: The UML definition of the EquivalenceArc and its context.

EquivalenceArc (figure 3.78) is the arc used to link a cross-reference to an EPN in another Process Description map (represented by CrossReference (see section 3.3.28)) with an EPN (EntityPoolNode (see section 3.3.7)) in this map.

Generalisation

- SBGNGlyph (see section 3.3.3)

Attributes

No additional attributes.

Associations

ref:CrossReference (1) The cross reference associated to be associated with an EPN by this class.

epn:EntityPoolNode (1) The EPN that the cross-reference refers to.

Rules and Constraints

No additional rules and constraints.

Notation

Glyph: *Equivalence arc*

SBO Term: Not applicable.

Origin: Any EPN (Section 3.3.7).

Target: Tag (Section 3.3.30).

End point: No particular symbol is used to represent an *equivalence arc*.

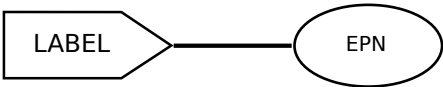


Figure 3.79: The Process Description glyph for Equivalence arc.

Changes from Previous Version

The relationship of EquivalenceArc to the SubmapTerminal (see section 3.3.29) was unclear in previous versions of the specification and has been clarified here.

3.3.37 CloneMarker

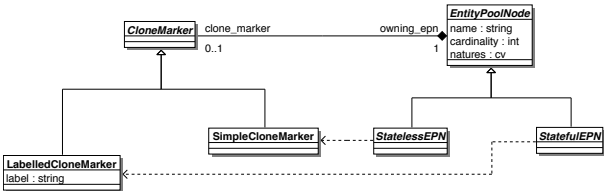


Figure 3.80: The UML definition of the StateVariable showing its relationship to StatefulEPN, Complex and Subunit.

If an EntityPoolNode (see section 3.3.7) is duplicated on a map, it is necessary to indicate this fact by the CloneMarker auxiliary unit (figure 3.80). The purpose of this marker is to provide the reader with a visual indication that this node has been cloned, and that at least one other occurrence of the EntityPoolNode can be found in the map (or in a submap; see Section 3.3.21). The clone marker takes two forms, simple and labeled, depending on whether the node being cloned can carry state variables

(i.e., whether it is a stateful EPN). Note that an EntityPoolNode belongs to a single compartment. If two classes named “X” are located in two different compartments, such as ATP in cytosol and ATP in mitochondrial lumen, they represent different Entity Pools, and therefore do not need to be marked as cloned.

Generalisation

- AuxiliaryUnit (see section 3.3.4)

Attributes

No additional attributes.

Associations

owning_epn:EntityPoolNode (1) The EPN that holds this clone marker.

Rules and Constraints

No additional rules and constraints.

Changes from Previous Version

Not defined in previous version.

3.3.38 SimpleCloneMarker

The SimpleCloneMarker (figure 3.80) is the unlabelled subclass CloneMarker. All duplicated instances of StatelessEPN must contain an instance of this class.

Generalisation

- CloneMarker (see section 3.3.37)

Attributes

No additional attributes.

Associations

No additional associations.

Rules and Constraints

- Only subclasses of StatefulEPN (see section 3.3.13) can contain labelled clone markers.

Notation

Simple clone marker

SBO Term: Not applicable.

Container: The simple (unlabeled) *clone marker* is a portion of the surface of an EPN that has been modified visually through the use of a different shade, texture, or color. Figure 3.81 illustrates this. The *clone marker* occupies the lower part of the EPN. The filled area must be smaller than the unfilled one.

Label: Not applicable.

Auxiliary Items: A *clone marker* does not carry any auxiliary items.

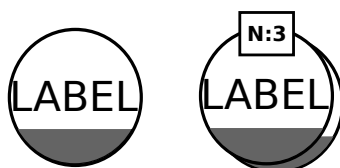


Figure 3.81: The Process Description glyph for simple clone marker applied to a simple chemical and a multimer of simple chemicals.

Figure 3.82 contains an example in which we illustrate the use of *clone markers* to clone the species ATP and ADP participating in different reactions. This example also demonstrates the chief drawbacks of using clones: it leads to a kind of dissociation of the overall network and multiplies the number of nodes required, requiring more work on the part of the reader to interpret the result. Sometimes these disadvantages are offset in larger maps by a reduction in the overall number of line crossings, but not always. In general, we advise that cloning should be used sparingly.

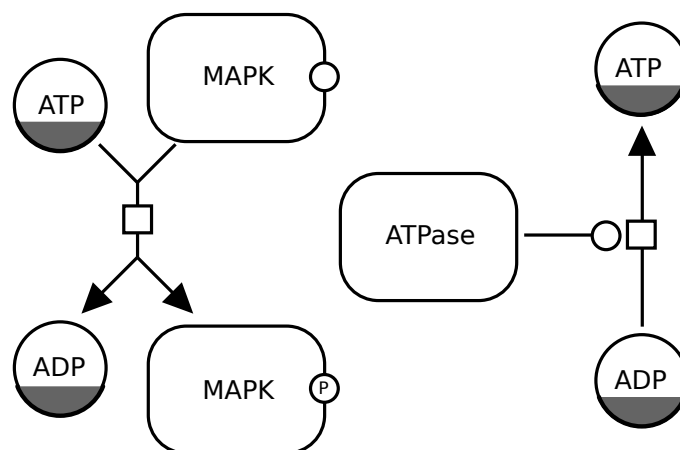


Figure 3.82: An example of using cloning, here for the species ATP and ADP.

Changes from Previous Version

No change from previous version.

3.3.39 LabelledClonerMarker

Unlike the class `SimpleCloneMarker`, the `LabeledCloneMarker` (figure 3.80) includes (unsurprisingly, given its name) an identifying label that can be used to identify equivalent clones elsewhere in the map. This is particularly useful for subclasses of `StatefulEPN` (see section 3.3.13), because these can have a large number of state variables displayed and therefore may be difficult to visually identify as being identical.

Generalisation

- `CloneMarker` (see section 3.3.37)

Attributes

label: string (**R**) The label that identified the clone. This label must start and end with an alphanumeric character, and cannot contain white space.

Associations

No additional associations.

Rules and Constraints

- At least two or more instances of a `LabeledCloneMarker` with the same label must exist in this same in a given Map (see section 3.3.2).
- Only subclasses of `StatefulEPN` (see section 3.3.13) can contain labelled clone markers.

Notation**Labeled clone marker****SBO Term:** Not applicable.

Container: The labeled *clone marker* is a portion of the surface of an *EPN* that has been modified visually through the use of a different shade, texture, or color. The *clone marker* occupies the lower part of the *EPN* glyph. The filled area must be smaller than the unfilled one, but the be large enough to have a height larger than the *clone marker's* label (cf below).

Label: A *clone marker* is identified by a label placed in an unbordered box containing a string of characters. The characters can be distributed on several lines to improve readability, although this is not mandatory. The label box must be attached to the center of the container. The label may spill outside of the container (the portion of the surface of the *EPN* that has been modified visually). The font color of the label and the color of the clone marker should contrast with one another. The label on a *labeled clone marker* is mandatory.

Auxiliary Items: A *clone marker* does not carry any auxiliary items.

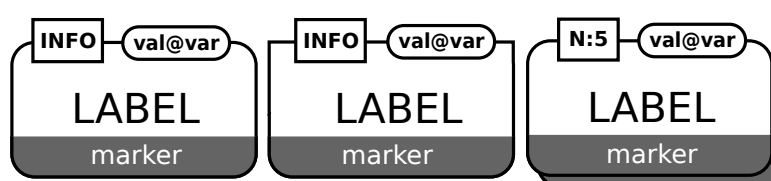


Figure 3.83: The Process Description glyph for labeled clone marker applied to a macromolecule, a nucleic acid feature and a multimer of macromolecules.

Changes from Previous Version

No changes from previous version.

3.4 Controlled vocabularies

Some classes in the SBGN Process Description language can contain particular kinds of textual annotation conveying information relevant to the class. Examples are the natures of an *EntityPoolNode* (see section 3.3.7) or *Compartment* (see section 3.3.24) or the value of the *StateVariable* (see section 3.3.26). The values held by these attributes can be taken from controlled vocabularies defined below. When displayed in some cases is mandatory to prefix a code indicating the type of controlled vocabulary used. This is in order to make it clear what the information is that the value refers to: for example 'mt' indicates that a value 'rna' is describing the material RNA.

In the rest of this section, we describe the controlled vocabularies (CVs) used in SBGN Process Description Level 1. They cover the following categories of information: an *EPN's* material type, an *EPN's* conceptual type, covalent modifications on macromolecules and the physical characteristics. These controlled vocabularies are *closed* in the sense that only the values defined for each CV can be used in a valid Process Description map and also closed because only the controlled vocabularies defined in this specification can be used in a valid Process Description map. We understand that this is of necessity restrictive, but in closing these definitions it means we can be clear about the meaning of all CV terms used in the specification. Updates to the CV terms and the CVs used are welcome and we encourage any changes or additions to be submitted as a tracker item at the address given on the front page of this specification.

3.4.1 Entity pool node material types

The material type of an *EPN* indicates its chemical structure. A list of common material types is shown in Table 3.2, but others are possible. The values are to be taken from the Systems Biology On-

tology (<http://www.ebi.ac.uk/sbo/>), specifically from the branch having identifier SBO:0000240 (material entity under entity). The labels are defined by SBGN Process Description Level 1.

Name	Label	SBO term
Non-macromolecular ion	mt:ion	SBO:0000327
Non-macromolecular radical	mt:rad	SBO:0000328
Ribonucleic acid	mt:rna	SBO:0000250
Deoxribonucleic acid	mt:dna	SBO:0000251
Protein	mt:prot	SBO:0000297
Polysaccharide	mt:psac	SBO:0000249

Table 3.2: A sample of values from the material types controlled vocabulary (Section 3.4.1).

The material types are in contrast to the *conceptual types* (see below). The distinction is that material types are about physical composition, while conceptual types are about roles. For example, a strand of RNA is a physical artifact, but its use as messenger RNA is a role.

3.4.2 Entity pool node conceptual types

An EPN's *conceptual type* indicates its function within the context of a given Process Description. A list of common conceptual types is shown in Table 3.3, but others are possible. The values are to be taken from the Systems Biology Ontology (<http://www.ebi.ac.uk/sbo/>), specifically from the branch having identifier SBO:0000241 (conceptual entity under entity). The labels are defined by SBGN Process Description Level 1.

Name	Label	SBO term
Gene	ct:gene	SBO:0000243
Transcription start site	ct:tss	SBO:0000329
Gene coding region	ct:coding	SBO:0000335
Gene regulatory region	ct:grr	SBO:0000369
Messenger RNA	ct:mRNA	SBO:0000278

Table 3.3: A sample of values from the conceptual types vocabulary (Section 3.4.2).

3.4.3 Macromolecule covalent modifications

A common reason for the introduction of state variables (Section 3.3.26) on an entity is to allow access to the configuration of possible covalent modification sites on that entity. For instance, a macromolecule may have one or more sites where a phosphate group may be attached; this change in the site's configuration (i.e., being either phosphorylated or not) may factor into whether, and how, the entity can participate in different processes. Being able to describe such modifications in a consistent fashion is the motivation for the existence of SBGN's covalent modifications controlled vocabulary.

Table 3.4 lists a number of common types of covalent modifications. The most common values are defined by the Systems Biology Ontology in the branch having identifier SBO:0000210 (addition of a chemical group under interaction→process→biochemical or transport reaction→

biochemical reaction→conversion). The labels shown in Table 3.4 are defined by SBGN Process Description Level 1; for all other kinds of modifications not listed here, the author of a Process Description must create a new label (and should also describe the meaning of the label in a legend or text accompanying the map).

Name	Label	SBO term
Acetylation	Ac	SB0:0000215
Glycosylation	G	SB0:0000217
Hydroxylation	OH	SB0:0000233
Methylation	Me	SB0:0000214
Myristoylation	My	SB0:0000219
Palmytoylation	Pa	SB0:0000218
Phosphorylation	P	SB0:0000216
Prenylation	Pr	SB0:0000221
Protonation	H	SB0:0000212
Sulfation	S	SB0:0000220
Ubiquitination	Ub	SB0:0000224

Table 3.4: A sample of values from the covalent modifications vocabulary (Section 3.4.3).

3.4.4 Physical characteristics

SBGN Process Description Level 1 defines a special unit of information for describing certain common physical characteristics. Table 3.5 lists the particular values defined by SBGN Process Description Level 1.

Name	Label	SBO term
Temperature	pc:T	SB0:0000147
Voltage	pc:V	SB0:0000259
pH	pc:pH	SB0:0000304

Table 3.5: A sample of values from the physical characteristics vocabulary (Section 3.4.4).

3.5 Entity Pool Node Identity and Cloning

All elements in an SBGN Process Description map have an implicit identity defined by the id attribute in SBGNElement (see section 3.3.1) so in that respect all elements drawn are unique. However, in some cases it is possible to draw two or elements that define identical information in the map. In some classes, such as the Compartment this is prohibited, but in others (EntityPoolNode) it is not, but requires special decoration (the clone marker) to indicate that the information is replicated on the map. We define this type of identity and “logical identity” and the attributes that so discriminate between elements as the “logical key”.

Clearly it is therefore important that we define the logical key for all the elements in the map and that we do this for the class definitions (section 3.3). The conventions used for this is as follows:

- If no logical key is specified then instance identity applies (see above).
- If the logical key is defined then this should be applied when determining equality between two elements.
- If an element can be replicated and two elements are determined to be logically identical, then instance identity should be used to determine uniqueness.
- If an element can be replicated if marked as a clone, then the above rule applies. If the element is not marked as cloned, then the only the logical key can be used when determining uniqueness.

Figure ?? illustrates how the above rules are applied in practise in the Process Description map. They also affect our understanding when reading a map. For example since all processes are unique, two *Process* glyphs connected to the same input and output indicates that the same “reaction” can be carried out by two distinct processes, with potentially different mechanisms.

3.6 Map and Submap Linking

Rules.

- Same namespace for map and submaps. Recursive so submaps of submaps share top-most map’s namespace. **Do we want to allow this?**
- Compartment’s logical key is it’s name. Compartment’s can be replicated across submaps and they all refer to the same compartment.
- EPNs replicated in both map and immediate submap must be associated with a cross-reference (terminal for main map, tag for submap).
- EPN replication is allowed between submaps separated by one or more other submaps, i.e. those not affected by the above rule.
- All reference_ids used must be unique within a given map and it’s immediate super-map and its immediate sub-map.
- submap folding: all tagged nodes in submap merge with those linked to terminals in super-map. If either node is cloned then all equivalent nodes in merged map must share the appropriate clone marker.

The submap is a visual device that allows the detail of an Process Description map to be exported into another Process Description map and replaced by a *submap* glyph, which acts as a place-holder. This is described and illustrated in Section 3.3.21. In the following discussion we will refer to the original map as the *main* map and the map containing the export detail as the submap.

1. For a valid mapping between an EPN in the map and submap to exist the identifiers in the *tag* and the submap terminal must be identical and their associated entity pool nodes must be identical.
2. If the same EPN is present in the map and a submap, then they must be mapped to each other.
3. Since the main map and submap share the same namespace, an EPN that is cloned in the main map must also be marked as cloned in the submap — even if there is only one copy of the EPN in the submap. The converse applies when the EPN in the submap is cloned².

A Submap is used to encapsulate processes (including all types of nodes and edges) one glyph. The *submap* hides its content to the users, and display only input terminals (or ports), linked to *EPNs* (Section 3.3.7). A *submap* is not equivalent to an *omitted process* (see Section 3.3.23). In the case of an SBGN description that is made available through a software tool, the content of a *submap* may be available to the tool. A user could then ask the tool to expand the *submap*, for instance by clicking on the icon representing the *submap*. The tool might then expand and show the *submap* within the same map (on the same canvas), or it might open it in a different canvas. In the case of an SBGN description made available in a book or a website, the content of the *submap* may be available on another page, possibly accessible via an hyperlink on the *submap*.

²This has the additional benefit of ensuring that main maps and submaps do not need to be modified if the submap is expanded and collapsed by a viewing or editing tool.

Figure 3.84 represents a *submap* that transforms glucose into fructose-6-phosphate. The *submap* carries five terminals, four linked to EPNs and one linked to a *compartment*. The latter is particularly important in the case of EPNs present only in a *compartment* enclosed in a *submap*, and that are not linked to terminals themselves. Note that the terminals do not define a “direction”, such as input or output. The flux of the reactions is determined by the context.

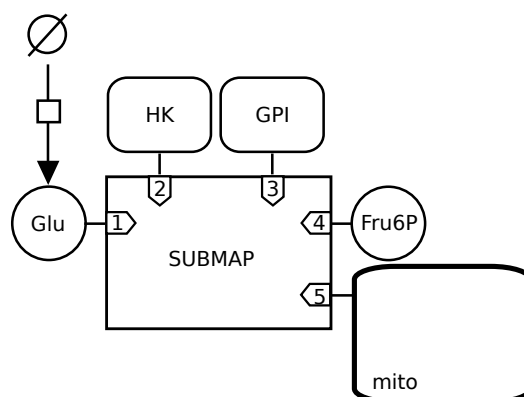


Figure 3.84: Example of a *submap* with contents elided.

The map in Figure 3.85 represents an unfolded version of a *submap*. Here, anything outside the *submap* has disappeared, and the internal *tags* are not linked to the corresponding external *terminals*. Note the tag 5, linking the compartment “mito” of the *submap* to the compartment “mito” outside the *submap*. The compartment containing Glu6P is implicitly defined as the same as the compartment containing Glu and Fru6P. There is no ambiguity because if Glu and Fru6P were in different compartments, one of them should have been defined within the *submap*.

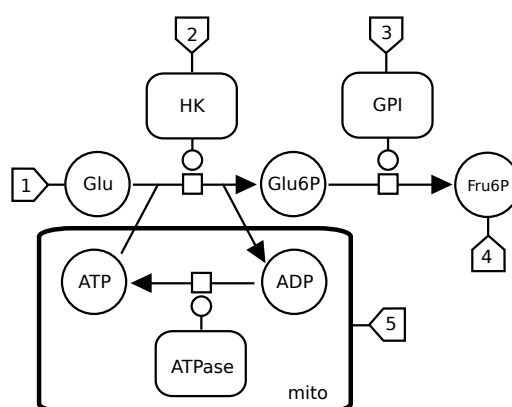


Figure 3.85: Example of an unfolded *submap*. The unfolded *submap* corresponds to the folded *submap* of Figure 3.84.

3.6.1 Compartment spanning

An *EPN* cannot *belong* to more than one *compartment*. However, an *EPN* can be *drawn* over more than one *compartment*. In such cases, the decision on which is the owning *compartment* is deferred to the drawing tool or the author. A *complex* may contain *EPNs* which belong to different *compartments* and in this way a *complex* can be used to describe entities that span more than one *compartment*.

This restriction makes it impossible to represent in a semantically correct way a macromolecule that spans more than one *compartment* — for example a receptor protein. It is clearly desirable to

be able to show a macromolecule in a manner that the biologist expects (i.e. spanning from the outside through the membrane to the inside). Therefore, the author is recommended to draw the macromolecule across compartment boundaries, but the underlying SBGN semantic model will assign it to only one. The assignment to a *compartment* may be decided by the software drawing tool or the author. Note that this has implications for auto-layout algorithms as they will only be able to treat such *entity pool nodes* as contained within a *compartment* and will have no way of knowing a macromolecule spans a compartment.

The current solution is consistent with other Systems Biology representations such as SBML and BioPAX. For more information about the problems representing membrane spanning proteins and the rationale behind the current solution see Section C.

Chapter 4

1552

Layout Rules for a Process Description

1553

4.1 Introduction

1554

The previous chapters describe the appearance and meaning of SBGN Process Description Level 1 components. Here we describe rules governing the visual appearance and aesthetics of the Process Description language. The components of a Process Description have to be placed in a meaningful way – a random distribution with spaghetti-like connections will most likely hide the information encoded in the underlying model, whereas an elegant placement of the objects, giving a congenial appearance of the maps, may reveal new insights. The arrangement of components in a map is called a *layout*.

1555
1556
1557
1558
1559
1560
1561

SBGN Process Descriptions should be easily recognisable not only by the glyphs used, but also by the general style of the layout. However, the arrangement of the components is a complex art in itself, and there is no simple rule which can be applied to all cases. Therefore this section provides rules for the layout of process description maps, divided into two categories:

1562
1563
1564
1565

1. requirements, i. e. rules which **must** be fulfilled by a layout, and
2. recommendations, i. e. rules which **should** be followed if possible.

1566
1567

In addition, we provide a list of additional suggestions which may help in producing aesthetically more pleasant layouts, possibly easier to understand.

1568
1569

Those layout rules are independent of the method used to produce the map, and apply to both manually drawn maps as well as maps produced by an automatic layout algorithm. The rules do not deal with interactive aspects (e. g. the effect of zooming). Further information about automatic network layout (graph drawing) can be found, for example, in the books of Di Battista and co-authors [1] and Kaufmann and Wagner [2].

1570
1571
1572
1573
1574

Please note that the color of objects do not carry any meaning in SBGN. Although one can use colors to emphasize part of a map or encode additional information, the meaning of the map should not depend on the colors. Furthermore, objects can have different sizes and size is also meaningless in SBGN. For example, a process node may be larger than a protein node. Also the meaning of a graph should be conserved upon scaling as far as possible.

1575
1576
1577
1578
1579

4.2 Requirements

1580

Requirements are rules which **must** be fulfilled by a layout to produce a valid Process Description map.

1581
1582

4.2.1 Node-node overlaps

1583

Nodes are only allowed to overlap in two cases when they are allowed to contain other nodes — as described in Chapter 3. Otherwise, nodes are not allowed to overlap (Figure 4.1). This includes the

1584
1585

touching of nodes. Touching is not allowed apart from the case where it has a specific meaning, e.g. two macromolecules touching each other within a complex because they form the complex.

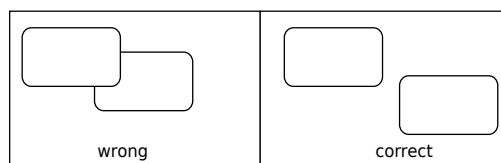


Figure 4.1: *Nodes must not overlap.*

4.2.2 Node-edge crossing

Edges must be drawn on the top of a the node (Figure 4.2). See also recommendation Section 4.3.1.

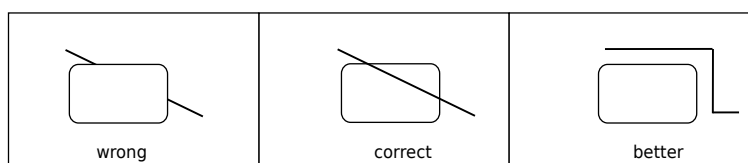


Figure 4.2: *If an edge crosses a node, the edge must be drawn on top of the node.*

4.2.3 Node border-edge overlaps

Edges are not allowed to overlap the border lines of nodes (Figure 4.3).

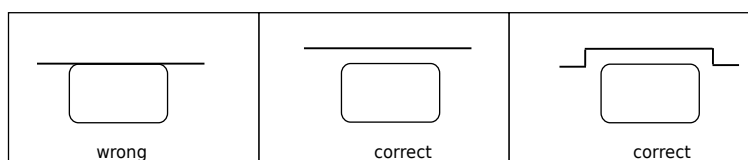


Figure 4.3: *Edges must not overlap node borders.*

4.2.4 Edge-edge overlaps

Edges are not allowed to overlap (Figure 4.4). This includes touching of edges. Furthermore, an edge is neither allowed to cross itself nor to cross a boundary of node more than twice or other edges more than once.

4.2.5 Node orientation

Nodes have to be drawn horizontally or vertically, any other rotation of elements is not allowed (Figure 4.5).

4.2.6 Node-edge connection

1. The arcs linking the square glyph of a *process* to the *consumption* and *production arcs* are attached to the center of opposite sides (Figure 4.6).

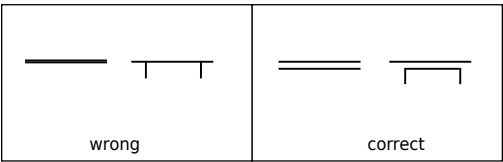


Figure 4.4: Edges must not overlap.

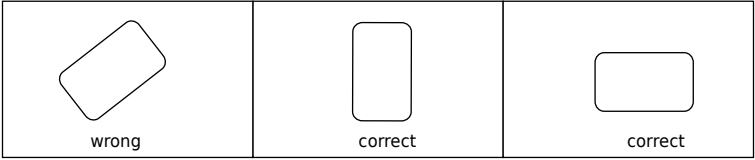


Figure 4.5: The node orientation must be horizontally or vertically.

2. The modulatory arcs are attached to the other two sides, but not necessarily all to the center, as several modifiers can affect the same process node.

1602
1603

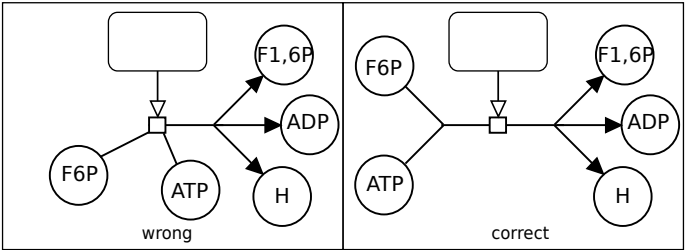


Figure 4.6: Arcs between a process and the consumption and production arcs must be attached to the center of opposite sides, modulatory arcs must be attached to the other two sides.

4.2.7 Node labels

1604

At least a part of the label (unbordered box containing a string of characters) has to be placed inside the node it belongs to. Node labels are not allowed to overlap other nodes or other labels (this includes touching of other nodes or labels).

1605
1606
1607

4.2.8 Edge labels

1608

Edge labels are not allowed to overlap nodes. This includes touching of nodes.

1609

4.2.9 Compartments

1610

If a process has all participants in the same compartment the process node and all edges/arcs should be drawn in this compartment. If a process has participants in at least two different compartments, the process node has to be either in a compartment where the process has at least one participant or in the empty space.

1611
1612
1613
1614

4.3 Recommendations

1615

Recommendations are rules which should be followed if possible and generally should improve the clarity of the diagram.

1616
1617

4.3.1 Node-edge crossing

Situations where edges and nodes cross should be avoided. Note that some crossings may be unavoidable, e. g. the crossing between an edge and a compartment border or an edge and a complex (if the edge connects an element inside the complex with something outside).

4.3.2 Labels

Labels should be horizontal. Node labels should be placed completely inside the node if possible. Edge labels should be placed close to the edge and avoid overlapping the edge as well as other edge labels.

4.3.3 Avoid edge crossings

The amount of crossings between edges should be minimized.

4.3.4 Branching of association and dissociation

The branching points of *association* and *dissociation* nodes should be placed closed to the symbol of the process, if possible at a distance comparable than, or smaller to, the diameter of the symbol defining the process (Figure 4.7).

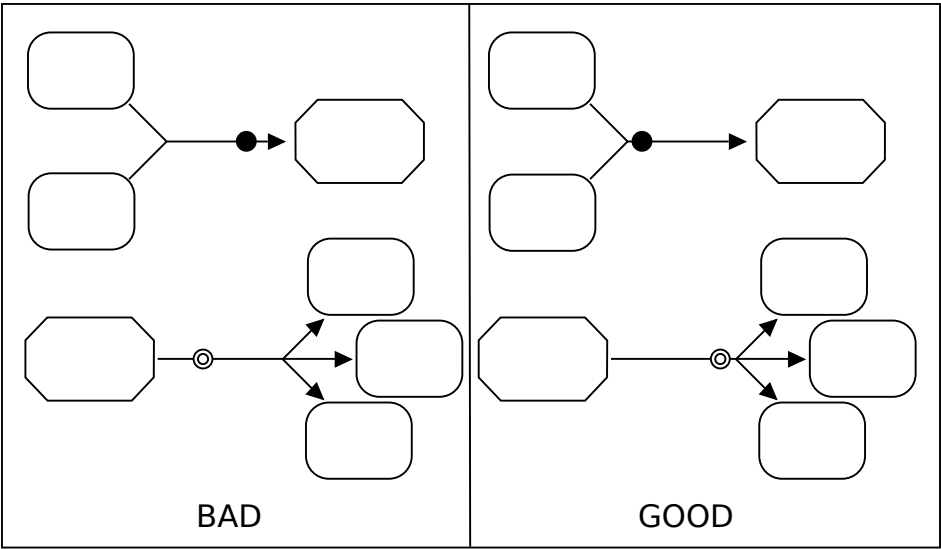


Figure 4.7: Branching points should be close to association and dissociation symbols.

4.3.5 Units of information

Units of information should not hide the structure of the corresponding node and should not overlap other elements (Figure 4.8).

4.4 Additional suggestions

Here is a list of additional layout suggestions which may help improve the aesthetics and clarity of Process Description maps.

- Angle of edge crossings: If edge crossing cannot be avoided then the edges should cross with an angle close to 90 degrees.

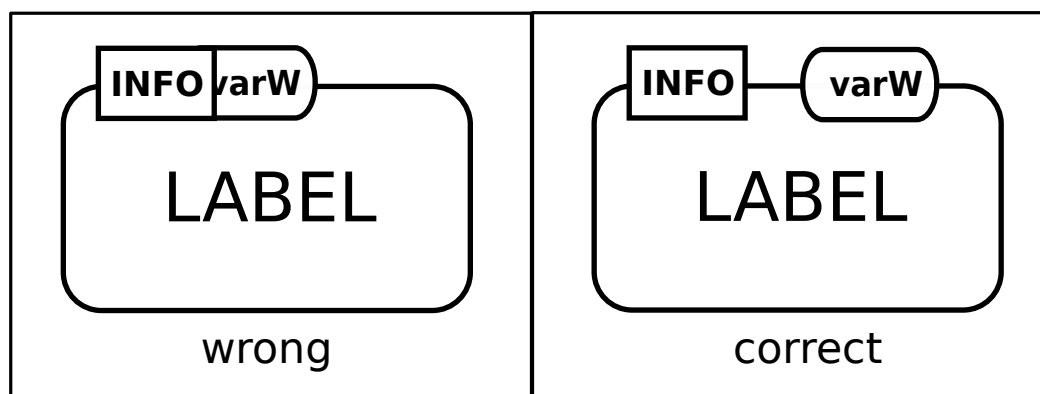


Figure 4.8: *Units of information should not overlap with any other element.*

- Drawing area and width/height ratio: The drawing should be compact and the ratio between the width and the height of the drawing should be close to 1. 1640 1641
- Edge length: Long edges should be avoided. 1642
- Number of edge bends: Edges should be drawn with as few bends as possible. 1643
- Similar and symmetric parts: Similar parts of a map should be drawn in a similar way, and symmetric parts should be drawn symmetrically. 1644 1645
- Proximity information: Related elements (e. g. nodes connected by a process or all elements within a compartment) should be drawn close together. 1646 1647
- Directional information: Subsequent processes (e. g. a sequence of reactions) should be drawn in one direction (e. g. from top to bottom or from left to right). 1648 1649
- Compartments: It can help clarity to use a different background shade or color for each compartment. 1650 1651

Chapter 5

1652

Acknowledgments

1653

Here we acknowledge those people and organisations that assisted in the development of this and previous releases of the SBGN Process Description language specification. First we specifically acknowledge those who contributed directly to each revision of the specification document, followed by a comprehensive acknowledgement of contributors that attended workshops and forum meetings or in some other way provided input to the standard. Finally, we acknowledge the bodies that provided financial support for the development of the standard.

1654

1655

1656

1657

1658

1659

5.1 Level 1 Release 1.0

1660

The specification of was written by Nicolas Le Novère, Stuart Moodie, Anatoly Sorokin, Michael Hucka, Falk Schreiber, Emek Demir, Huaiyu Mi, Yukiko Matsuoka, Katja Wegner and Hiroaki Kitano. In addition, the specification benefited much from the help of (in alphabetical order) Frank Bergmann, Sarala Dissanayake, Ralph Gauges, Peter Ghazal, Lu Li, and Steven Watterson.

1661

1662

1663

1664

5.2 Level 1 Release 1.1

1665

The specification of SBGN PD Level 1.1 was written by Stuart Moodie and Nicolas Le Novère, with contributions from (in alphabetical order) Frank Bergmann, Sarah Boyd, Emek Demir, Sarala Wimalaratne, Yukiko Matsuoka, Huaiyu Mi, Falk Schreiber, Anatoly Sorokin, Alice Villéger.

1666

1667

1668

5.3 Level 1 Release 1.2

1669

The specification of SBGN PD Level 1.2 was modified by Stuart Moodie, with contributions from (in alphabetical order) Sarah Boyd, Nicolas Le Novère, Huaiyu Mi.

1670

1671

5.4 Level 1 Release 1.3

1672

The specification of SBGN PD Level 1.3 was modified by Stuart Moodie, with contributions from (in alphabetical order), Tobias Czauderna, Nicolas Le Novère, Anatoly Sorokin.

1673

1674

5.5 Comprehensive list of acknowledgements

1675

Here is a more comprehensive list of people who have been actively involved in SBGN development, either by their help designing the languages, their comments on the specification, help with development infrastructure or any other useful input. We intend this list to be complete. We are very sorry if we forgot someone, and would be grateful if you could notify us of any omission.

1676

1677

1678

1679

Mirit Aladjemm, Frank Bergmann, Sarah Boyd, Laurence Calzone, Melanie Courtot, Emek Demir, Ugur Dogrusoz, Tom Freeman, Akira Funahashi, Ralph Gauges, Peter Ghazal, Samik Ghosh, Igor Goryanin, Michael Hucka, Akiya Jouraku, Hideya Kawaji, Douglas Kell, Sohyoung Kim, Hiroaki Kitano, Kurt Kohn, Fedor Kolpakov, Nicolas Le Novère, Lu Li, Augustin Luna, Yukiko Matsuoka, Huaiyu Mi, Stuart Moodie, Sven Sahle, Chris Sander, Herbert Sauro, Esther Schmidt, Falk Schreiber, Jacky Snoep, Anatoly Sorokin, Jessica Stephens, Linda Taddeo, Steven Watterson, Alice Villéger, Katja Wegner, Sarala Wimalaratne, Guanming Wu.

The authors are also grateful to all the attendees of the SBGN meetings, as well as to the subscribers of the sbgn-discuss@sbgn.org mailing list.

5.6 Financial Support

The development of SBGN was mainly supported by a grant from the Japanese *New Energy and Industrial Technology Development Organization* (NEDO, <http://www.nedo.go.jp/>). The *Okinawa Institute of Science and Technology* (OIST, <http://www.oist.jp/>), the *AIST Computational Biology Research Center* (AIST CBRC, <http://www.cbrc.jp/index.eng.html>) the *British Biotechnology and Biological Sciences Research Council* (BBSRC, <http://www.bbsrc.ac.uk/>) through a Japan Partnering Award, the *European Media Laboratory* (EML Research gGmbH, <http://www.eml-r.org/>), and the *Beckman Institute at the California Institute of Technology* (<http://bnmc.caltech.edu>) provided additional support for SBGN workshops. Some help was provided by the *Japan Science and Technology Agency* (JST, <http://www.jst.go.jp/>) and the *Genome Network Project* of the Japanese Ministry of Education, Sports, Culture, Science, and Technology (MEXT, <http://www.mext.go.jp/>) for the development of the gene regulation network aspect of SBGN, and from the *Engineering and Physical Sciences Research Council* (EPSRC, <http://www.epsrc.ac.uk>) during the redaction of the specification.

Appendix A

Complete examples of Process Description Maps

The following maps present complete examples of SBGN Process Descriptions representing Biological processes. They by no mean exhaust the possibilities of SBGN Process Description Level 1.

Figure A.1 presents an example of metabolic pathway, that exemplifies the use of the *EPNs simple chemical, macromolecule, and clone marker*, the *PNs process*, and the *connecting arcs consumption, production and catalysis*.

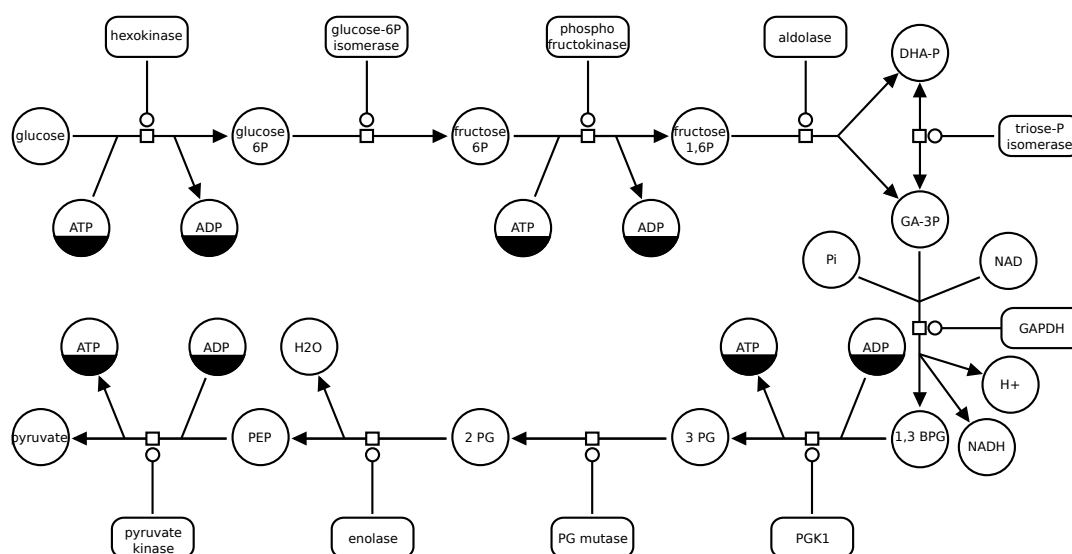


Figure A.1: Glycolysis. This example illustrates how SBGN can be used to describe metabolic pathways.

Figure A.2 presents an example of signalling pathway, that exemplifies in addition the use of the *EPNs phenotype, and state variable*, the *containers complex, compartment and submap*, the *PNs association*, and the *connecting arcs stimulation*. Note the complex IGF and IGF receptor, located on the boundary of the compartment. This position is only for user convenience. The complex has to belong to a given compartment in SBGN Process Description Level 1.

Figure A.3 is an expanded version of the submap present on the map present in Figure A.2. It shows the use of *tag*.

Figure A.4 introduces an SBGN Process Description that spans several compartments. Note that the compartment "synaptic vesicle" is not **contained** in the compartment "synaptic button" but **overlaps** it. The *simple chemical* "ACh" of the "synaptic vesicle" is not the same *EPN* than the "ACh" of the

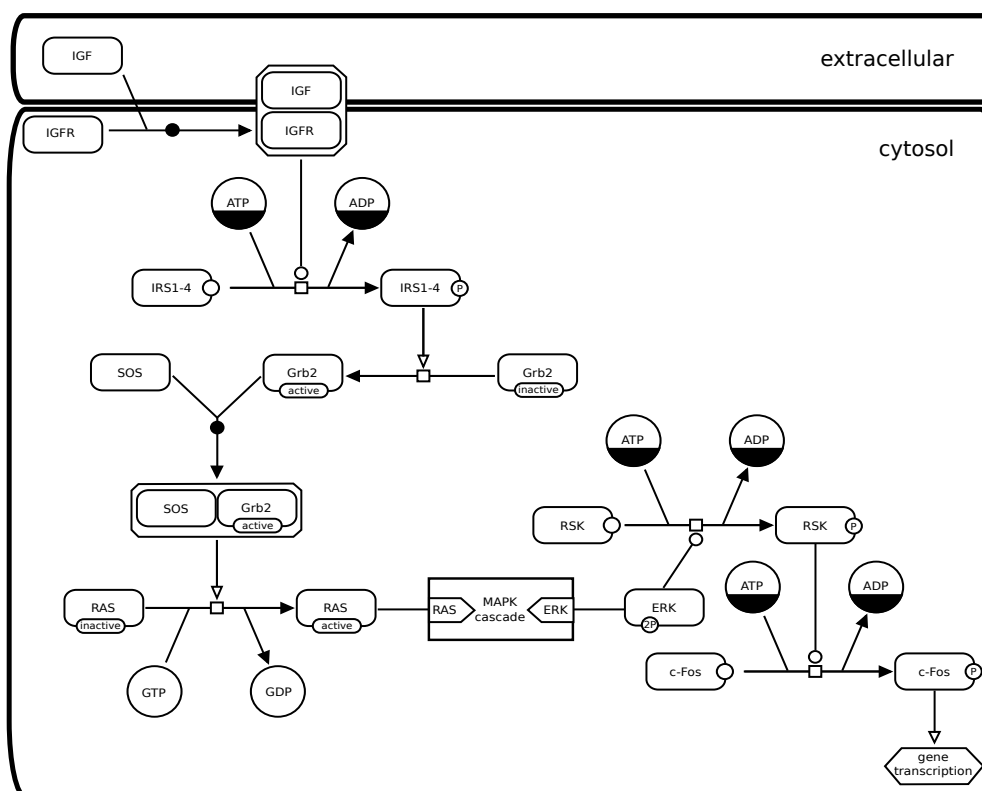


Figure A.2: Insulin-like Growth Factor (IGF) signalling. This example shows the use of compartments and how details can be hidden by using a submap. The submap is shown on Figure A.3.

“synaptic button” and of “synaptic cleft”. The situation is similar with the compartments “ER” and “muscle cytosol”. The map exemplifies the use of the *PN omitted* and *dissociation*, and the *connecting*

1721

1722

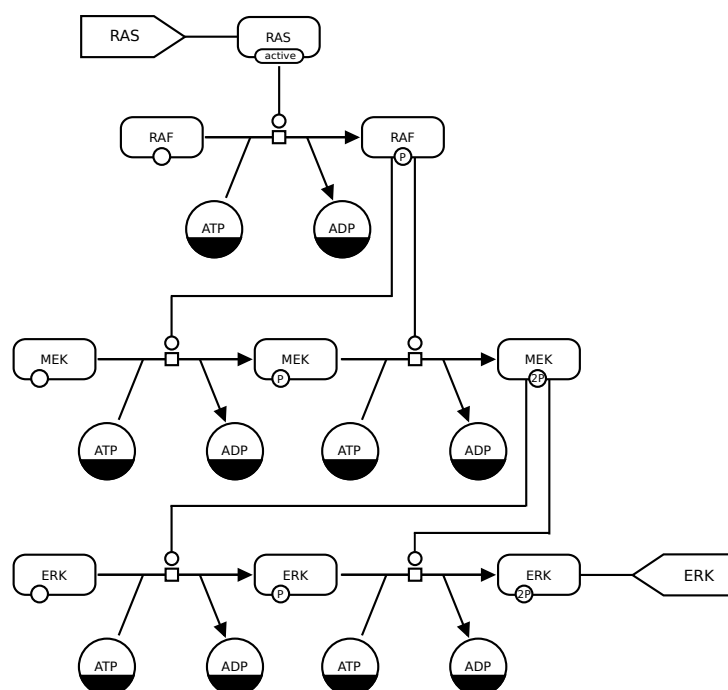


Figure A.3: A submap of the previous map showing the MAPK cascade.

arc necessary stimulation.

1723

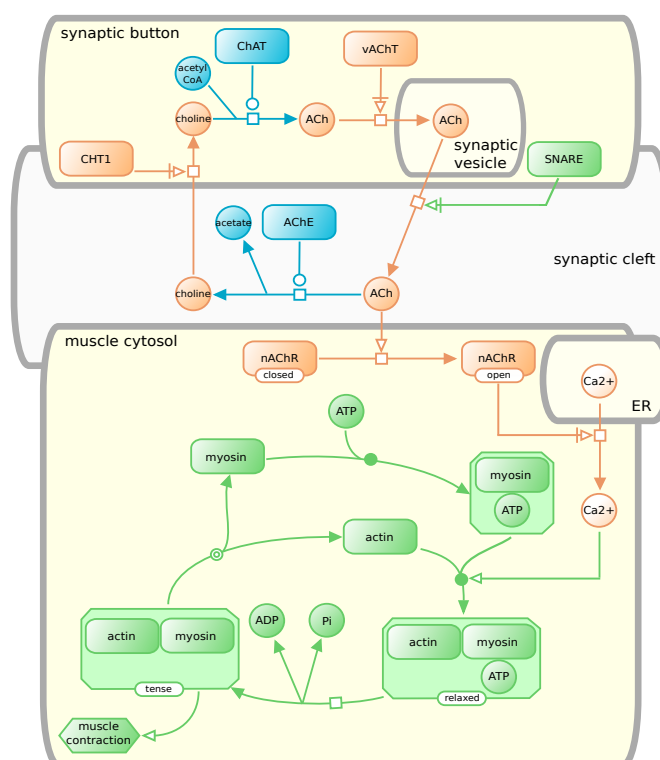


Figure A.4: Neuronal/Muscle signalling. A description of inter-cellular signalling using SBGN.

Figure A.5 introduces the use of SBGN Process Description Level 1 to encode gene regulatory networks. It also show the use of the EPNs Source and the logical operator and.

1724

1725

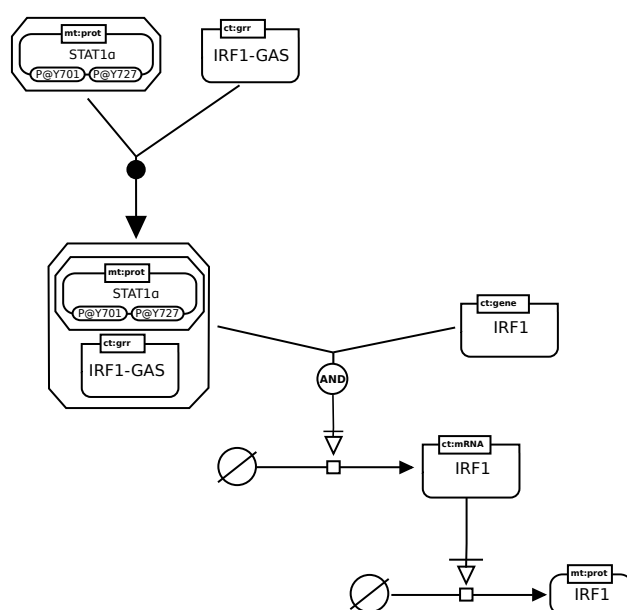


Figure A.5: Activated *STAT1α* induction of the *IRF1* gene. An example of gene regulation using logical operators.

Appendix B

1726

Reference card

1727

Print the summary of SBGN symbols on the next page for a quick reference.

1728

Appendix C

1730

Issues postponed to future levels

1731

C.1 Multicompartment entities

1732

The problem of entities, such as macromolecules, spanning several compartments proved to be a challenge for the community involved in the development of SBGN Process Description Level 1. It was thus decided to leave it for a future Level. It turns out there is at the moment no obvious solution satisfactory for everyone. Three broad classes of solutions have been identified so far:

- One can systematically locate an *EPN* in a given *compartment*, for instance a transmembrane receptor in a membrane. However, the reactions of this entity with entities represented by *EPN* in other compartments, such as extracellular ligands and second messenger systems, will create artificial transport reactions.
- One can represent the domains of proteins in different compartments by *macromolecules*, and link all those macromolecules in a *complex* spanning several compartments. However, such a representation would be very confusing, implying that the domains are actually different molecules linked through non-covalent bonds.
- One can accept *macromolecules* that span several compartments, and represent domains as *units of information*. Those *units of information* should then be located in given compartments. To make a full use of such a representation, one should then start and end connecting arcs on given *units of information*, something prohibited by the current specification.

C.2 Logical combination of state variable values

1749

The value of a *state variable* has to be perfectly defined in SBGN Process Description Level 1. If a state variable can take the alternative values 'A', 'B' and 'C', one cannot attribute it values such as 'non-A', 'A or B', 'any' or 'none'. As a consequence some biochemical processes cannot be easily represented because of the very large number of state to enumerate. The decision to forbid such a Boolean logic lies in the necessity of maintaining truth path all over an SBGN map.

C.3 Non-chemical entity nodes

1755

The current specification cannot represent combinations of events and entities. For instance a variable "voltage" cannot be controlled by a difference of concentration between different entities, such as a given ion in both sides of a membrane.

C.4 Generics

1759

SBGN Process Description Level 1 does not provide mechanisms to sub-class *EPNs*. There is no specific means of specifying that *macromolecules* or *nucleic acid features* X1, X2 and X2 are subclasses of X. Therefore, any process that applies to all the subtypes of X has to be triplicated. That situation can easily generate combinatorial explosions of the number of *EPNs* or *PNs*.

1760

1761

1762

1763

C.5 State and transformation of compartments

1764

In SBGN Process Description Level 1 a *compartment* is a stateless entity. It cannot carry *state variables*, and cannot be subjected to process modifying a state. As a result, a *compartment* cannot be transformed, moved, split or merged with another. If one want to represent the transformation of a compartment, one has to create the start and end compartments, and represent the transport of all the *EPNs* from one to the other. This is not satisfactory, and should be addressed in the future.

1765

1766

1767

1768

1769

Appendix D

1770

Revision History

1771

D.1 Version 1.0 to Version 1.1

1772

Below are the changes incorporated into Version 1.1 of the SBGN Process Description Level 1 specification.

1773

1774

Description	Tracker ID
Regarding modulation of reversible processes, changed “should” to “must” be represented by two <i>process</i>	
Removed “The connectors and the box move as a rigid entity” in the definition of <i>process</i>	
Changed the definition of process node to “represent processes that transform one or several EPNs into <i>one or several EPNs, identical or different</i> ”	
Changed SBO term of <i>compartment</i> From SBO:0000289 (functional compartment) to SBO:0000290 (physical compartment)	
Reorganised classification of glyphs	
Reorganised glyph section to reflect the above changes	
Revised reference card to reflect changes in glyph organisation	
Revised logic operators throughout spec to make sure input and output arcs meet before attaching to the glyph - as with processes.	
Added enumerated rules to grammar section. This is probably not complete, but should help the implementation of semantic validation by software tools. The hope is this will be refined as tools start validating maps.	
Updated UML maps and data dictionary to be consistent with rest of changes to spec.	
Definition of cardinality is ambiguous	2840996
<i>Sink and source</i> are lumped together	2726435
SBO terms are incorrect or missing.	2841261
<i>Compartment</i> description is confusing and contradictory.	2841122
<i>Clone marker</i> fill percentages unhelpful.	2841114
Use of CV for physical characteristic not clear.	2841085
Definition of Cardinality is ambiguous.	2840996
input to AND on IFN example.	2804326
more SBO terms for <i>multimers</i>	2803593
Legend of figure 2.20 is incorrect	2803537
legend of figure 3.2	2802990
continued on next page	

1775

<i>continued from previous page</i>	
Description	Tracker ID
Compartment colouring	2745703
Errors in diag a4.	2664912
Change name of trigger glyph.	2664908
Transition should be renamed process.	2664862
Converting arcs tautological.	2664843
Example invalid.	2545870
consumption and production.	2388317
Should require circles to be distinguishable from ellipses	2219388
Figure 2.53	2162619
Reference card: production	2104471
Figure 2.42 is wrong	2104465
Mistake in the multi-cellular example	2395488
Should not prevent processes having identical in and out	2664933
No description of linking to subunit rules.	2545810
Extensively revised the grammar section. The UML diagrams have been simplified to show glyph taxonomy, and the data dictionary has been pruned to just show glyph identity. The some syntax rules have been moved into semantics and the rules reformulated to make them easier to understand.	
Eliminated duplicate rules in layout section and revised text slightly.	
Phenotype cloning?	2989007
Perturbing agent description	2940021

1776

D.2 Version 1.1 to Version 1.2

1777

Below are the changes incorporated into Version 1.2 of the SBGN Process Description Level 1 specification.

1778

1779

Description	Tracker ID
Perturbing agent description	2940021
Members of complex touching	2849273
PD Reference card error for submap glyph	3029242

1780

D.3 Version 1.2 to Version 1.3

1781

Below are the changes incorporated into Version 1.3 of the SBGN Process Description Level 1 specification.

1782

1783

Description	Tracker ID
Incorrect editor on title page	
Typos in acknowledgements	
Fixed typo in item on catalysis in section ??.	
State variables figure 2.6 V1.2	3090543

1784

Bibliography

1785

- [1] G. Di Battista, P. Eades, R. Tamassia, and I.G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, New Jersey, 1998. 1786
1787
- [2] M. Kaufmann and D. Wagner. *Drawing Graphs: Methods and Models*, volume 2025 of *Lecture Notes in Computer Science Tutorial*. Springer, 2001. 1788
1789