Systems Biology Graphical Notation: Process Description language Level 1

Release 1.1 RC3

Date: 28 August, 2009

Disclaimer: This is a release candiate of the SBGN Process Description language Level 1 Release 1.1 specification. Please report any errors to the issues tracker at http://sourceforge.net/projects/sbgn/develop. It is not a normative document.

Editors:

Stuart Moodie	*School of Informatics, University of Edinburgh, UK*
Nicolas Le Novère	*EMBL European Bioinformatics Institute, UK*
Anatoly Sorokin	*School of Informatics, University of Edinburgh, UK*
Huaiyu Mi	*SRI International, USA*
Falk Schreiber	*IPK Gatersleben & MLU Halle, Germany*

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://sourceforge.net/tracker/?group_id=178553&atid=1082245.

Contents

Chapter 1

Introduction

The goal of the **S**ystems **B**iology **G**raphical **N**otation (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

Process Description Glyphs

2.1 Overview

To set the stage for what follows in this chapter, 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.

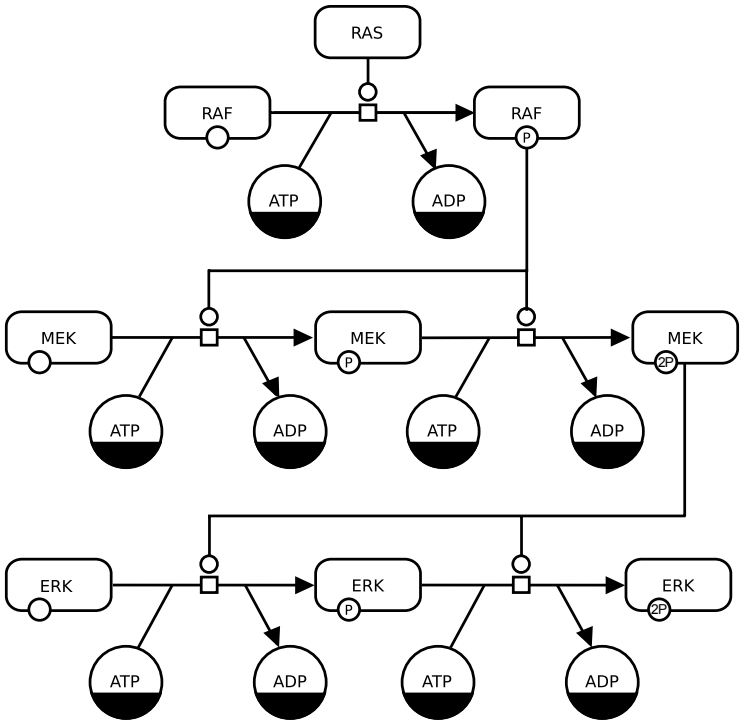

Figure 2.1: *This example of a Process Description uses two kinds of entity pool nodes: one for pools of different macromolecules (Section 2.4.3) and another for pools of simple chemicals (Section 2.4.2). Most macromolecule nodes in this map are adorned with state variables (Section 2.3.2) representing phosphorylation states. This map uses one type of process node, the process node (Section 2.8.1), and three kind of connecting arc, consumption (Section 2.9.1), production (Section 2.9.2) and catalysis (Section 2.9.5). Finally, some entity pool nodes have dark bands along their bottoms; these are clone markers (Section 2.3.3) 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 phosphorylated 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 on the preceding page, 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 on the previous page, those links describe consumption Section 2.9.1, production Section 2.9.2 and catalysis Section 2.9.5, 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 2.3.3). The details of this and other aspects of Process Description notation are explained in the rest of this chapter.

Table 2.1 summarizes the different SBGN abstractions described in this chapter.

Component	Abbrev.	Role	Examples
Entity pool node	EPN	A population of entities that cannot be distinguished from each other	Specific macromolecules or other chemical species
Container node	CN	An encapsulation of one or more other SBGN constructs	Complexes, compartments
Process node	PN	A process that transforms one or more EPNs into one or more other EPNs	Process, association, dissociation
Arc	—	Links between EPNs or CNs to PNs or CNs to indicate influences	Production, catalysis, inhibition
Logical operators	—	Combines one or several inputs into one output	Boolean *and*, *or*, *not*

Table 2.1: *Summary of Process Description components and their roles.*

2.2 Controlled vocabularies used in SBGN Process Description Level 1

Some glyphs in SBGN Process Descriptions can contain particular kinds of textual annotations conveying information relevant to the purpose of the glyph. These annotations are *units of information* (Section 2.3.1) or *state variable* (Section 2.3.2). An example is in the case of multimers, which can have a unit of information conveying the number of monomers composing the multimer. Other cases are described throughout the rest of this chapter.

The text that appears as the unit of information decorating an Entity Pool Node (EPN) must in most cases be prefixed with a controlled vocabulary term indicating the type of information being expressed. The prefixes are mandatory except in the case of macromolecule covalent modifications (Section 2.2.3). Without the use of controlled vocabulary prefixes, it would be necessary to have different glyphs to indicate different classes of information; this would lead

to an explosion in the number of symbols needed.

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, the physical characteristics of compartments, and cardinality (e.g., of multimers). In each case, some CV terms are predefined by SBGN, but unless otherwise noted, *they are not the only terms permitted*. Authors may use other CV values not listed here, but in such cases, they should explain the term's meanings in a figure legend or other text accompanying the map.

2.2.1 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 2.2, 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: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 2.2: *A sample of values from the* material types *controlled vocabulary (Section 2.2.1).*

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.

2.2.2 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 2.3, 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 2.3: *A sample of values from the* conceptual types *vocabulary (Section 2.2.2).*

2.2.3 Macromolecule covalent modifications

A common reason for the introduction of state variables (Section 2.3.2) 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 2.4 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 2.4 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`	`SBO:0000215`
Glycosylation	`G`	`SBO:0000217`
Hydroxylation	`OH`	`SBO:0000233`
Methylation	`Me`	`SBO:0000214`
Myristoylation	`My`	`SBO:0000219`
Palmytoylation	`Pa`	`SBO:0000218`
Phosphorylation	`P`	`SBO:0000216`
Prenylation	`Pr`	`SBO:0000221`
Protonation	`H`	`SBO:0000212`
Sulfation	`S`	`SBO:0000220`
Ubiquitination	`Ub`	`SBO:0000224`

Table 2.4: *A sample of values from the* covalent modifications *vocabulary (Section 2.2.3).*

2.2.4 Physical characteristics

SBGN Process Description Level 1 defines a special unit of information for describing certain common physical characteristics. Table 2.5 lists the particular values defined by SBGN Process Description Level 1. It is anticipated that these will be used to describe the nature of a *perturbing agent* (section 2.4.8) or a *phenotype* (section 2.8.6).

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

Table 2.5: *A sample of values from the* physical characteristics *vocabulary (Section 2.2.4).*

2.2.5 Cardinality

SBGN Process Description Level 1 defines a special unit of information usable on multimers for describing the number of monomers composing the multimer. Table 2.6 shows the way in which the values must be written. Note that the value is an positive non-zero integer, and not (for example) a range. There is no provision in SBGN Process Description Level 1 for specifying a range in this context because it leads to problems of entity identifiability.

Name	Label	SBO term
cardinality	`N:#`	`SBO:0000364`

Table 2.6: *The format of the possible values for the* cardinality *unit of information (Section 2.2.5). Here, # stands for the number; for example, "`N:5`".*

2.3 Auxiliary Units

Auxiliary units are glyphs that decorate other glyphs, providing additional information that may be useful to the reader. These can provide annotation (*unit of information*), state information (*state variable*) or indicate duplication of entity pool nodes (*clone marker*).

2.3.1 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.

The label defines the information carried by the *unit of information*. For certain predefined types of information having controlled vocabularies associated with them, SBGN defines specific prefixes that must be included in the label to indicate the type of information in question. The controlled vocabularies predefined in SBGN Process Description Level 1 are described in Section 2.2 and summarized in the following list:

`pc` container physical characteristic

`mt` entity pool material type
`ct` entity pool conceptual type
`N` multimer cardinality

Auxiliary items:

A *unit of information* does not carry any auxiliary items.

Figure 2.2: *The Process Description glyph for* unit of information.

2.3.2 Glyph: *State variable*

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 *state variables* with an entity; 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 current values of all its *state variables*, and the values of the *state variables* of all its possible components, recursively.

SBO Term:

Not applicable.

Container:

A *state variable* is represented by an elliptical container, as shown in Figure 2.3. The ellipse's long 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 of information* should be located on the mid-line of the border of the *EPN*.

Label:

The identification of an instance of a *state variable* is carried by one or two unbordered boxes, each containing a string of characters. The characters cannot be distributed on several lines. One box is mandatory, and contains the value of the *state variable*. The value may be empty; an example of a situation where this might arise is an unphosphorylated phosporylation site. The second box is optional and carries the identification of the *state variable*. This identification should be present if confusion is possible between several state varibles (e.g., several phosphorylation sites). The center of the combination of the boxes located in the container box is superposed to the center of this container box. Optionally, the identification of the *state variable* can be located outside the *state variable* container box. This is **strongly** discouraged. See Figure 2.4 on the next page for some examples of problems arising if the identification of a state variable is located outside the state variable. 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*.

Auxiliary items:

A *state variable* does not carry any auxiliary items.

Figure 2.3: *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".

The label of a *state variable* should, if possible, be displayed within the ellipse. In the top half of Figure 2.4, we show some examples of pathological cases that lead to confusion in the association between variable labels and values. Compare the discouraged examples with the recommended version in the bottom half of the figure.

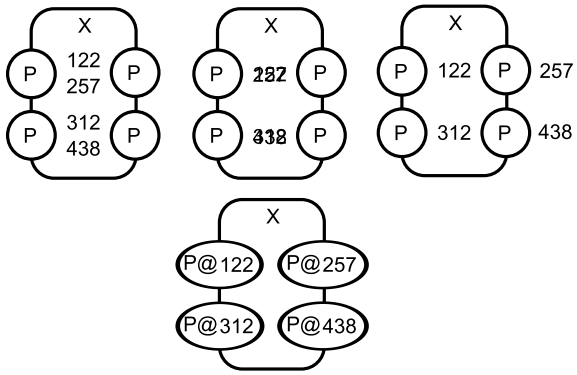

Figure 2.4: *(Upper part) Examples of incorrect* state variables. *(Lower part) Correct version.*

2.3.3 Glyph: *Clone marker*

If an *EPN* is duplicated on a map, it is necessary to indicate this fact by using the *clone marker* auxiliary unit. 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 *EPN* can be found in the map (or in a submap; see Section 2.7.1). 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 *EPN* belongs to a single compartment. If two glyphs labelled "X" are located in two different compartments, such as ATP in cytosol and ATP in mitochondrial lumen, they represent different *EPNs*, and therefore do not need to be marked as cloned.

2.3.3.1 Simple clone marker

As mentioned above, the *simple clone marker* is the unlabeled version of the *clone marker*. See below for the labeled version.

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 2.5 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.

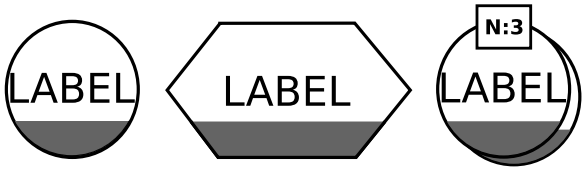

Figure 2.5: *The Process Description glyph for* simple clone marker *applied to a* simple chemical*, an* phenotype *and a* multimer *of* simple chemicals.

2.3.3.2 Labeled clone marker

Unlike the *simple clone marker*, the *labeled clone marker* 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 stateful *EPNs*, because these can have a large number of state variables displayed and therefore may be difficult to visually identify as being identical.

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.

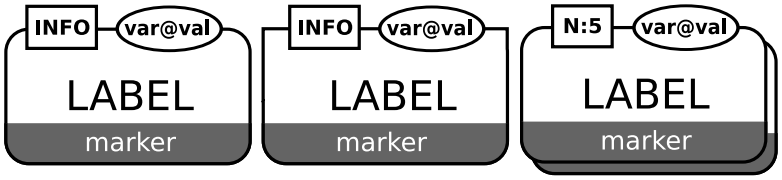

Figure 2.6: *The Process Description glyph for* labeled clone marker *applied to a* macromolecule*, a* nucleic acid feature *and a* multimer *of* macromolecules*.*

Figure 2.7 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.

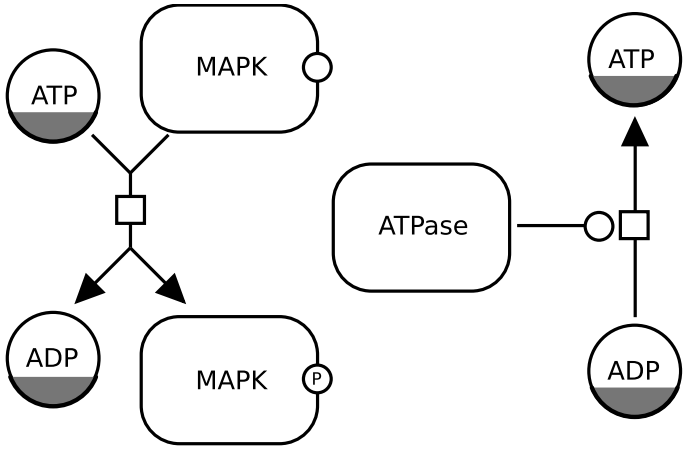

Figure 2.7: *An example of using cloning, here for the species ATP and ADP.*

2.4 Entity pool nodes

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.

The Process Description contains six glyphs representing classes of material entities: *unspecified entity* (Section 2.4.1), *simple chemical* (Section 2.4.2), *macromolecule* (Section 2.4.3), *nucleic acid feature* (Section 2.4.4), *multimer* (Section 2.4.5) and *complex* (Section 2.4.6). (Specific types of macromolecules, such as protein, RNA, DNA, polysaccharide, and specific simple chemicals are not defined by Process Description but may be part of future levels of SBGN.) In addition to the material entities, Process Description represents three conceptual entities: *source*, *sink* (Section 2.4.7), and *perturbing agent* (Section 2.4.8). Material and conceptual entities can optionally carry auxiliary units such as *units of information* (Section 2.3.1), *state variables* (Section 2.3.2) and *clone markers* (Section 2.3.3).

2.4.1 Glyph: *Unspecified entity*

The simplest type of EPN is the *unspecified entity*: 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 *unspecified entity* glyph 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.)

SBO Term:

SBO:0000285 ! material entity of unspecified nature

Container:

An *unspecified entity* is represented by an elliptic container, as shown in Figure 2.8. Note that this must remain an ellipse to avoid confusion with the Simple Chemical glyph, which is a circle (c.f. 2.4.2).

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.

Auxiliary items:

An *unspecified entity* may carry a *clone marker* (Section 2.3.3).

Figure 2.8: *The Process Description glyph for* unspecified entity.

2.4.2 Glyph: *Simple chemical*

A simple chemical in SBGN is defined as the opposite of a macromolecule (Section 2.4.3): it is a chemical compound that is *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.

SBO Term:

SBO:0000247 ! simple chemical

Container:

A *simple chemical* is represented by a circular container, as depicted in Figure 2.9 on the following page. To avoid confusion with the Unspecified Entity (2.4.1), 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:

A *simple chemical* may be decorated with one or more *units of information* (Section 2.3.1). A particular *unit of information* describes the material type. A *simple chemical* may also carry a *clone marker* (Section 2.3.3).

Figure 2.9: *The Process Description glyph for* simple chemical.

2.4.3 Glyph: *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.)

SBO Term:

SBO:0000245 ! macromolecule

Container:

A macromolecule is represented by a rectangular container with rounded corners, as illustrated in Figure 2.10 on the next page.

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.

Auxiliary items:

A *macromolecule* can carry state variables that can add information about its state (Section 2.3.2). The state of a macromolecule is therefore defined as the vector of all its state variables.

A *macromolecule* can also carry one or several *units of information* (Section 2.3.1). The units of information can characterize a domain, such as a binding site. Particular *units of information* are available for describing the material type (Section 2.2.1) and the conceptual type (Section 2.2.2) of a macromolecule.

A *macromolecule* may also carry a *clone marker* (see Section 2.3.3)

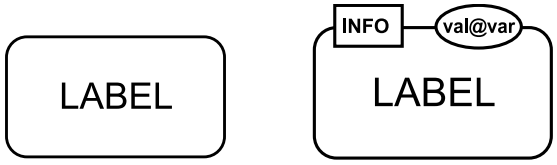

Figure 2.10: *The Process Description glyph for* macromolecule*, shown plain and unadorned on the left, and with with an additional state variable and a unit of information on the right.*

2.4.4 Glyph: *Nucleic acid feature*

The *Nucleic acid feature* construct in SBGN is meant to represent 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 *units of information* are often important for making the purpose clear to the reader of a map.

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 2.11. 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.

Auxiliary items:

A *nucleic acid feature* can carry state variables (Section 2.3.2) that add information about its precise state. The state of a *nucleic acid feature* is therefore defined as the vector of all its state variables.

A *nucleic acid feature* can also carry one or several *units of information* (Section 2.3.1). These can characterize a *nucleic acid feature*'s domain, such as a binding site, or an exon. Particular *units of information* carry the material type (Section 2.2.1) and the conceptual type (Section 2.2.2) of the *nucleic acid feature*.

A *nucleic acid feature* may also carry a *clone marker* (Section 2.3.3).

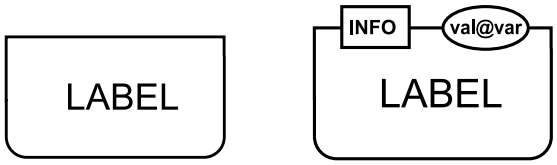

Figure 2.11: *The Process Description glyph for* nucleic acid feature*, shown plain and unadorned on the left and with an additional state variable and a unit of information on the right.*

2.4.5 Glyph: *Multimer*

As its name implies, a multimer is an aggregation of multiple identical or pseudo-identical entities held together by non-covalent bonds (Thus, they are distinguished from polymers by

the fact that the later involve covalent bonds). Here, *pseudo-identical* refers to the possibility that the entities differ chemically but retain some common global characteristic, such as a structure or function, and so can be considered identical within the context of the SBGN Process Description. An example of this are the homologous subunits in a hetero-oligomeric receptor. SBGN Process Description accepts multimers of *simple chemical* (Section 2.4.2), *macromolecule* (Section 2.4.3), *nucleic acid feature* (Section 2.4.4) or *complex* (Section 2.4.6).

SBO Term:

Macromolecule	SBO:0000420 ! multimer of macromolecules
Complex	SBO:0000418 ! multimer of complexes
Nucleic Acid Feature	SBO:0000419 ! multimer of informational molecule segments
Simple Chemical	SBO:0000421 ! multimer of simple chemicals

Container:

A *multimer* is represented by two identical containers shifted horizontally and vertically and stacked one on top of the other. Figure 2.12 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:

A *multimer* can carry state variables that can add information about its state (Section 2.3.2). The state of a multimer is therefore defined as the vector of all its state variables. Note that a *state variable* carried by a multimer actually applies to each of the constituent monomers individually. If instead the state variables are meant to apply to the whole multimeric assembly, a *macromolecule* (Section 2.4.3) should be used instead of *multimer*. An assembly containing some state variables applicable to the components, and others state variable applicable to the assembly (for instance opening of a channel and phosphorylation of each of its subunits) should be represented by a *complex* (Section 2.4.6).

A *multimer* can also carry one or several *units of information* (Section 2.3.1). The information can characterize a domain, such as a binding site. Particular *units of information* exist for describing the material type (Section 2.2.1), the conceptual type (Section 2.2.2), and the cardinality (Section 2.2.5) of the multimer. Note that a *unit of information* carried by a multimer actually applies to each of the constituent monomers individually. If instead a *unit of information* should be applicable to the whole multimeric assembly, a *macromolecule* should be used (Section 2.4.3). An assembly containing units of information applying to the components, and others to the assembly should be represented by a *complex* (Section 2.4.6).

A *multimer* may also carry a *clone marker* (Section 2.3.3).

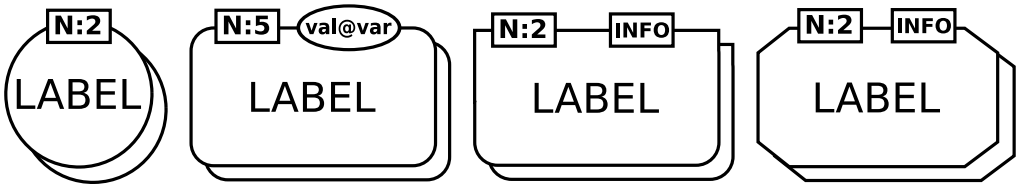

Figure 2.12: *The Process Description glyph for* multimer *with an additional unit of information containing the cardinality.*

2.4.6 Glyph: *Complex*

A *complex* node represents a biochemical entity composed of other biochemical entities, whether macromolecules, simple chemicals, multimers, or other complexes. The resulting entity may have its own identity, properties and function in an SBGN map.

SBO Term:

SBO:0000253 ! non-covalent complex

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. The label box has to be attached to the midway between the border of the complex's container box and the border of the components' container boxes.

Auxiliary items:

A *complex* can carry state variables (see Section 2.3.2). 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 2.3.1). Those units of information can characterize a domain, such as a binding site. Particular *units of information* carry the material type and the conceptual type of the macromolecules. A *complex* may carry a *clone marker* (see Section 2.3.3).

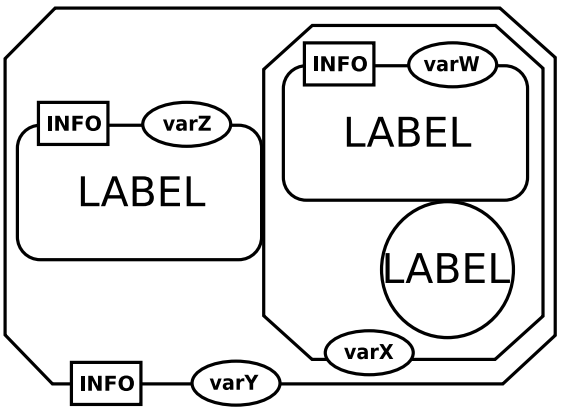

Figure 2.13: *An example Process Description glyph for* complex.

2.4.7 Glyph: *Source* and *Sink*

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 two glyphs that use the same symbol for explicitly representing the involvement of an unspecified source or sink. 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 these sources or sinks are conceptually outside the scope of the map. The reason that we regard the *Source* and *Sink* as different glyphs is that they have different syntax and semantics (the former only connects to a *Consumption* arc and the latter a *Production* arc). This is mainly an issue for software tools and those mapping to and from SBGN from other notations of exchange formats.

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.

SBO Term:

SBO:0000291 ! empty set

Container:

A *source* or *sink* is 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 (∅). Figure 2.14 illustrates this. The symbol should be linked to one and only one edge in a map.

Label:

A *source* or *sink* does not carry any labels.

Auxiliary items:

A *source* or *sink* does not carry any auxiliary items.

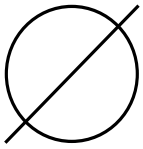

Figure 2.14: *The* source *and* sink *glyphs.*

2.4.8 Glyph: *Perturbing agent*

Biochemical networks can be affected by external influences. Those influences can be 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 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.

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 2.15 on the next page.

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.

Auxiliary items:

A *perturbing agent* may carry a *clone marker* (Section 2.3.3).

A *perturbing agent* can optionally carry one or more *units of information* (Section 2.3.1). A controlled vocaulary is available to describing the physical characteristic of the perturbing agent (see Section 2.2.4).

Figure 2.15: *The Process Description glyph for* perturbing agent.

2.4.9 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 2.16 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*.

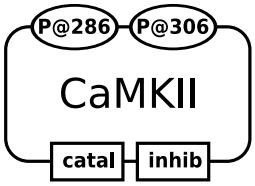

Figure 2.16: *An example representation of calcium/calmodulin kinase II.*

Figure 2.17 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.

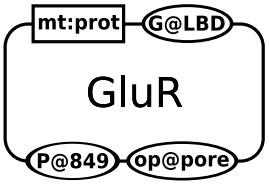

Figure 2.17: *An example of a glutamate receptor in the open state.*

2.5 Referring to other Nodes

Reference nodes handle links or relationships between elements of a map and sub-map. At present there is only one reference glyph, *tag*, which can be used in a map refered to by a

submap (Section 2.7.1) or as an auxilary unit on the *submap*. The *clone marker* can also provide additional reference mechanisms and is discussed below (Section 2.3.3).

2.5.1 Glyph: *Tag*

A *tag* is a named handle, or reference, to another EPN (Section 2.4) or compartment (Section 2.6.1). *Tags* are used to identify those elements in *submaps* (Section 2.7.1).

SBO Term:
Not applicable.

Container:
A *tag* is represented by a rectangle fused to an empty arrowhead, as illustrated in Figure 2.18. 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 2.18: *The Process Description glyph for* tag.

2.6 Defined sets of entity Pool nodes

2.6.1 Glyph: *Compartment*

A compartment is a logical or physical structure that contains entity pool nodes. An EPN can only belong to one compartment. Therefore, the "same" biochemical species located in two different compartments are in fact two different pools.

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 2.3.1. The center of the bounding box of a *unit of information* is located on the mid-line of the border of the compartment.

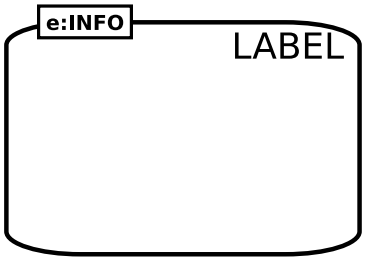

Figure 2.19: *The Process Description glyph for* compartment.

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 2.20 shows two semantically equivalent placement of compartments:

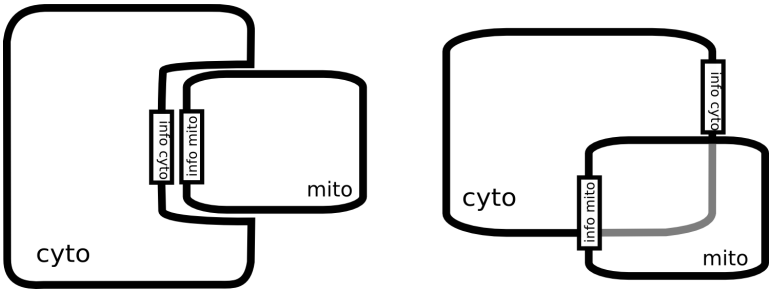

Figure 2.20: *Overlapped compartments are permitted, but the overlap does not imply containment.*

Overlapped (hidden) part of the compartment should not contain any object which could be covered by an overlapping compartment. Figure 2.21 illustrates the problem using an incorrect map.

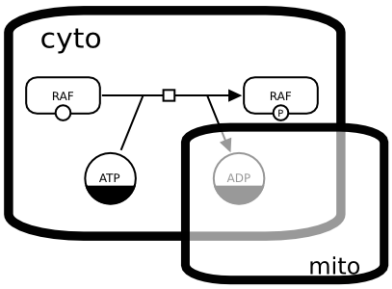

Figure 2.21: *Example of an* ***incorrect*** *map. Overlapped compartments must not obscure other objects.*

2.7 Encapsulation

2.7.1 Glyph: *Submap*

A *submap* is used to encapsulate processes (including all types of nodes and edges) within one glyph. The *submap* hides its content to the users, and display only input terminals (or ports), linked to *EPNs* (Section 2.4). A *submap* is not equivalent to an *omitted process* (see Section 2.8.2). 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*.

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 2.4). In the unfolded view, exposing the internal structure of the *submap*, a set of *tags* point to the corresponding internal *EPNs* Section 2.4.

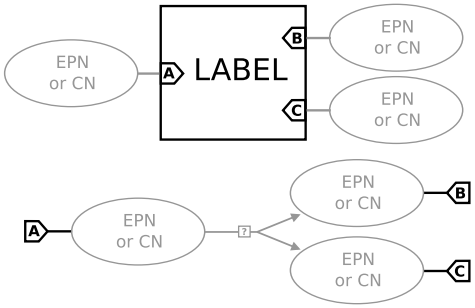

Figure 2.22: *The Process Description glyph for* submap. *(Upper part) folded submap. (Lower part) content of the submap.*

Figure 2.23 on the next page 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.

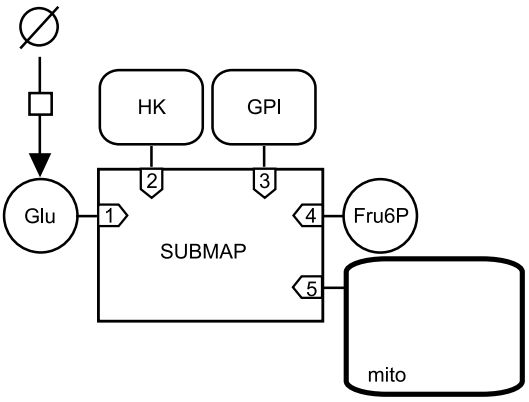

Figure 2.23: *Example of a submap with contents elided.*

The map in Figure 2.24 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*.

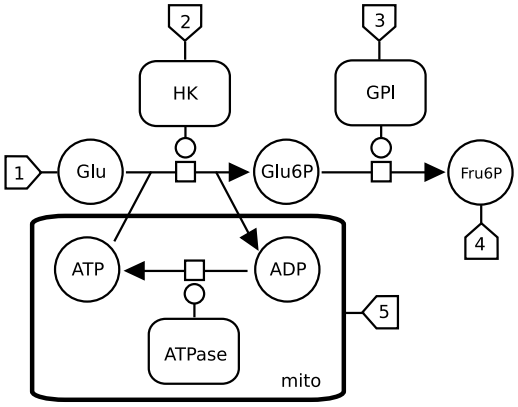

Figure 2.24: *Example of an unfolded submap. The unfolded submap corresponds to the folded submap of Figure 2.23.*

2.8 Process nodes

Process nodes represent processes that transform one or several entity pools into one or several entity pools, identical or different. SBGN Process Description Level 1 defines a generic *process* (Section 2.8.1), as well as five more specific ones: the *omitted process* (Section 2.8.2), the *uncertain process* (Section 2.8.3), the *association* (Section 2.8.4), the *dissociation* (Section 2.8.5), and the *phenotype* (Section 2.8.6). In future levels of the SBGN Process Description language, more processes may be defined. (One can even envision the development of a controlled vocabulary of processes, as is done now for *EPNs*; see Section 2.2.)

2.8.1 Glyph: *Process*

A process transforms a set of entity pools (represented by *EPNs* in SBGN Process Description Level 1) into another set of entity pools.

SBO Term:

SBO:0000375 ! process

Origin:

One or several *consumption* arcs (Section 2.9.1) or one or several *production* arcs (Section 2.9.2).

Target:

One or several *production* arcs (Section 2.9.2).

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 2.9.1) and production (Section 2.9.2) arcs are linked to the extremities of those connectors. The modulatory arcs (Section 2.9) point to the other two sides of the box. A *process* connected to *production* arcs on opposite sides is a reversible process.

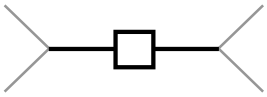

Figure 2.25: *The Process Description glyph for* process.

A process is the basic process node in SBGN. It 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 2.9.1) or *production* (Section 2.9.2) 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 3.5.7).

The example in Figure 2.26 illustrates the use of a *process* node to represent the phosphorylation of a protein in a Process Description.

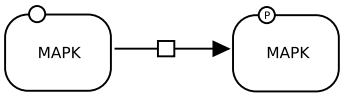

Figure 2.26: *Phosphorylation of the protein MAP kinase.*

The example in Figure 2.27 on the following page illustrates the use of a *process* node to represent a reaction between two reactants that generates three products.

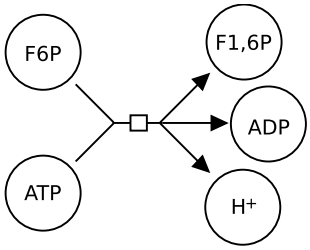

Figure 2.27: *Reaction between ATP and fructose-6-phosphate to produce fructose-1,6-biphosphate, ADP and a proton.*

The example in Figure 2.28 illustrates the use of a *process* node to represent a translocation. The large round-cornered rectangle represents a compartment border (see Section 2.6.1).

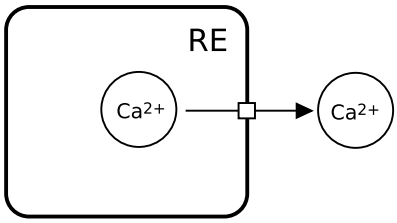

Figure 2.28: *Translocation of calcium ion out of the endoplasmic reticulum. Note that the* process *does not have to be located on the boundary of the* compartment*. A* process *is not attached to any* compartment*.*

The example in Figure 2.29 illustrates the use of a *process* node to represent the reversible opening and closing of an ionic channel in a Process Description.

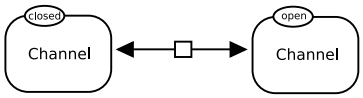

Figure 2.29: *Reversible opening and closing of an ionic channel.*

When such a reversible process is asymmetrically modulated, it must be represented by two different processes in a Process Description. Figure 2.30 on the next page 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.

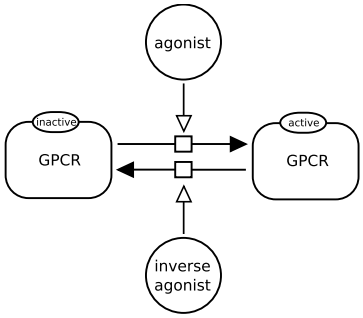

Figure 2.30: *The reversible activation of a G-protein coupled receptor.*

The example in Figure 2.31 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.

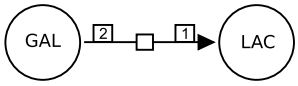

Figure 2.31: *Conversion of two galactoses into a lactose.*

2.8.2 Glyph: *Omitted process*

Omitted processes are processes that are known to exist, but are omitted from the map for the sake of clarity or parsimony. A single *omitted process* can represent any number of actual processes. The *omitted process* is different from a *submap*. While a *submap* references to an explicit content, that is hidden in the main map, the *omitted process* does not "hide" anything within the context of the map, and cannot be "unfolded".

SBO Term:

SBO:0000397 - omitted process.

Origin:

One or several *consumption* arcs (Section 2.9.1) or one or several *production* arcs (Section 2.9.2).

Target:

One or several *production* arcs (Section 2.9.2).

Node:

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

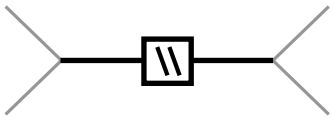

Figure 2.32: *The Process Description glyph for* omitted process.

2.8.3 Glyph: *Uncertain process*

Uncertain processes are processes that may not exist. A single *uncertain process* can represent any number of actual processes.

SBO Term:
SBO:0000396 ! uncertain process.

Origin:
One or several *consumption* arcs (Section 2.9.1) or one or several *production* arcs (Section 2.9.2).

Target:
One or several *production* arcs (Section 2.9.2).

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

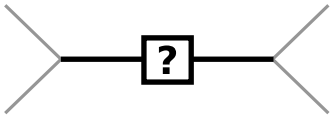

Figure 2.33: *The Process Description glyph for an* uncertain process.

2.8.4 Glyph: *Association*

The association between one or more *EPNs* represents the non-covalent binding of the biological objects represented by those *EPNs* into a larger complex.

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

Origin:
One or more *consumption* arcs (Section 2.9.1).

Target:
One *production* arc (Section 2.9.2).

Node:
An *association* between several entities is represented by a filled disc linked to two connectors, small arcs attached on point separated by 180 degrees. The consumption (Section 2.9.1) and production (Section 2.9.2) arcs are linked to the extremities of those connectors.

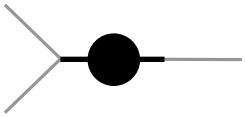

Figure 2.34: *The Process Description glyph for* association.

The example in Figure 2.35 on the next page illustrates the association of cyclin and CDC2 kinase into the Maturation Promoting Factor.

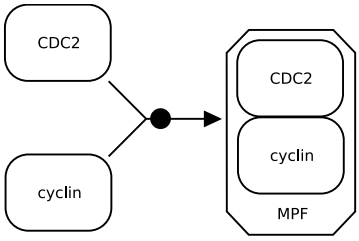

Figure 2.35: *Association of cyclin and CDC2 kinase into the Maturation Promoting Factor.*

Figure 2.36 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.

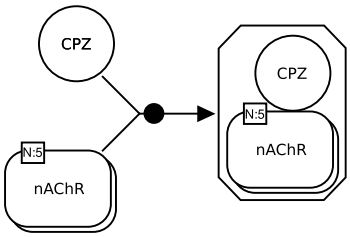

Figure 2.36: *The association of a pentameric macromolecule with a simple chemical in an unnamed complex.*

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). Figure 2.37 gives an example of this, using the formation of hemoglobin.

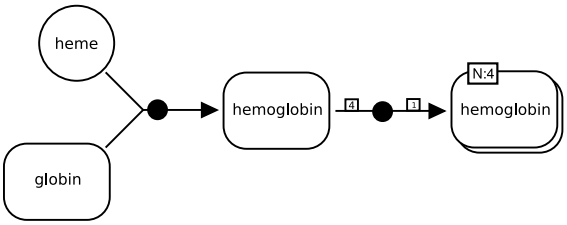

Figure 2.37: *Formation of hemoglobin.*

2.8.5 Glyph: *Dissociation*

The dissociation of an *EPN* into one or more *EPNs* represents the rupture of a non-covalent binding between the biological entities represented by those *EPNs*.

SBO Term:

SBO:0000180 ! dissociation.

Origin:

One *consumption* arc (Section 2.9.1).

Target:

One or more *production* arc (Section 2.9.2).

Node:

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 2.9.5). Moreover, the existence of two circles reminds the dissociation, by contrast with the filled disc of the *association* (Section 2.8.4).

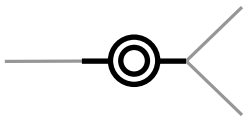

Figure 2.38: *The Process Description glyph for* dissociation.

The example in Figure 2.39 illustrates the dissociation of the small and large ribosomal subunits from a messenger RNA.

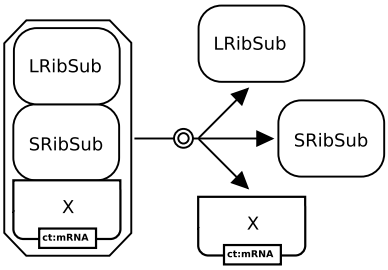

Figure 2.39: *Dissociation of the small and large ribosomal subunits from a messenger RNA.*

2.8.6 Glyph: *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* glyph.

SBO Term:

SBO:0000358 ! phenotype

Container:

A *phenotype* is represented by an elongated hexagon, as illustrated in Figure 2.40.

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.

Auxiliary items:

A *phenotype* may carry a *clone marker* (Section 2.3.3).

Figure 2.40: *The Process Description glyph for* phenotype.

2.9 Arcs

Arcs are lines that link *EPNs* and *PNs* together. The symbols attached to their extremities indicate their semantics.

2.9.1 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 2.4).

Target:
Any *process node* (Section 2.8).

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

A cardinality label may be associated with *consumption* (Section 2.9.1) or *production* (Section 2.9.2) 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.

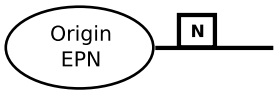

Figure 2.41: *The Process Description glyph for* consumption.

2.9.2 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 2.8).

Target:
Any *EPN* (Section 2.4).

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.

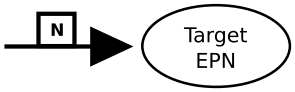

Figure 2.42: *The Process Description glyph for* production.

Figure 2.43 illustrates the use of consumption/production arc cardinality labels to represent the stoichiometry of a process.

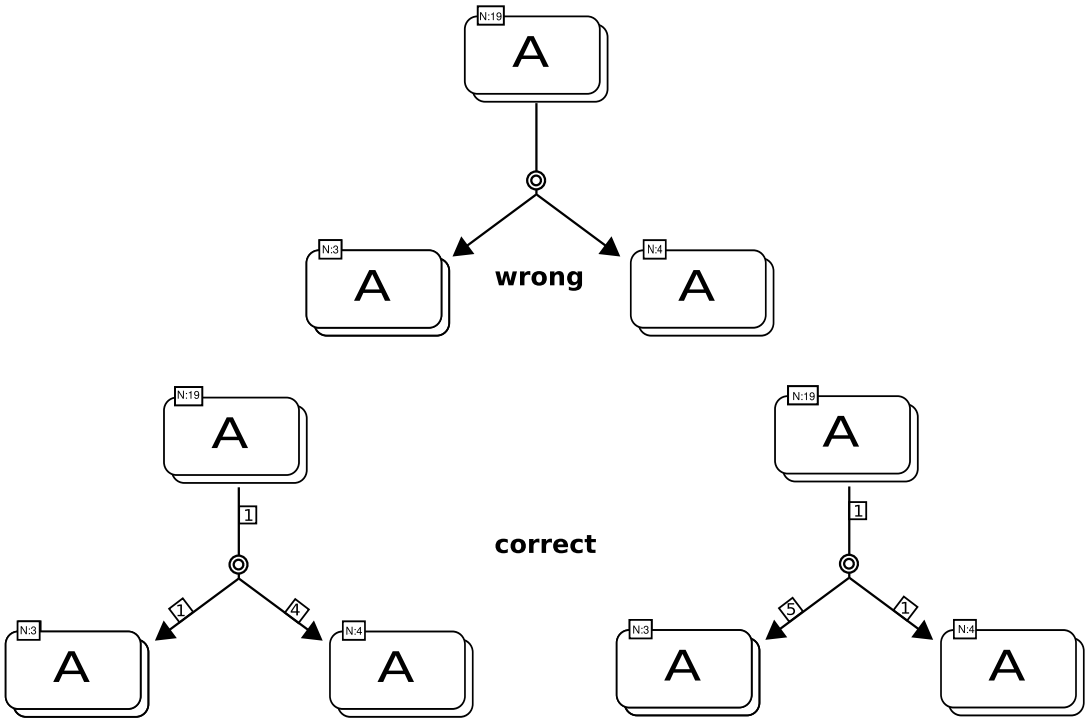

Figure 2.43: *Cardinality for production arcs.*

2.9.3 Glyph: *Modulation*

A modulation 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. A *modulation* can also be used when one does not know the precise direction of the effect.

SBO Term:

SBO:0000168 ! control.

Origin:

Any *EPN* (Section 2.4) or any *logical operator* (Section 2.10).

Target:

Any *process node* (Section 2.8).

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

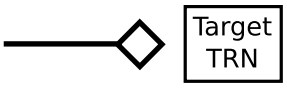

Figure 2.44: *The Process Description glyph for* modulation.

Figure 2.45 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.

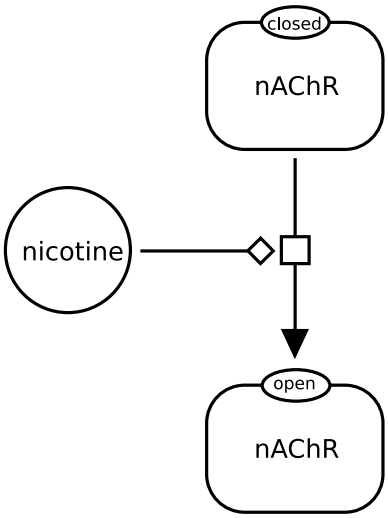

Figure 2.45: *Modulation of nicotinic receptor opening by nicotine.*

2.9.4 Glyph: *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 *catalysis* exists independently in SBGN, see Section 2.9.5.

SBO Term:
SBO:0000170 ! stimulation.

Origin:
Any *EPN* (Section 2.4) or any logical operator (Section 2.10).

Target:
Any *process node* (Section 2.8).

End point:
The target extremity of a *stimulation* carries an empty arrowhead.

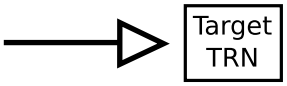

Figure 2.46: *The Process Description glyph for* stimulation.

2.9.5 Glyph: *Catalysis*

A catalysis is 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.

SBO Term:
SBO:0000172 ! catalysis.

Origin:
Any *EPN* (Section 2.4) or any *logical operator* (Section 2.10).

Target:
Any *process node* (Section 2.8).

Node:
The target extremity of a *catalysis* carries an empty circle.

Figure 2.47: *The Process Description glyph for* catalysis.

2.9.6 Glyph: *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.

SBO Term:
SBO:0000169 ! inhibition.

Origin:
Any *EPN* (Section 2.4) or any *logical operator* (Section 2.10).

Target:
Any *process node* (Section 2.8).

Node:
The target extremity of an *inhibition* carries a bar perpendicular to the arc.

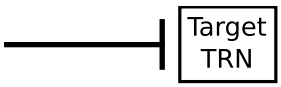

Figure 2.48: *The Process Description glyph for* inhibition.

2.9.7 Glyph: *Necessary stimulation*

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.

SBO Term:
SBO:0000171 ! necessary stimulation.

Origin:
Any *EPN* (Section 2.4) or any *logical operator* (Section 2.10).

Target:
Any *process node* (Section 2.8).

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.

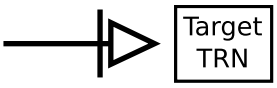

Figure 2.49: *The Process Description glyph for* Necessary Stimulation.

The example in Figure 2.50 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.)

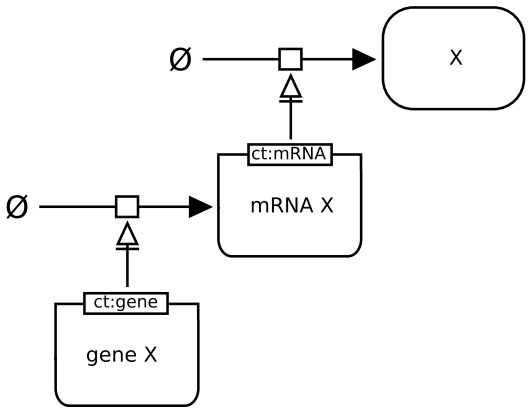

Figure 2.50: *The creation of a messenger RNA X triggered by the gene X.*

The example in Figure 2.51 on the next page 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.

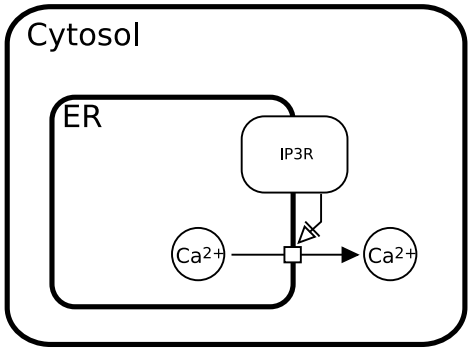

Figure 2.51: *The transport of calcium ions out of the endoplasmic reticulum into the cytosol. Note that IP3R crosses both compartment boundaries. This is allowed, but the Macromolecule should only belong to one of the compartments see section C.1 for more discussion of this issue.*

2.9.8 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 2.4) or *logical operator* (Section 2.10).

Target:
Any *logical operator* (Section 2.10).

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

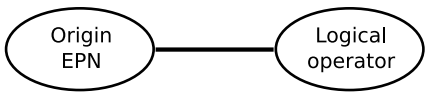

Figure 2.52: *The Process Description glyph for* logic arc.

2.9.9 Glyph: *Equivalence arc*

Equivalence arc is the arc used to represent the fact that all entities marked by a *tag* are equivalent.

SBO Term:
Not applicable.

Origin:
Any *EPN* (Section 2.4).

Target:
Tag (Section 2.5.1).

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

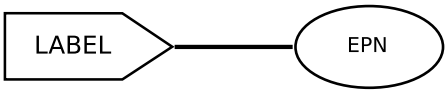

Figure 2.53: *The Process Description glyph for* Equivalence arc.

2.10 Logical operators

2.10.1 Glyph: *And*

The glyph *and* is used to denote that all the *EPNs* linked as input are necessary to produce the output.

SBO Term:
SBO:0000173 ! and.

Origin:
More than one *EPN* (section 2.4) or *logical operator* (section 2.10).

Target:
One modulation (section 2.9.3), stimulation (section 2.9.4), catalysis (section 2.9.5), inhibition (section 2.9.6) or necessary stimulation (section 2.9.7) arc.

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

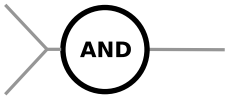

Figure 2.54: *The Process Description glyph for* and. *Only two inputs are represented, but more would be allowed.*

2.10.2 Glyph: *Or*

The glyph *or* is used to denote that any of the *EPNs* linked as input is sufficient to produce the output.

SBO Term:
SBO:0000174 ! or.

Origin:
More than one *EPN* (section 2.4) or *logical operator* (section 2.10).

Target:
One modulation (section 2.9.3), stimulation (section 2.9.4), catalysis (section 2.9.5), inhibition (section 2.9.6) or necessary stimulation (section 2.9.7) arc.

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

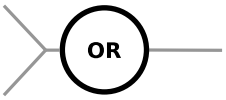

Figure 2.55: *The Process Description glyph for* or. *Only two inputs are represented, but more would be allowed.*

2.10.3 Glyph: *Not*

The glyph *not* is used to denote that the *EPN* linked as input cannot produce the output.

SBO Term:
SBO:0000238 ! not.

Origin:
One *EPN* (section 2.4) or *logical operator* (section 2.10).

Target:
One modulation (section 2.9.3), stimulation (section 2.9.4), catalysis (section 2.9.5), inhibition (section 2.9.6) or necessary stimulation (section 2.9.7) arc.

Node:
Not is represented by a circle carrying the word "NOT".

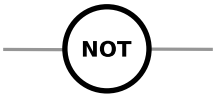

Figure 2.56: *The Process Description glyph for* not.

Chapter 3

Process Description Language Grammar

3.1 Overview

In this chapter, we describe how the glyphs of SBGN Process Description Level 1 can be combined to make a valid SBGN Process Description map. To do this, we must at the very least define what glyphs can be connected to each other. This is called syntax. Next, we must define rules over and above connection rules, such as whether duplicate symbols are permitted. In addition, we must define what the notation "means" — how does it represent a biological pathway? This is semantics, and it is essential if a reader is to understand a Process Description map without external help, and a writer is to create one that reflects his understanding of a biological system.

In this section we start off by describing the concepts of the Process Description language. Next a detailed description of the syntax is provided followed by a description of the syntactic rules of the notation.

3.2 Concepts

The SBGN Process Description language is more than a collection of symbols. It is a visual language that uses specific abstractions to describe the biological processes that make up a quantitative model, a signalling pathway or a metabolic network. This abstraction is the semantics of SBGN, and to describe it requires more than a definition of the symbols and syntax of the language. We first need to define the abstractions we are using.

A Process Description in SBGN describes biological processes involving biological entities. An *Entity pool node* (Section 2.4), such as a molecule, is converted into other *EPNs* via a *process*.

It may be convenient to think of a SBGN Process Description as describing the flow of a fluid. In this analogy, each *EPN* represents a tank of fluid, which may be emptied via a pipe (the consumption arc, Section 2.9.1) that is connected to a valve (the process node, Section 2.8), which in turn is connected to other pipes (the production arcs, Section 2.9.2) that fill other tanks (*EPNs*). A valve can be controlled in relation to the amount of fluid in another tank. This precise nature of the relationship between valve opening and amount of fluid determines the nature of the modulation.

Finally, the system needs a source of fluid from one or more external sources (*source*, section Section 2.4.7) and somewhere for it to drain away (the *sink*).

3.3 The conceptual model

In order to formalize the conceptual representation described above and the semantic rules that we will describe later, we have developed a UML Domain Object Model of SBGN. This defines what would otherwise be vaguely defined in the SBGN specification, and enables us to define clearly what is a valid SBGN map.

The model is separated into two layers. The Visual Layer (Figure 3.1 on the following page) directly maps to the glyphs used to visualise SBGN and reflects the syntax of the notation. The Conceptual Layer describes what is "meant" by the visual SBGN representation (Figure 3.2 on page 40). It is important in defining concepts such as Entity types and state variables that are implicit in the Visual Layer.

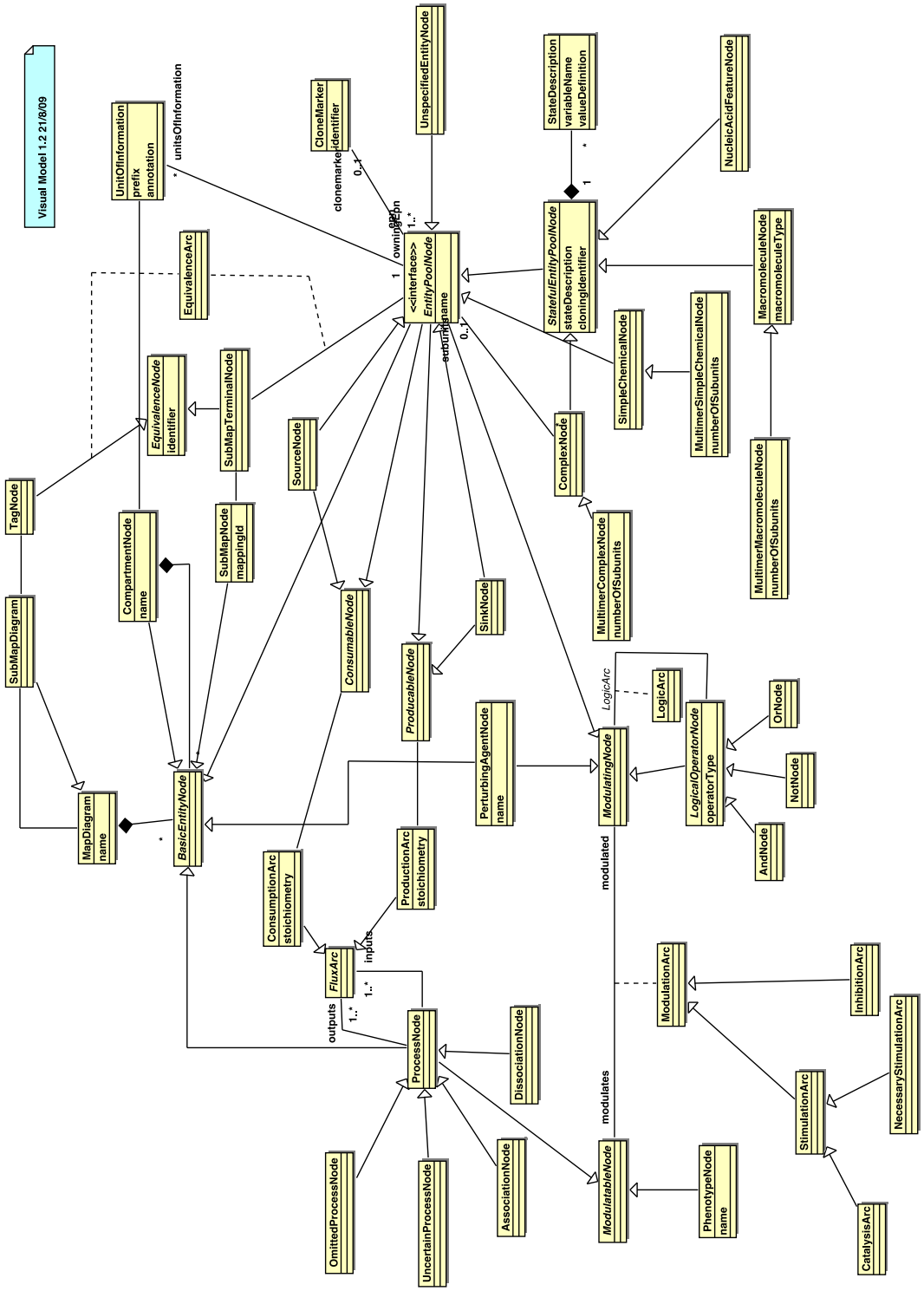

Figure 3.1: *UML model of SBGN visual representation. Class names shown in italics are abstract classes and do not correspond to a glyph. Classes in non-italicised script with the suffixes "Node" and "Arc" correspond to node and arc glyphs respectively.*

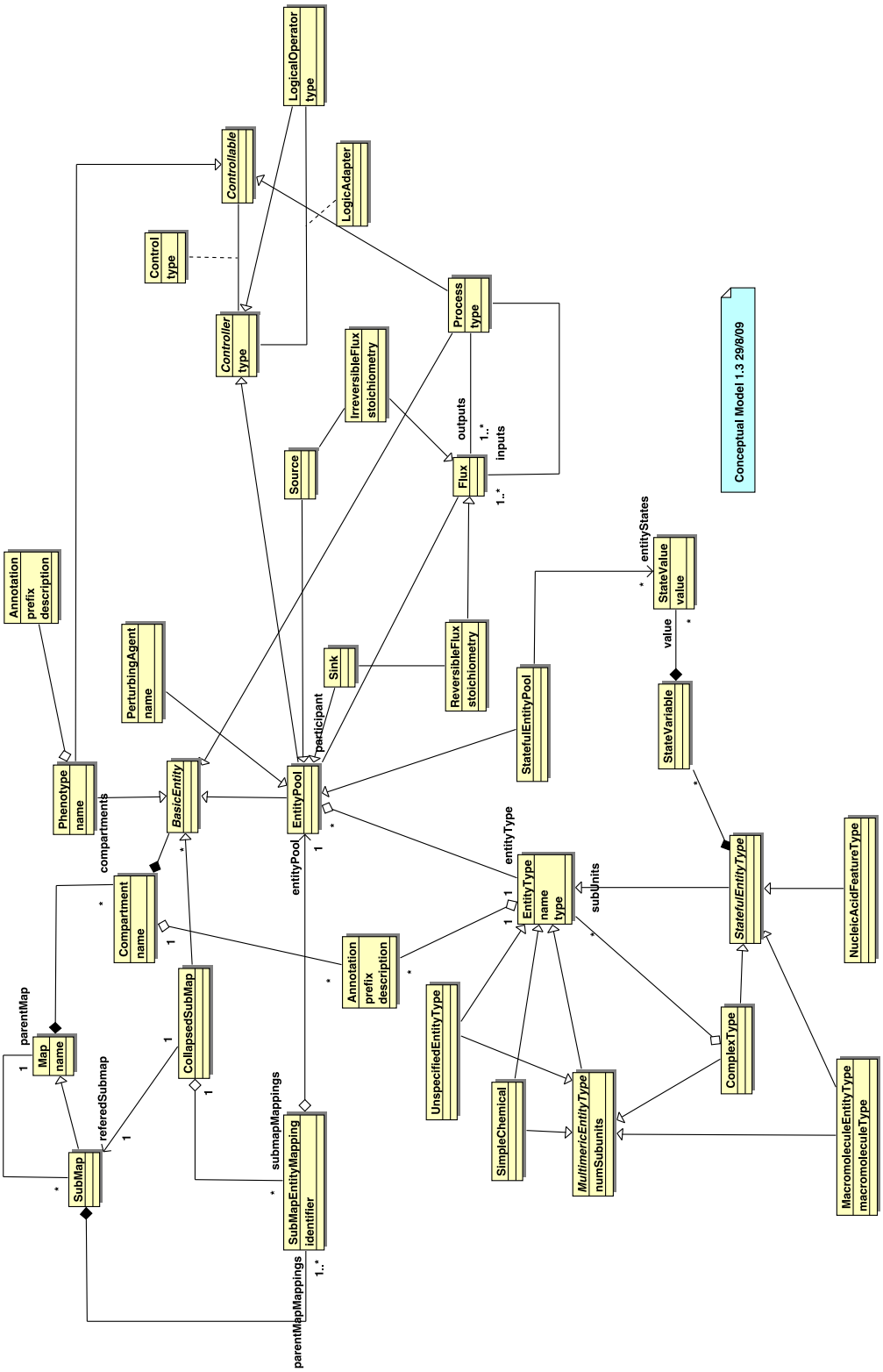

Figure 3.2: *UML model of the SBGN conceptual representation.*

The classes shown in the UML model are described in more detail in the tables below. In particular the tables define the properties that uniquely identify each component of the notation. This is important in defining the rules for duplication and the submap interface later.

Table 3.1: *A data dictionary for the Visual Layer of the SBGN Conceptual Model. Attributes that uniquely identify a class are followed by (I) and required attributes by (R): it is implied that identifying attributes are also required. The default cardinality of an attribute is one (or zero or one if it is not required) unless otherwise specified. A standard notation for cardinality is used, for example {0 ... *} should be read zero, one or many. The Glyph column states the Glyph the class maps to, unless the class is purely conceptual and so defined as Abstract.*

Visual Layer Data Dictionary				
Name	Attributes		Glyph	Comment
	Name	Description		
Map	name (I)	Name of the map.	N/A	This is the visualisation of a SBGN map.
Submap	mainMapName (I) name (I)	Name of the main map this is a submap of. Name of the submap.	N/A	A SBGN map used to define a SBGN submap
BasicEntityNode			Abstract	The most fundamental node type. Any node that can be drawn on a Map inherits from this.
ConsumableNode			Abstract	Nodes that are connected to the start of a consumption arc.
ProducableNode			Abstract	Nodes that linked to the end of a production arc.
ModulatableNode	modulatingNodes (I) {0 ... *}	Any number of nodes that modulate this one.	Abstract	Nodes representing a process that can be modulated and so are linked to the end of a modulation arc.
ModulatingNode			Abstract	Nodes that can modulate modulatable nodes. They are linked to the origin of a modulating arc.
CompartmentNode	name (I) unitsOfInformation {0 ... *}	Name of compartment being defined. At present the name of the compartment is not restricted, i.e. a controlled vocabulary is not used. any number of Units of Information.	Compartment	
EntityPoolNode (EPN)	name (I) compartment (I) unitsOfInformation {0 ... *}	The name of the entity. If no compartment is specified then this is implicitly defined as the "default" compartment. Any number of Units of Information.	Abstract	Nodes that inherit from this type are generally chemical species that can be produced, consumed or transformed in chemical processes. It refers to a population (or pool) of such species.
StatefulEntityPool-Node	As EPN stateDescriptions (I) {0 ... *}	 Any number of StateDescriptions associated with this EPN.	Abstract	All EPN that inherit from this class may have state variables assigned to them.
SimpleChemicalNode	As EPN		SimpleChemical	
UnspecifiedEntity-Node	As EPN		UnspecifiedEntity	

continued on next page

continued from previous page

Name	Attributes		Glyph	Comment
	Name	Description		
SourceNode			Source/Sink	The SourceNode can only be associated with *one* consumption arc. Note that this shares the same Glyph as SinkNode.
SinkNode			Source/Sink	The SinkNode can only be associated with *one* production arc. Note that this shares the same Glyph as SourceNode.
PhenotypeNode	name (I)	Name of the phenotype.	Phenotype	
Perturbing Agent Node	name (I)	Name of the perturbing agent.	Perturbing Agent	
MacromoleculeNode	As StatefulEntityPoolNode macromoleculeType (I)	 The type of the macromolecule. Currently there are no constraints on permitted types.	Macromolecule	
NucleicAcidFeature-Node	As StatefulEntityPoolNode		NucleicAcid-Feature	
ComplexNode	As StatefulEntityPoolNode subunits {0 . . . *} subunitStates (I) {0 . . . *}	 A list of EPNs that together compose the complex. The set of state descriptions of any Subunits that are StatefulEntityPoolNodes.	Complex	
Multimer-MacromoleculeNode	As MacromoleculeNode numberOfSubunits (I)	 The number of identical subunits of this macromolecule.	Multimer-Macromolecule	Note that all subunits have the *same* StateDescription values.
MultimerSimple-ChemicalNode	As SimpleChemicalNode numberOfSubunits (I)	 The number of identical subunits of this simple chemical.	MultimerSimple-Chemical	
MultimerComplex-Node	As ComplexNode numberOfSubunits (I)	 The number of identical subunits of this complex.	MultimerComplex	Note that all subunits have the *same* StateDescription values.
UnitOfInformation	owningEPN (I) prefix annotation	Identity of the owning EntityPoolNode (not displayed, but implied by "ownership" of the glyph). Prefix (I) Annotation (R)	Unit Of Information	
StateDescription	owningStatefulEPN variable	Identity of the owning StatefulEntityPoolNode (not displayed, but implied by "ownership" of the glyph). The name (identification) of the "state variable" glyph associated a StatefulEntityPoolNode.	State Variables	

continued on next page

continued from previous page

Name	Attributes		Glyph	Comment
	Name	Description		
	value	The definition of a value for the given variable.		
ProcessNode	As ModulatableNode outputNodes (I) {1 . . . *} inputNodes (I) {1 . . . *}	 The OutputNodes linked to the process by a ProductionArc. The InputNodes linked to the process by either a ProductionArc or ConsumptionArc.	Process	
OmittedProcessNode	As Process Node		Omitted Process	
UncertainProcessNode	As Process Node		Uncertain Process	
AssociationNode	As Process Node		Association	
DissociationNode	As Process Node		Dissociation	
LogicOperatorNode	gateType (I) modulatingNodes (I) modulatedNode (I)	The type of boolean gate. Currently one of OR, AND, NOT. The modulating nodes that are the input the Boolean logic operator. They must be linked via a LogicArc. A ModulatableNode connected via a ModulationArc or another LogicOperatorNode connected via a LogicArc.	Abstract	Not a glyph but a placeholder for the common behaviour of all logic gates.
ANDNode	As LogicalOperatorNode		And	
ORNode	As LogicalOperatorNode		Or	
NOTNode	As LogicalOperatorNode		Not	
SubMapNode	subMapName (I) compartment (R)	The name of the submap it is defining. If no compartment is specified then this is implicitly defined as the “default” compartment.	Submap	
TagNode	subMapName (I) identifier (I)	Name of submap that owns tag. A name that uniquely identifies the tag on the submap.	Tag	
SubMapTerminalNode	subMapName (I) identifier (I)	Name of submap that owns labeled terminal. A name that uniquely identifies the terminal on the submap.	SubMapTerminal	
ProductionArc	processNode (I) producableNode (I)	The ProcessNode that this is the output of.	Production	It is assumed that the arc is unidirectional.
ConsumptionArc	consumableNode (I) processNode (I)	The ConsumableNode that is the input to the process.	Consumption	It is assumed that the arc is unidirectional.
ModulationArc	modulatingNode (I)	The ModulatingNode that is the origin of this link.	Modulation	

continued on next page

continued from previous page

Name	Attributes		Glyph	Comment
	Name	Description		
	modulatedNode (I)	The ModulatableNode that is controlled (modulated) by this arc.		
StimulationArc	As ModulationArc		Stimulation	
CatalysisArc	As ModulatioArc		Catalysis	
InhibitionArc	As ModulationArc		Inhibition	
Necessary Stimulation-Arc	As ModulationArc		Inhibition	
LogicArc	modulatingNode (I) logicalOperator (I)	The ModulatingNode that is the origin of this link. The logicalOperatorNode that is the target of this link.	LogicArc	Conceptually this class can be thought of as converting a continuous input (the population of the EntityPool) into a discrete Boolean output (True or False).
EquivalenceArc	Tag (I) EPN (I)	The tag assigned to with the EPN. The EPN assigned to the tag.	Tag	

Table 3.2: *A data dictionary for the Conceptual Layer of the SBGN Model. Attributes that uniquely identify a class are followed by (I) and required attributes by (R): it is implied that identifying attributes are also required. The default cardinality of an attribute is one (or zero or one if it is not required) unless otherwise specified. A standard notation for cardinality is used, for example {0 . . . *} should be read zero, one or many. A mapping to a class in the Visual Layer is provided where appropriate.*

Conceptual Layer Data Dictionary				
Name	Attributes		Description	Mapping to Visual Layer
	Name	Description		
Map	name (I)	A name or identifier for the map.	The biological pathway that is represented by the SBGN map.	Map
SubMap	name (I) mainMapName (I)	The name of the submap. The name of the map that refers to this submap.		SubMap
Compartment	name (I)	The name of the compartment.		CompartmentNode
BasicEntity			Root class of all Entities that can be contained by a Compartment.	BasicEntityNode
Controller			Something that can control a Controllable.	ModulatingNode
Controllable			Something that can be controlled by a Controller	ModulatableNode
EntityType	Name (I) type (I) annotations {0 . . . *}	The name of the EntityType. Type of the Entity. One of: SimpleChemical, UnspeciedEntity, Complex, Macromolecule, and NucleicAcidFeature. Annotation must be associated with the type and be general to all Entity Pool Node. If it were associated with the Species then this would be synonymous with a state description.	This is the type of the EntityPool.	No direct maps to the visual layer. It is an implied attribute of the EntityPoolNode.

continued on next page

continued from previous page

Name	Attributes		Description	Visual Mapping
	Name	Description		
MultimericEntityType	As EntityType cardinality (I)	 The number of subunits in the monomer.	An Entity that can form homo-multimeric complexes. All copies share the same attributes.	
StatefulEntityType	As EntityType stateVariables { 0 . . . * }		An EntityType that can have state variables associated with it.	
ComplexEntityType	As StatefulEntityType cardinality (I) subunits { 0 . . . * }	 from MultimericEntityType. The set of EntityTypes that the complex composed of.		Implicitly defined by a ComplexNode.
MacromoleculeEntity-Type	macromoleculeType (I)	The type of the macromolecule. The permitted type names are not constrained as yet. Suggest types are protein, mRNA, miRNA, tRNA, rRNA, DNA, polysaccaride. See also controlled vocabularies for material type 2.2.1 and conceptual type 2.2.2		Implicitly defined by a MacromoleculeNode and MultimerMacromolecule-Node
StateVariable	name (I)	The name of the state variable.	The state variable is assigned to a StatefulEntityType and can have one or more values assigned to it when used in a StatefulEntityPool.	The name of a StateDescription.
StateValue	value (I)	The value of the state.	A value that can be assigned to a state variable.	The value string or "reg-exp" used in the State-Description.
EntityPool	entityType (I)	The EntityType of the pool.	An instantiation of an EntityType.	EntityPoolNode
StatefulEntityPool	As EntityPool stateValueSets {1 . . . * }	 A set of state values assigned to a the State-Variables of the associated StatefulEntity-Type.		StatefulEntityPoolNode
ComplexEntityPool	As StatefulEntity subunitValues (I) {0 . . . * }	 The set of StateValues for all subunits.	The instantiation of a complex that is defined by it's own state and the states of its subunits.	ComplexNode
Annotation	prefix (I) annotation	A classifier for the annotation. Annotation that is associated with a Entity-Type or Compartment.		UnitOfInformation
Process	processType (I)	One of: Unspecified, Dissociation, Association, Omitted Process, Uncertain Process.		The type is specified by the type of Glyph used: ProcessNode, Dissociation-Node, AssociationNode, OmittedProcessNode, UncertainProcessNode.

continued on next page

continued from previous page

Name	Attributes		Description	Visual Mapping
	Name	Description		
Flux	process (I) participant (I)	The process that this is the output of. The EPN that is involved with the above process.	The flux represent the flow of EPNs to or from a process.	ProductionArc
Reversible Flux	As Flux			ProductionArc
Input	As Flux		Here the flow can only go from the EPN to the Process.	ConsumptionArc
Control	type (R)	The type of the control. One of Modulation, Stimulation, Inhibition, Necessary Stimulation, Catalysis or LogicalOperator.		ModulationArc
LogicalOperator	type (R)	The type of LogicalOperator. One of AND, OR, NOT.		LogicalOperatorNode
LogicAdapter			The LogicalOperator takes a set of Boolean inputs (T or F) from an EntityPool which is a non-Boolean quantity. Conceptually, the adapter converts the continuous quantity into a discrete Boolean quantity.	LogicArc
Perturbing Agent	name	Name of the perturbing agent.		Perturbing Agent Node
Phenotype	name	Name of the phenotype.		PhenotypeNode
Source				SourceNode
Sink				SinkNode
CollapsedSubMap	referredSubMap (I)	Submap that this refers too.	The submap is "collapsed" down to this element, which acts as a placeholder for the submap.	SubMapNode
SubMapEntity-Mapping	identifier (I) entityPool (R)	Identifier in Visual layer to link EntityPool-Nodes in the main map and submap. The EntityPool node that is mapped between the submap and the main map.	A definition of the mapping between the submap and main map for a single EntityPool.	SubMapTerminalNode and TagNode

3.4 Syntax

The syntax of the SBGN Process Description language is defined in the form of an incidence matrix. An incidence matrix has arcs as rows and nodes as columns. Each element of the matrix represents the role of an arc in connection to a node. Input (I) means that the arc can begin at that node. Output (O) indicates that the arc can end at that node. Numbers in parenthesis represent the maximum number of arcs of a particular type to have this specific connection role with the node. Empty cells means the arc is not able to connect to the node.

For simplicity logical operators are treated as process nodes.

3.4.1 Entity Pool Nodes connectivity definition

Arc\EPN	*macromolecule*	*simple chemical*	*unspecified entity*	*multimer*	*complex*	*nucleic acid feature*	*tag*	*source*	*sink*	*perturbing agent*	*submap*
consumption	I	I	I	I	I	I		I			
production	O	O	O	O	O	O			O		
modulation	I	I	I	I	I	I				I	
stimulation	I	I	I	I	I	I				I	
catalysis	I	I	I	I	I					I	
inhibition	I	I	I	I	I	I				I	
necessary stimulation	I	I	I	I	I	I				I	
logic arc	I	I	I	I	I	I					
equivalence arc	I	I	I	I	I	I	O				O

3.4.2 Process Nodes connectivity definition

Arc\PN	*process*	*omitted process*	*uncertain process*	*phenotype*	*association*	*dissociation*	*and*	*or*	*not*
consumption	O	O	O		O	O(1)			
production	I	I	I		I(1)	I			
modulation	O	O	O	O			I(1)	I(1)	I(1)
stimulation	O	O	O	O			I(1)	I(1)	I(1)
catalysis	O	O	O	O			I(1)	I(1)	I(1)
inhibition	O	O	O	O			I(1)	I(1)	I(1)
necessary stimulation	O	O	O	O			I(1)	I(1)	I(1)
logic arc							O	O	O(1)
equivalence arc									

3.4.3 Containment definition

There are two Process Description glyph that allow containment of glyphs: *compartment* and *complex*. The next table describe relationship between Process Description glyphs and these containers. A plus sign means that the element is able to be contained within a container. An empty cell means containment is not allowed.

Glyph\Containers	*complex*	*compartment*
unspecified entity	+	+
simple chemical	+	+
macromolecule	+	+
nucleic acid feature	+	+
multimer	+	+
source	-	+
sink	-	+
perturbing agent	-	+
phenotype	-	+
tag	-	+
complex	+	+
compartment	-	-
submap	-	+
process	-	+
omitted process	-	+
uncertain process	-	+
association	-	+
dissociation	-	+
consumption	-	+
production	-	+
modulation	-	+
stimulation	-	+
catalysis	-	+
inhibition	-	+
necessary stimulation	-	+
logic arc	-	+
equivalence arc	-	+
and	-	+
or	-	+
not	-	+

3.4.4 Syntactic rules

There are additional syntactic rules that must be applied in addition to those defined above.

3.4.4.1 EPNs

1. If *macromolecule* has more than one *state variable*, all *state variables* should have a name;
2. All *state variables* of the *macromolecule* should have different names;
3. *Complex* should consists of different EPNs. If two or more elements of the complex are identical they should be replaced by multimer.
4. An EPN that is a subunit of a *Complex* can only be linked to a modulation arc.

3.4.4.2 Source *and* Sink

1. Only one arc can be attached to these glyphs (degree 1).

3.4.4.3 Process

1. PN should have nonzero number of *production* links.
2. All substrates of the process should be different. If several copies of the same EPN are involved in the process, cardinality label of *consumption* arc should be used.

3. All products of the process should be different. If several copies of the same EPN are produced in the process, cardinality label of *production* arc should be used.

4. Once cardinality label set to one arc of the EPN all other arcs should make their cardinality visible, even if it is undefined. In a case cardinality is undefined or unknown question mark ("?") should be placed as cardinality label.

5. PN should have only one *Catalysis* arc connected to it. If there more than one catalyst known for the process several PNs should be drawn.

6. PN should have only one *Necessary Stimulation* arc connected to it. If there is more than one EPN acting as a necessary stimulator on a process then several a logic gate should be used.

3.4.4.4 Association

1. Composition of the *complex* or *multimer* produced by *association* should be identical to set of *association* substrates.

2. If *association* produces *complex*, than number of *consumption* arcs should be two or more.

3. If *association* produces *multimer*, than it can have only one *consumption* arc. In that case if substrate is not a *multimer*, then the number of monomers in a product *multimer* should be equal to cardinality of that arc. If substrate is another *multimer*, then cardinality of the consumption arc should be equal to ratio of number of monomers in product and in substrate *multimers*.

3.4.4.5 Dissociation

1. Composition of *complex* or *multimer* consumed by *dissociation* should be identical to set of products of the process.

2. If *complex* is consumed by the *dissociation*, than the process should have two or more *production* arcs.

3. If *multimer* is split by *dissociation* process, than one *production* arc could be connected to the process node. In that case if product is not a *multimer*, then the number of monomers in a substrate *multimer* should be equal to cardinality of that arc. If product is another *multimer*, then cardinality of the *production* arc should be equal to ratio of number of monomers in substrate and in product *multimers*.

3.5 Semantic rules

3.5.1 Namespaces

The notation has a concept of a namespace within which entities with the same identifying attributes are regarded as identical. The SBGN namespaces are shown in table 3.3.

Table 3.3: *Namespace scope definitions.*

Namespace Scope	Entities affected	Notes
Map	CompartmentNode, SubMap, EquivalenceNode	
CompartmentShape	BasicEntityNode	If no *compartment* is drawn then all BasicEntityNodes are assumed to belong to an invisible "default" compartment.

continued on next page

continued from previous page

Namespace scope	Entities Effected	Notes
EntityType	StateVariable, Annotation	
ComplexType	EntityType	

3.5.2 Cloning

SBGN only allows identical nodes to duplicated on a map if they are explicitly marked as such. This is using a *labeled clone marker* with all the nodes that are identical. The details are shown in table 3.4.

Table 3.4: *Duplication rules.*

Node	Can Duplicate?	Indication	Additional Rules
CompartmentNode	N		
SimpleChemicalNode	Y	*Simple clone marker*	
UnspecifiedEntityNode	Y	*Simple clone marker*	
SourceNode	N		
SinkNode	N		
Perturbing Agent	Y	*Simple clone marker*	
Phenotype	Y	*Simple clone marker*	
MultimerChemicalEntity	Y	*Simple clone marker*	
StatefulEntityPoolNode	Y	*Labeled clone marker*	
MacromoleculeNode	Y	*Labeled clone marker*	
MultimerMacromoleculeNode	Y	*Labeled clone marker*	
NucleicAcidFeatureNode	Y	*Labeled clone marker*	
ComplexNode	Y	*Labeled clone marker*	
ProcessNode	Y	None	Duplication is implied when all EPNs linked to the ProcessNode are marked as clones.
OmittedProcessNode	Y	As Process Node	
UncertainProcessNode	Y	As Process Node	
AssociationNode	Y	As Process Node	
DissociationNode	Y	As Process Node	
LogicalOperatorNode	Y	None	
ANDNode	Y	None	
ORNode	Y	None	
NOTNode	Y	None	

3.5.3 State variables

States variables are very simple. The variable can have a name. If the name is set, it should be displayed. The variable can take any value. The names used in the controlled vocabulary of post-translational modification in section 2.2 are reserved. Authors should avoid attaching any other meaning to them.

An entity pool node has a set of state variables, which is the union of all the state variables used in the map. All stateful EPNs must explicitly display the same set of state variables and each state variable must be assigned a value or be assigned the value “Not Set”.

3.5.4 Compartment spanning

In all cases 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. ComplexNodes may contain EPNs which belong to different compartments and in this way a complex can be used to describe an entity that spans more than one compartment.

This restriction makes it impossible to represent in a semantically correct way a macromolecule that spans more then 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 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.

3.5.5 Compartments

The layout of compartments in an SBGN map does not imply anything about the topology of compartments in the cell. Compartments should be bounded and may overlap. However, adjacency and the nesting of compartments does not imply that these compartments are next to each other physically or hat one compartment contains the other.

3.5.6 Modulation

It is implied, but not defined explicitly that a process has a rate at which it converts its input EPNs to its output EPNs. This concept is important in understanding how SBGN 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. Modulation changes the basal rate in an unspecified fashion.
3. Stimulation is a modulation that's effect is to increase the basal rate.
4. Inhibition is a modulation that's effect is to decrease 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.
7. At most one necessary stimulation can be assigned to a process. Two necessary stimulations would imply an implicit Boolean AND or OR operator. For clarity only one necessary stimulation can be assigned to a process and such combinations must be explicitly expressed as the Boolean operators.
8. At most one catalysis can be assigned to a process. A catalysis modulation implies that the exact biochemical mechanism underlying the process is known. In this context two catalysis reactions cannot be assigned to the same process as they are independent reactions. Other EPNs that modulate the catalysis can be assigned to the same process as modulators, stimulators, and inhibitors and will have a multiplicative modulation on the reaction rate defined by the catalysis.

3.5.7 Reversible Processes

Process nodes are deemed to be reversible if they have production arcs on both "sides" of the process. A mixture of consumption and production arcs on the same side of a process is not permitted. Semantically the production arc can be thought of as allowing a reversible flow of entities between the process and EPN. A Consumption arc only permits an irreversible flow from the EPN to the process. In this way the consumption arc forces the process to be irreversible.

Consumption arcs cannot be associated with both sides of a process as this would prohibit any flow through the process.

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.

Note that a sink cannot be linked to a reversible process as a sink and only received entities, and so would effectively make the process irreversible.

3.5.8 Submaps

Submaps are a visual device that allow a map to be split into several views. They remain, however, part of the main map and share its namespace. As a test of validity it should be possible to reintroduced a submap into the main map by eliminating the SubMapNode and merging the equivalent EntityPoolNodes in both maps.

3.5.8.1 Rules for mapping to submaps

An EntityPoolNode in the main map can be mapped to one in the submap using a TagNode in the submap and SubMapTerminals (see Section 2.7.1) in the main map. For a mapping between map and submap to exists the following must be true:

1. The identifiers in the TagNode and SubMapTerminals must be identical.
2. The EntityPoolNodes must be identical.

3.5.8.2 Requirement to define a mapping

If a map and submap both contain the same EntityPoolNode, then a mapping between them must be defined as above.

3.5.8.3 Cloning consistency

If an EntityPoolNode is cloned in the main map then it must also be marked as cloned in the submap even if there is only one copy of the EntityPoolNode in the submap. The converse rule also applies where the EntityPoolNode is cloned in the submap, but not the main map.

3.6 Summary of Rules

This section summarises the rules of SBGN-PD in a form that is intended to be accessible to tool developers and those interested in validating process maps. Each rule has been given an identifier for ease of reference. Note that no meaning is attached to the rule identifier and any perceived ordering of the identifiers is not significant.

3.6.1 Entity Pool Nodes

PD1 The identity of an EPN is defined by a combination of its compartment, entity type (e.g. complex or macromolecule), name and state variables (if any).

PD2 A Complex's identity consists of the identity of its subunits and a name is optional. All complexes with the same name should have the same subunits.

PD3 An EPN may belong to only one compartment.

PD4 An EPN belongs to only one compartment. If no compartment is draw it is assumed to belong to a "default" compartment.

PD5 An EPN can overlap more than one compartment and in this case it is deferred to the drawing tool or author to assign the owning compartment. Note this rule need not apply in cases where SBGN is draw by hand.

PD6 A Complex may contain subunits that belong to different compartments (the complex itself will belong to only one, however).

PD7 The layout or organisation of a compartment does not imply anything about its topology.

PD8 The layout or organisation of the EPNs in a complex does not imply any information about topology.

PD10 Complexes can be nested. This does imply information about the complex's topology.

PD11 A complex should consist of different EPNs. If two or more elements of the complex are identical then they should be replaced by a multimer.

PD12 Source and sink nodes must have no name and be attached to only one consumption or production arc.

PD13 An EPN must be connected to at least 1 consumption, production or modulation arc.

PD14 An EPN is not allowed to be a substrate and product of the same process. This applies to cloned EPNs as well. States

PD15 All state variables in a stateful EPN should have different names.

PD16 A blank state variable has the value "unset".

PD17 The state of a complex is the sum of its subunits' and its own state variables.

PD48 The *Sink* cannot be linked to a reversible process.

3.6.2 Compartments

PD18 Compartments cannot be nested. Compartments may overlap, but overlap does not imply containment.

PD19 The layout or organisation of a compartment does not imply anything about its topology.

PD20 If no compartment is draw it is assumed to belong to a "default" compartment.

PD21 If one or more compartment glyphs are drawn then all EPNs must owned by a compartment glyph.

3.6.3 Process Nodes (PN)

PD22 A Process Node should have non-zero number of consumption and production links.

PD23 All substrates of the Process Node should be different. If several copies of the same EPN are involved in the process, the cardinality label of consumption arc should be used.

PD24 All products of the PN should be different. If several copies of the same EPN are produced in the process, the cardinality label of production arc should be used.

PD25 Once the cardinality label is added to one arc connected to a PN all other such arcs should display a cardinality label.

PD26 The cardinality of an arc can be undefined or unknown, in which case a question mark ("?") should be used.

PD27 A PN should correspond to only one process or series of connected process. The same set of EPNs should be connected by different PNs if they are consumed and produced by different processes.

PD28 The composition of the products of an association process should be equivalent to its substrates

PD29 The composition of the products of a dissociation process should be equivalent to its substrate.

3.6.4 Modulation and Logical Operators

PD30 A PN with no modulations has an underlying "basal rate" which describes the rate at which it converts inputs to outputs.

PD31 Modulation changes the basal rate in an unspecified fashion.

PD32 Stimulation is a modulation that's effect is to increase the basal rate.

PD33 Inhibition is a modulation that's effect is to decrease the basal rate.

PD34 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.

PD35 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.

PD36 At most one trigger can be assigned to a process. Two triggers would imply an implicit Boolean AND or OR operator, so for clarity only one trigger can be assigned to a process and such combinations must be explicitly expressed as the Boolean operators.

PD37 The PN should have only one Catalysis arc connected to it. A catalysis modulation implies that the exact biochemical mechanism underlying the process is known.

PD38 The PN should have only one Trigger arc connected to it.

PD39 AND and OR Boolean logic gates should have two or more input and one output.

PD40 A NOT gate can only have one input and output.

3.6.5 Cloning and Sub-Maps

PD41 Duplicate EPNs must be marked as clones.

PD42 Duplicate stateful EPNs must use a Labelled Clone Marker.

PD43 Duplicate non-stateful EPNs must use a simple clone marker.

PD44 A sub-map shares the same namespace as its main map.

PD45 To map an EPN in the main map to one in a sub-map, the SubMap glyph in the main map must contain an identifier that matches the identifier of a Tag in the sub-map. Both EPNs must be identical.

PD46 If a main map and a sub-map contain an identical EPN then a mapping must exist between them.

PD47 If an EPN is cloned in the main map, then it must be marked as cloned in the sub-map, with the same identifier in both maps. This is true if there is only one EPN of this type in the submap.

Chapter 4

Layout Guidelines for a Process Description

4.1 Introduction

The previous chapters describe the appearance and meaning of SBGN Process Description Level 1 components. Objects are *EPNs*, *process nodes*, *container nodes*, *logical operators* as well as *connecting arcs*. 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*.

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 guidelines for the layout of process description maps, divided into two categories:

1. requirements, i. e. rules which **must** be fulfilled by a layout, and

2. recommendations, i. e. rules which **should** be followed if possible.

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

Those layout guidelines 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 guidelines 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].

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.

4.2 Layout guidelines

4.2.1 Requirements

Requirements are rules which **must** be fulfilled by a layout to produce a valid SBGN Process Description Level 1 graph.

4.2.1.1 Node-node overlaps

Nodes are only allowed to overlap in two cases:

1. the overlapping nodes define a glyph (e. g. a *multimer* composed by stacking of two containers representing the monomers).
2. nodes overlapping compartments (e. g. a complex placed on the compartment border).

Otherwise, nodes are not allowed to overlap (Figure 4.1). This includes the 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. Submaps are not allowed to overlap.

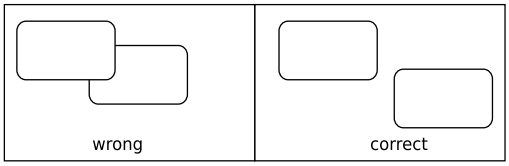

Figure 4.1: *Nodes must not overlap.*

4.2.1.2 Node-edge crossing

In case of node-edge crossing the edge must be drawn on the top of the node (Figure 4.2). See also recommendation 4.2.2.1 (crossing between edges and nodes should be avoided).

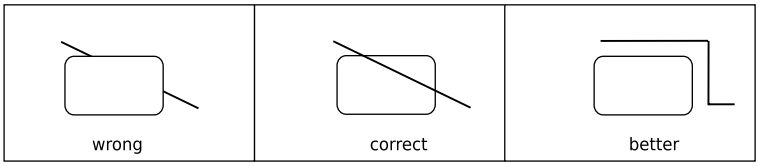

Figure 4.2: *If an edge crosses a node, the edge must be drawn on top of the node.*

4.2.1.3 Node border-edge overlaps

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

4.2.1.4 Edge-edge overlaps

Edges are not allowed to overlap (Figure 4.4 on the next page). 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.

4.2.1.5 Node orientation

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

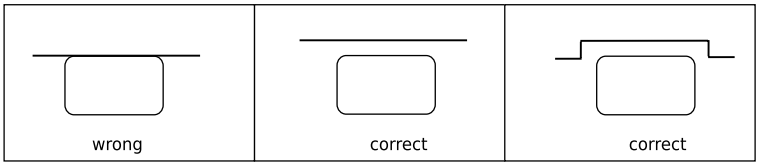

Figure 4.3: *Edges must not overlap node borders.*

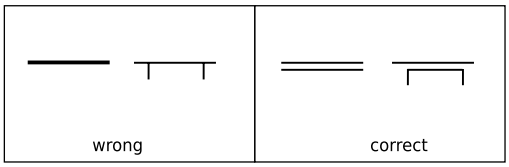

Figure 4.4: *Edges must not overlap.*

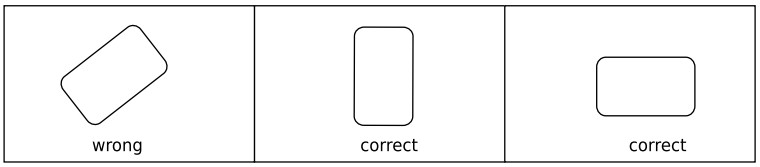

Figure 4.5: *The node orientation must be horizontally or vertically.*

4.2.1.6 Node-edge connection

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). 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*. A *process* connected to *production arcs* on opposite sides is a reversible process.

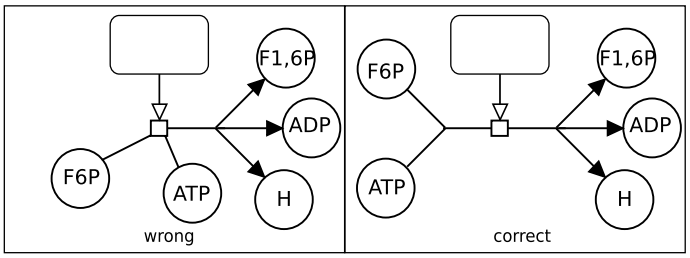

Figure 4.6: *Arcs between a* process *and the* consumption *and* production arcs *must be attached to the center of opposite sides, modulatory arcs must be attached to the other two sides.*

4.2.1.7 Node labels

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 nodes or other labels (this includes touching of other nodes or labels).

4.2.1.8 Edge labels

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

4.2.1.9 Compartments

If a process has all participants in the same compartment the process node and all edges/arcs have to be in this compartment. If a process has participants in at least two different compart-

ments, the process node has to be either in a compartment where the process has at least one participant or in the empty space.

4.2.2 Recommendations

Recommendations are rules which should be followed if possible to produce layouts may be easier to understand.

4.2.2.1 *Node-edge crossing*

Crossings between edges and nodes should be avoided. 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). See also requirement 4.2.1.2 (in case of node-edge crossings the edge must be drawn on the top of the node).

4.2.2.2 *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.

4.2.2.3 *Avoid edge crossings*

The amount of crossings between edges should be minimized.

4.2.2.4 *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).

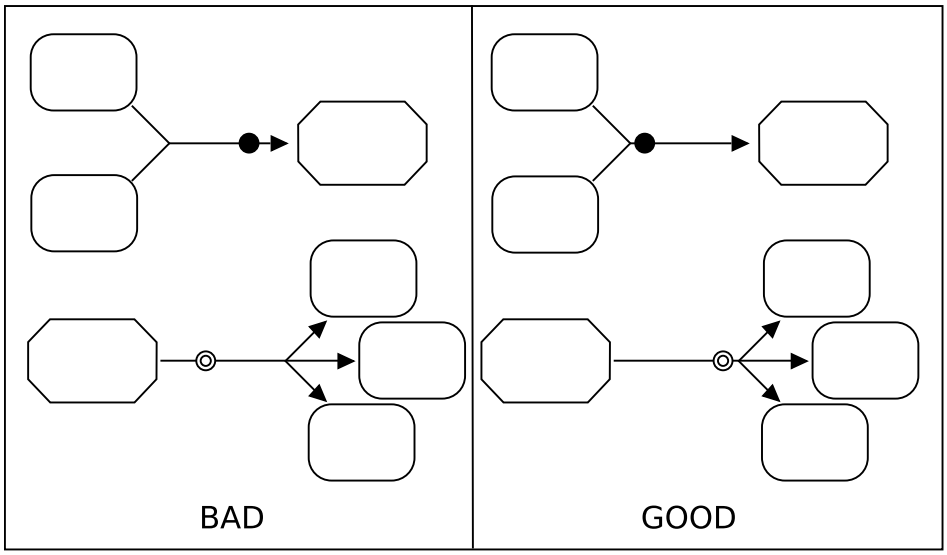

Figure 4.7: *Branching points should be close to association and dissociation symbols.*

4.2.2.5 *Units of information*

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

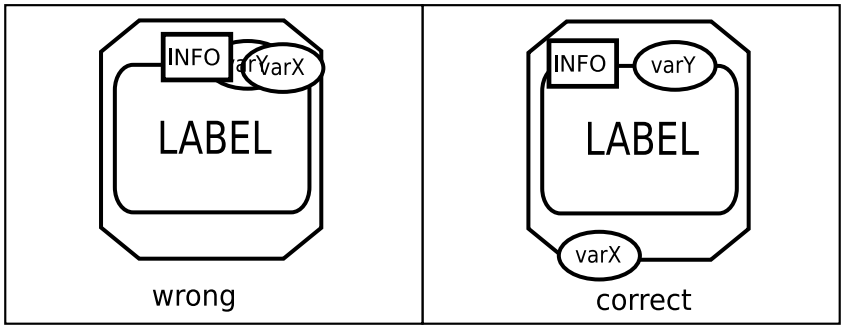

Figure 4.8: *Units of information should not overlap with any other element.*

4.2.3 Additional suggestions

Here is a list of additional layout suggestions which may help in producing aesthetically more pleasing layouts which may be easier to understand.

- Angle of edge crossings: If edge crossings are not avoidable edges should cross with an angle close to 90 degrees.
- Placement of substrates and products of a process: Substrate and product nodes should be placed on different sides of the process node.
- 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.
- Edge length: Long edges should be avoided.
- Number of edge bends: Edges should be drawn with as few bends as possible.
- Similar and symmetric parts: Similar parts of a map should be drawn in a similar way, and symmetric parts should be drawn symmetrically.
- Proximity information: Related elements (e. g. nodes connected by a process or all elements within a compartment) should be drawn close together.
- 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).
- Compartments: It can help clarity to use a different background shade or color for each compartment.

Chapter 5

Acknowledgments

Here we acknowledge those people and organisations the assisted in the develeopment 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.

5.1 Level 1 Release 1.0

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 Frank Bergmann, Sarala Dissanayake, Ralph Gauges, Peter Ghazal, Lu Li, and Steven Watterson.

5.2 Level 1 Release 1.1

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

5.3 Comprehensive list of acknowledgements

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 aim this list to be rather complete. We are very sorry if we forgot someone, and will be grateful if you notified us of any omission.

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.

5.4 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 SBGN Process Description Level 1 graphs

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 examplifies the use of the *EPNs simple chemical*, *macromolecule*, and *clone marker*, the *PNs process*, and the *connecting arcs consumption*, *production* and *catalysis*.

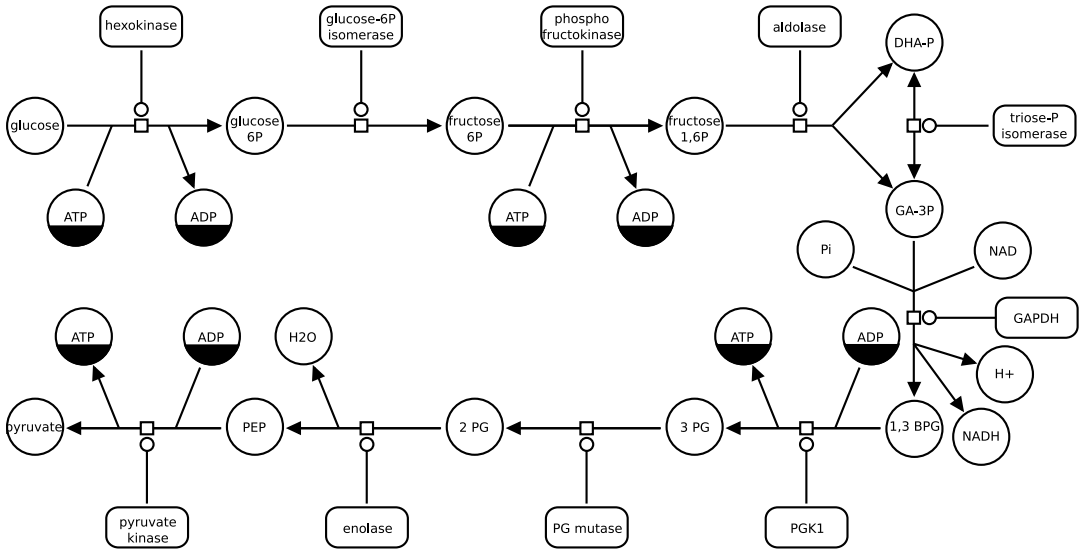

Figure A.1: *Glycolysis. This example illustrates how SBGN can be used to describe metabolic pathways.*

Figure A.2 on the next page presents an example of signalling pathway, that examplifies 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 on page 65 is an expanded version of the submap present on the map present in Figure A.2 on the next page. It shows the use of *tag*.

Figure A.4 on page 66 introduces an SBGN Process Description that spans several compart-

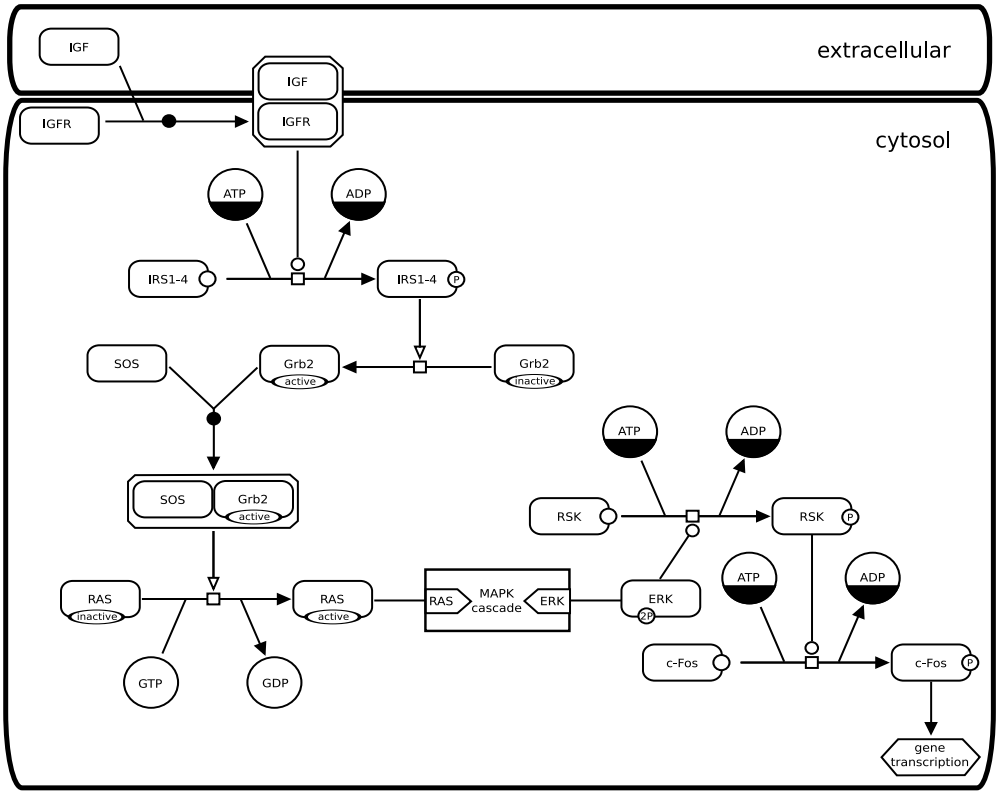

Figure A.2: *Insulin-like Growth Factor (IGF) signalling. This example shows the use of compartments and how details can be hidden by using a submap. The submap is shown on Figure A.3 on the next page.*

ments. 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 "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 arc necessary stimulation*.

Figure A.5 on page 66 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*.

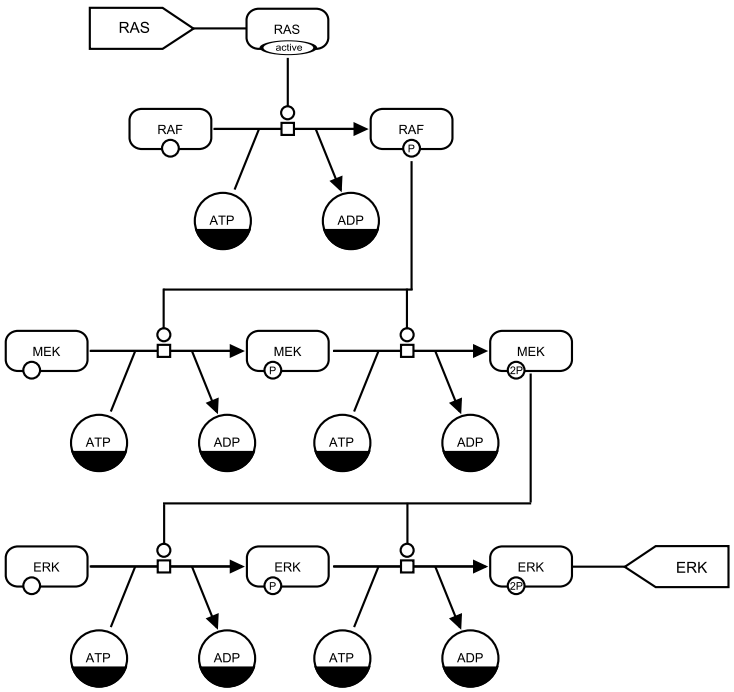

Figure A.3: *A submap of the previous map showing the MAPK cascade.*

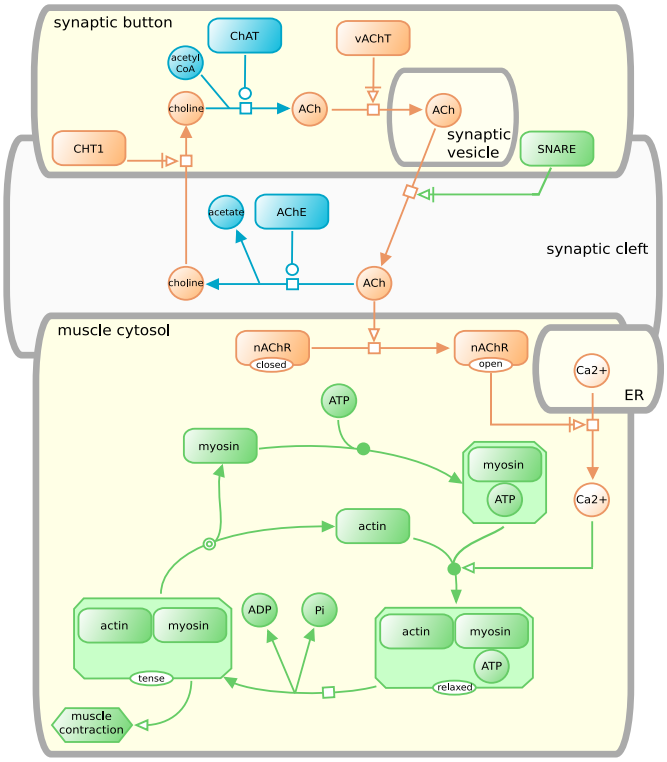

Figure A.4: *Neuronal/Muscle signalling. A description of inter-cellular signalling using SBGN.*

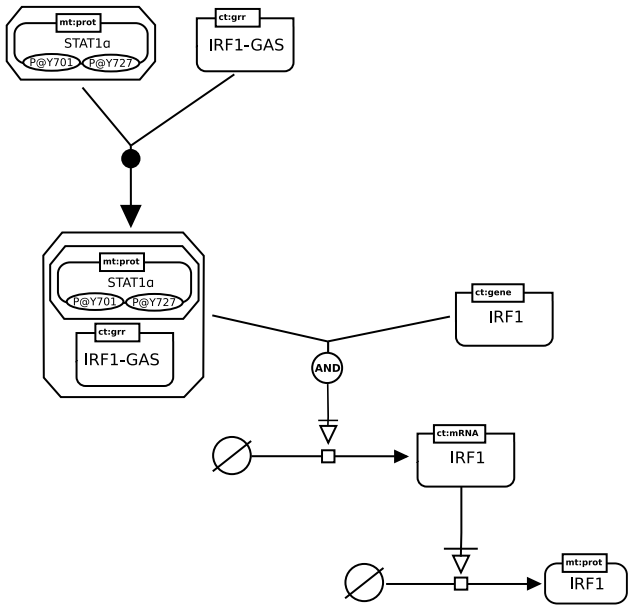

Figure A.5: *Activated STAT1α induction of the IRF1 gene. An example of gene regulation using logical operators.*

Appendix B

Reference card

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

Entity Pool Nodes
LABEL
unspecified entity
LABEL
simple chemical
LABEL
macromolecule
LABEL
nucleic acid feature
LABEL
perturbing agent
source
sink
INFO
varZ
varY
varX
varW
complex
Auxiliary Units
pre:label
unit of information
val@var
state variable
LABEL
marker
clone marker
N:5
N:2
multimers
Process Nodes
process
omitted process
uncertain process
association
dissociation
phenotype
Container Nodes
e:INFO
compartment
Reference Nodes
EPN
or CN
A
B
C
submap
tag
Connecting arcs
Source
EPN
N
Target
EPN
consumption
production
modulation
Target
PN
stimulation
catalysis
inhibition
necessary stimulation
Source
EPN
Logical
operator
logic arc
EPN
equivalence arc
Logical Operators
AND
and operator
OR
or operator
NOT
not operator

Appendix C

Issues postponed to future levels

C.1 Multicompartment entities

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.
- On 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

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

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

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*.

C.5 State and transformation of compartments

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 recult, 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.

Appendix D

Revision History

D.1 Release 1.0 to Release 1.1

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

Description	Tracker ID
Regarding modulation of reversible processes, changed "should" to "must" be represented by two *process*	
Removed "The connectors and the box move as a rigid entity" in the definition of *process*	
Changed the definition of process node to "represent processes that transform one or several EPNs into *one or several EPNs, identical or different*"	
Changed SBO term of *compartment* From SBO:0000289 (functional compartment) to SBO:0000290 (physical compartment)	
Reoganised classification of glyphs	
Reoganised 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 semantc 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
Sink and source are lumped together	2726435
SBO terms are incorrect or missing.	2841261
Compartment description is confusing and contradictory.	2841122
Clone marker fill percentages unhelpful.	2841114
Use of CV for physical charactetistic not clear.	2841085
Definition of Cardinality is ambiguous.	2840996
input to AND on IFN example.	2804326
more SBO terms for *multimers*	2803593
Legend of figure 2.20 is incorrect	2803537
continued on next page	

continued from previous page

Description	Tracker ID
legend of figure 3.2	2802990
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

Bibliography

[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.

[2] M. Kaufmann and D. Wagner. *Drawing Graphs: Methods and Models*, volume 2025 of *Lecture Notes in Computer Science Tutorial*. Springer, 2001.