Systems Biology Graphical Notation: Process Diagram Level 1

Draft of June 13, 2008

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

Editors:

Nicolas Le Novère	*EMBL European Bioinformatics Institute, UK*
Stuart Moodie	*University of Edinburgh, UK*
Anatoly Sorokin	*University of Edinburgh, UK*
Michael Hucka	*California Institute of Technology, USA*

Principal Authors:

Nicolas Le Novère	*EMBL European Bioinformatics Institute, UK*
Stuart Moodie	*University of Edinburgh, UK*
Anatoly Sorokin	*University of Edinburgh, UK*
Michael Hucka	*California Institute of Technology, USA*
Falk Schreiber	*Leibniz Institute IPK, Germany*
Emek Demir	*MSKCC Computational Biology Center, USA*
Huaiyu Mi	*SRI International, USA*
Yukiko Matsuoka	*The Systems Biology Institute, Japan*
Katja Wegner	*University of Hertfordshire, UK*
and	
Hiroaki Kitano	*The Systems Biology Institute, Japan*

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-team@sbgn.org.

Preface

Acknowledgements

The authors are grateful to all the attendees of the SBGN meetings, as well as to the subscribers of the sbgn-discuss@sbgn.org mailing list. The authors would like to acknowledge especially the help of Frank Bergmann, Sarala Dissanayake, Ralph Gauges, Peter Ghazal, Igor Goryanin and Lu Li. SM and AS would also like to acknowledge Igor Goryanin whose financial support and encouragement enabled us to commit the neccessary time to the development of this specification.

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/) and also benefitted from the help of a Japan Partering Award of the British *Biotechnology and Biological Sciences Research Council* (BBSRC, http://www.bbsrc.ac.uk/)

Notes on typographical conventions

The concept represented by a glyph is written using a normal font, while a *glyph* means the SBGN visual representation of the concept.

Contents

Chapter 1

What is the Systems Biology Graphical Notation?

The goal of the **S**ystems **B**iology **G**raphical **N**otation (SBGN) is to standardize the graphical/visual representation of essential biochemical and cellular processes studied in systems biology. SBGN defines a comprehensive set 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 represented in machine-readable form, to ensure consistent storage, exchange, and reproduction of graphical representation between different software systems.

Standardizing graphical notations for describing biological interactions is an important step towards the efficient and accurate transmission of biological knowledge between different communities. Traditionally, diagrams representing interactions among genes and molecules have been drawn in an informal manner, using simple unconstrained shapes and edges such as arrows. Until the development of SBGN, no standard agreed-upon convention existed defining exactly how to draw such diagrams in a way that helps readers interpret them consistently, correctly, and unambiguously. By standardizing the visual notation, SBGN can serve as a bridge between different communities such as computational and experimental biologists, and even more broadly in education, publishing, and more.

For SBGN to be successful, it must satisfy a majority of technical and practical needs, and must be embraced by the community of researchers in biology. With regards to the technical and practical aspects, a successful visual language must meet at least the following goals:

1. Allow the representation of diverse biological objects and interactions;
2. Be semantically and visually unambiguous;
3. Allow implementation in software that can aid the drawing and verification of diagrams;
4. Have semantics that are sufficiently well defined that software tools can convert graphical models into mathematical formulas for analysis and simulation; and
5. Be unrestricted in use and distribution, so that the entire community can freely use the notation without encumbrance or fear of intellectual property infractions.

This document defines the *Process Diagram* visual language of SBGN. As explained more fully in Section 1.2 on the following page, Process Diagrams are one of three views of a model offered by SBGN. It is the product of many hours of discussion and development by many individuals and groups. In the following sections, we describe the background, motivations, and context of Process Diagrams.

1.1 History of SBGN development

The effort to create a well-defined visual notation was pioneered by Kurt Kohn with his Molecular Interaction Map (MIM), a notation defining symbols and syntax to describe the interactions of molecules [1]. MIM is essentially a variation of entity-relationship diagrams [**?**]. Kohn's work was followed by numerous other attempts to define both alternative notations for diagramming cellular processes (e.g., the work of Pirson and colleagues [2], BioD [3], and others), as well as extensions of Kohn's notation (e.g., the Diagrammatic Cell Language of Maimon and Browning [4]).

Kitano originated the idea of having multiple views of the *same* model. This addresses two problems: no single view can satisfy the needs of all users, and a given view can only represent a subset of the semantics necessary to express biological knowledge. Kitano proposed the development of process diagrams, entity-relationship diagrams, timing charts (to decribe temporal changes in a system), and abstract flow charts [5]. The Process Diagram notation was the first to be fully defined using a well-delineated set of symbols and syntax [6]. It lead to a desire to establish a unified standard for graphical representation of biochemical entities, and from this arose the current SBGN effort. Separately and roughly concurrently, other groups designed similar notations, for example the Edinburgh Pathway Notation [7]. All of these efforts began to attract attention as more emphasis in biological research was placed on networks of interactions and not just characterization of individual entities.

In 2005, thanks to funding from the Japanese agency *The New Energy and Industrial Technology Development Organization* (NEDO, `http://www.nedo.go.jp/`), Kitano initiated the Systems Biology Markup Language project as a community effort. The first SBGN workshop was held in February 2006 in Tokyo, with over 30 participants from major organizations interested in this effort. From the in-depth discussions held during that meeting emerged a set of decisions that are the basis of the current SBGN specification. These decisions are:

- SBGN should be made up of two different visual grammars, describing Entity Relationship and Process Diagram diagrams (called *State Transition* diagrams at the time). See Section 1.2.
- In order to promote wide acceptance, the initial version(s) of SBGN should stick to at most a few dozens symbols that non-specialists could easily learn.

The second SBGN workshop was held in October, 2006, in Yokohama, Japan. This meeting featured the first technical discussions about which symbols to include in SBGN Level 1, as well as discussions about the syntax, semantics, and layout of graphs. A follow-up technical meeting was held in March, 2007, in Heidelberg, Germany; the participants of that meeting fleshed out most of the design of SBGN. The third SBGN workshop, held in Long Beach in October of 2007, was dedicated to reaching agreement on the final outstanding issues of notation and syntax. The participants of that meeting collectively realized that a third language would be necessary: the Activity Flow diagrams. The specification for the Process Diagram language was finalized and largely completed during a follow-up technical meeting held in Okinawa, Japan, in January of 2008. At this last meeting, attendees also held the first in-depth discussions about the syntax of the Entity Relationship language.

SBGN workshops are an opportunity for public discussions about SBGN, allowing interested persons to learn more about SBGN and help identify needs and issues. More meetings are expected to be held in the future, long after this specification document has been issued.

1.2 The three languages of SBGN

Readers may well wonder, why are there *three* languages in SBGN? The reason is that this approach solves a problem that was found insurmountable any other way: attempting to include

all relevant facets of a biological system in a single diagram causes the diagram to become hoplessly complicated and incomprehensible to human readers.

The three different notations in SBGN correspond to three different *views* of the same model. These views are representations of different classes of information, as follows:

1. *Process Diagram*: the causal sequences of molecular processes and their results
2. *Entity Relationship*: the interactions between entities irrespective of sequence
3. *Activity Flow*: the flux of information going from one entity to another

In the Process Diagram view, each node in the diagram represents a given *state* of a species, and therefore a given species may appear multiple times in the same diagram if it represents the same entity in different states. Conversely, in the Entity Relationship view, a given species appears only once in a diagram. Process Diagrams are suitable for following the temporal aspects of interactions, and are easy to understand. The drawback of the Process Diagram, however, is that because the same entity appears multiple times in one diagram, it is difficult to understand which interactions actually exist for the entity. Conversely, Entity Relationship diagrams are suitable for understanding relationships involving each molecule, but the temporal course of events is difficult or impossible to follow because Entity Relationship diagrams do not describe the sequence of events.

Process Diagrams can quickly become very complex. Moreover, when diagramming a biochemical network, one often wants to ignore the biochemical basis underlying the action of one entity on the activity of another. A common desire is to represent only the flow of activity between nodes, without representing the transitions in the states of the nodes. This is the motivation for the creation of the Activity Flow view. Activity Flow diagrams permit the use of *modulation*, *stimulation* and *inhibition* and allow them to point to State/Entity nodes rather than process nodes. The Activity Flow view is thus a hybrid between Process Diagram and Entity Relationship diagrams. It is particularly convenient for representing the effect of perturbations, whether genetic or environmental in nature.

A recurring argument in SBGN deveopment is that these these three types of diagrams should be merged into one. Unfortunately, each view has such different meanings that merging them would compromise the robustness of the representation and destroy the mathematical integrity of the notation system. While having three different notations makes the overall system more complex, much of the complexity and increase in burden on learning is mitigated by reusing most of the same symbols in all three notations. It is primarily the syntax and semantics that change between the different views, reflecting fundamental differences in the underlying mathematics of what is being described.

1.3 SBGN Levels

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 require a vast language that most newcomers would shun as being too complex. Thus, the SBGN community followed an idea used in SBML development [8]: stratify language development into levels.

A *level* of SBGN 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. 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. In this way, SBGN development is envisioned to proceed in stages, with each higher SBGN level adding richness compared to the levels below it.

1.4 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@sbgn.org) 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.

Chapter 2

Process Diagram Glyphs

This chapter provides a catalog of the graphical symbols available for representing entities in Process Diagrams. There are different classes of glyphs corresponding to different classes of material or conceptual entities, containers, processes, connecting arcs, and logical operators. In Chapter 3 beginning on page 37, we describe the rules for combining these glyphs into a legal SBGN Process Diagram, and in Chapter 4 beginning on page 54, we describe requirements and guidelines for the way that diagrams are visually organized.

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 Diagram notation with the help of an example shown in Figure 2.1.

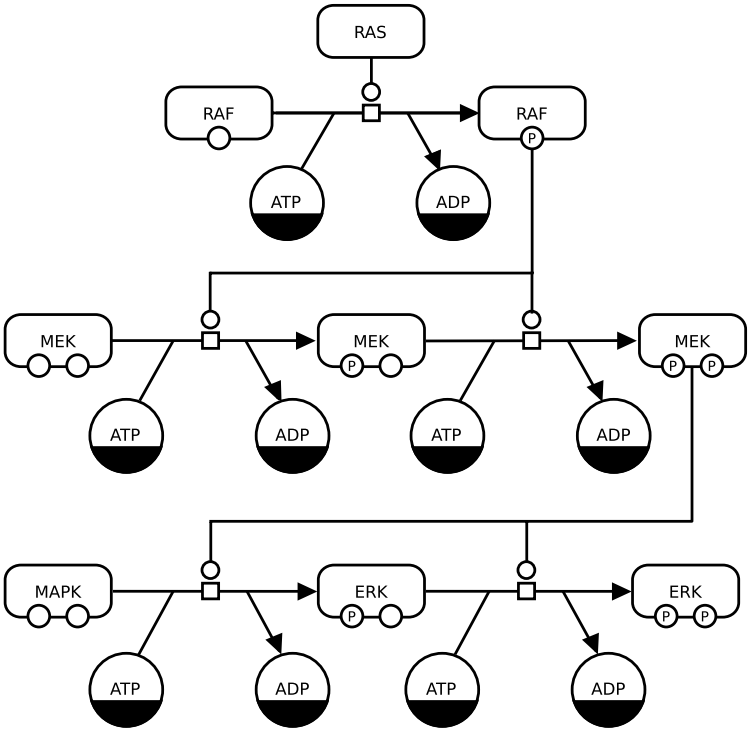


Figure 2.1: *This example of a Process Diagram uses two kinds of entity pool nodes: one for pools of different macromolecules (Section 2.3.3) and another for pools of simple chemicals (Section 2.3.2). Most macromolecule nodes in this diagram are adorned with units of information (Section 2.3.10) representing phosphorylation states. This diagram uses one type of process node, the transition node (Section 2.5.1), and one kind of connecting arc, catalysis (Section 2.6.5). Finally, some entity pool nodes have dark bands along their bottoms; these are clone markers indicating that the same pool nodes appear multiple times in the diagram.*

The diagram in Figure 2.1 on the previous page is a simple diagram 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 SBGN by lines with arrows and other decorations. In this particular diagram, all of the processes happen to be the same: transitions catalyzed by biochemical entities. The directions of the arrows indicate the direction of the transitions; for example, unphosphorylated RAF kinase transitions to phosphorylated RAF kinase via a process catalyzed by RAS. ATP and ADP are also involved in the same process, but the focus of the diagram is on what happens to RAF, so ATP and ADP are shown as incidental to the main transition. The small circles on the nodes for RAF and other entity pools represent units of information (in this case, phosphorylation sites), while the other small circles touching small squares indicate a specific kind of process. The details of this and other aspects of Process Diagram are explained in the rest of this chapter.

The essence of the Process Diagram is *change*: it shows how different entities in the system transition 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 material and conceptual 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 a type of entity is transformed into another is conveyed by *process nodes*, 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, they happen all to be catalysis, Section 2.6.5, but others are possible. Finally, nodes in Process Diagrams are not usually repeated; if they do need to be repeated, they are marked with *clone markers*—specific modifications to the appearance of the node (Section 2.3.12).

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	Transition, association, dissociation
Connecting arc	—	Links from EPNs to PNs to indicate influence by the EPNs on the PNs	Stimulation, modulation, catalysis, inhibition
Logical operators	—	Combines multiple inputs into one output	Boolean *and*, *or*, *not*

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

2.2 Controlled vocabularies used in SBGN Process Diagram Level 1

Some glyphs in SBGN Process Diagrams can contain particular kinds of textual annotations conveying information relevant to the purpose of the glyph. These annotations are *units of information* (Section 2.3.10). 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 be prefixed with a controlled vocabulary term indicating the type of information being expressed. The prefixes are mandatory. 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 Diagram 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 terms' meanings in a figure legend or other text accompanying the diagram.

2.2.1 Entity Pool Node material types

The material type of an Entity Pool Node (EPN) indicates its chemical structure. A list of common material types is shown in Figure 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 *participant→physical participant*). The labels are defined by SBGN Process Diagram 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

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

lThe 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 Diagram. A list of common conceptual types is shown in Figure 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 *participant→physical participant*). The labels are defined by SBGN Process Diagram 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
Messenger RNA	`ct:mRNA`	SBO:0000278

Figure 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 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 many 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.

Figure 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* under *events→reaction→biochemical reaction→conversion→addition*). The labels shown in Figure 2.4 are defined by SBGN Process Diagram Level 1; for all other kinds of modifications not listed here, the author of a Process Diagram must create a new label (and should also describe the meaning of the label in a legend or text accompanying the diagram).

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

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

2.2.4 Physical characteristics of compartments

SBGN Process Diagram Level 1 defines a special unit of information for describing certain common physical characsteristics of compartments. Figure 2.5 lists the particular values defined by SBGN Process Diagram Level 1. The values correspond to the Systems Biology Ontology branch with identifier `SBO:0000255` (*physical characteristic* under *quantitative parameter*).

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

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

2.2.5 Cardinality

SBGN Process Diagram Level 1 defines a special unit of information usable on multimers for describing the number of monomers composing the multimer. Figure 2.6 on the following page

shows the way in which the values must be written. Note that the value is unitary number, and not (for example) a range. There is no provision in SBGN Process Diagram Level 1 for specifying a range in this context because it leads to problems of entity identifiability.

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

Figure 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 Entity pool nodes

SBGN Process Diagram Level 1 contains five glyphs representing classes of material entities: *unspecified entity*, *simple chemical*, *macromolecule*, *genetic entity*, and *multimer*. (Specific types of macromolecules, such as protein, RNA, DNA, polysaccharide, and specific simple chemicals are not defined by SBGN Process Diagram Level 1 but may be part of future levels of SBGN.) In addition to the material entities, SBGN Process Diagram Level 1 represents four conceptual entities: *source/sink*, *perturbation*, *observable*, and *tag*. Material and conceptual entities can optionally carry auxiliary units such as *units of information*, *state variables* and *clone markers*.

2.3.1 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 model. 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 the model, 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 SBGN Process Diagram Level 1 specification.)

SBO Term:

SBO:0000285 ! material entity of unknown nature

Container:

An *unspecified entity* is represented by an ellipsoid container, as shown in Figure 2.7.

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 2.3.12).

Figure 2.7: *The Process Diagram glyph for* unspecified entity.

2.3.2 Glyph: *Simple chemical*

A simple chemical in SBGN is defined as the opposite of a macromolecule (Section 2.3.3): 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 2.8.

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 2.3.10). A particular *unit of information* describes the material type. A *simple chemical* may also carry a *clone marker* (Section 2.3.12).

Figure 2.8: *The Process Diagram glyph for* simple chemical.

2.3.3 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 Diagram 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 diagram 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 macromolecules.)

SBO Term:

SBO:0000245 ! macromolecule

Container:

A macromolecule is represented by a rectangular container with rounded corners, as illustrated in Figure 2.9 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 2.3.11). The state of a macromolecule is therefore defined as the vector of all its state variables. A state variable is represented by an ellipsoid container, with the long axis of the ellipsoid placed on the border of the *macromolecule*'s container as illustrated in Figure 2.9. The label of the state variable (which can precise the type of characteristic represented by the state variable, residue type, residue number etc.) can be optionally written either within the *macromolecule* container, or next to the border of the state variable container, or within the state variable's container itself.

A *macromolecule* can also carry one or several *units of information* (Section 2.3.10). The units of information can characterise 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. The center of the bounding box of a *unit of information* is located on the midline of the border of the macromolecule.

A *macromolecule* may also carry a *clone marker* (see section 2.3.12)

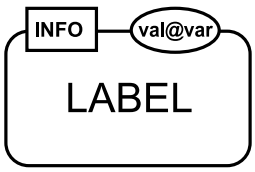


Figure 2.9: *The Process Diagram glyph for* macromolecule.

2.3.4 Glyph: *Genetic Entity*

The *genetic entity* 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 diagram.

SBO Term:

SBO:0000354 ! genetic entity

Container:

A *genetic entity* is represented by a rectangular container whose bottom half has rounded corners, as shown in Figure 2.10 on the next page.

Label:

The identity of a particular *genetic entity* 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 *genetic entity* can carry state variables (Section 2.3.11) that add information about its precise state. The state of a genetic entity is therefore defined as the vector of all its state variables. A state variable is represented by an ellipsoid container, with the long axis of the ellipsoid placed on the border of the *genetic entity*'s container as illustrated

in Figure 2.10. The label of the state variable (type of characteristic, nucleotide number) can be optionally written either within the *genetic entity* container, or next to the border of the state variable container, or within the variable's container itself.

A *genetic information* can also carry one or several *units of information* (Section 2.3.10). These can characterise a *genetic entity*'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 of the genetic entity. The center of the bounding box of a *unit of information* is located on the midline of the border of the *genetic entity*.

A *genetic entity* may also carry a *clone marker* (Section 2.3.12).

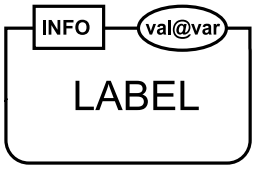


Figure 2.10: *The Process Diagram glyph for* genetic entity.

2.3.5 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.) *Pseudo-identical* here refers to 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 diagram. An example of this are the homologous subunits in a hetero-oligomeric receptor.

SBO Term:

SBO:0000286 ! multimer

Container:

A *multimer* is represented by two identical containers shifted horizontally and vertically and stacked one on top of the other. Figure 2.11 on the next page illustrates the glyph.

Label:

A *multimer* has no identity on its own. However, the first of the monomers carries an identifying label. The label placed 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 2.3.11). The state of a multimer is therefore defined as the vector of all its state variables. A state variable is represented by an ellipsoid container, with the long axis of the ellipsoid placed on the border of the top container of the *multimer* as illustrated in Figure 2.11 on the following page. Note that a *state variable* carried by a multimer actually applies to each of the constituent monomers individually. If instead a state variable is meant apply to the whole multimeric assembly, a *macromolecule* (Section 2.3.3) 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 2.4.1).

A *multimer* can also carry one or several *units of information* (Section 2.3.10). 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 2.3.3). An assembly containing unit of informations applying to the components, and others to the assembly should be represented by a *complex* (Section 2.4.1). The center of the bounding box of a *unit of information* is located on the midline of the border of the top multimer container. (See Figure 2.11 for an example.)

A *multimer* may also carry a *clone marker* (Section 2.3.12).

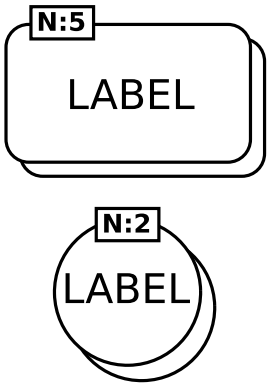


Figure 2.11: *The Process Diagram glyph for* multimer.

2.3.6 Glyph: *Source/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 a 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 conceptualy outside the scope of the diagram.

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 diagram. This is specially true if the diagram is rescaled, causing the spacing of elements in the diagram 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/sink* is represented by a glyph 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 2.12 illustrates this. The symbol should only be linked to one and only one edge in a diagram.

Label:

An *source/sink* does not carry any labels.

Auxiliary items:

An *source/sink* does not carry any auxiliary items.

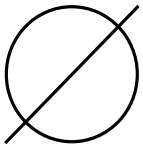


Figure 2.12: *The Process Diagram glyph for* source/sink.

2.3.7 Glyph: *Perturbation*

Biochemical networks can be affected by external influences. Those influences can be well-defined physical perturbations, 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 *perturbation* glyph.

SBO Term:

SBO:0000357 ! perturbation

Container:

A *perturbation* is represented by a modified hexagon having two opposite concave faces, as illustrated in Figure 2.13.

Label:

A *perturbation* 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 *perturbation* container. The label may spill outside of the container.

Auxiliary items:

A *perturbation* may carry a *clone marker* (Section 2.3.12).

Figure 2.13: *The Process Diagram glyph for* perturbation.

2.3.8 Glyph: *Observable*

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 diagram, SBGN defines the *observable* glyph.

SBO Term:

SBO:0000358 ! observable

Container:

An *observable* is represented by an elongated hexagon, as illustrated in Figure 2.14.

Label:

An *observable* 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 *observable* container. The label may spill outside of the container.

Auxiliary items:

An *observable* may carry a *clone marker* (Section 2.3.12).

Figure 2.14: *The Process Diagram glyph for* observable.

2.3.9 Glyph: *Tag*

A *tag* is a named handle, or reference, to another EPN (Section 2.3) or a container node (Section 2.4). *Tags* can be used to identify elements in SBGN *submaps* (Section 2.4.3).

SBO Term:

Not applicable.

Container:

A *tag* is represented by a rectangle fused to an empty arrowhead, as illustrated in Figure 2.15. The symbol should only be linked to one and only one edge (i.e., it should reference only one EPN or container).

Label:

A *tag* is idenfied 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 2.15: *The Process Diagram glyph for* tag.

2.3.10 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 SBGN *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 is oriented parallel to the border of the glyph being annotated by the *unit of information*.

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*. It may optionally be prefixed by a sequence of characters indicating the type of the information (e.g., taken from a controlled vocabulary). The controlled vocabularies predefined in SBGN Process Diagram 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.

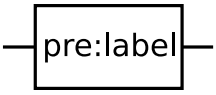


Figure 2.16: *The Process Diagram glyph for* unit of information.

2.3.11 Glyph: *State variable*

Biological entities can often exist under different states that affect their function. These "states" can arise for a variety of reasons. For example, macromolecules can be subject to post-synthesis modifications. In particular, residues of macromolecules (amino acids, nucleosides, or glucid residues) can be modified by covalent linkage to other chemicals. Another example of states is alternative conformations, for instance, the closed, open or desensitized conformations of a transmembrane channel, or the active or inactive forms of an enzyme. When an entity can exist under different states, the state of the whole entity (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 ellipsoid container, as shown in Figure 2.17. The ellipsoid's long axis should be tangent to the border of the glyph of the EPN being modified by the *state variable*.

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 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 2.18 on the next page below for some examples of problems arising if the identification of a state variable is located outside the state variable.

Auxiliary items:

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

Figure 2.17: *The Process Diagram glyph for* state variable.

A *state variable* does not necessarily have to be a boolean-valued variable. For example, an ion channel can possess several conductance states; a receptor can be inactive, active and desensitized; and so on. A *state variable* "ubiquitin" can also carry numerical values corresponding to the number of ubiquitin molecules present in the tail. However, a state variable can only take one defined value. Furthermore, 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". Note that "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 2.18 on the following page below, 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.

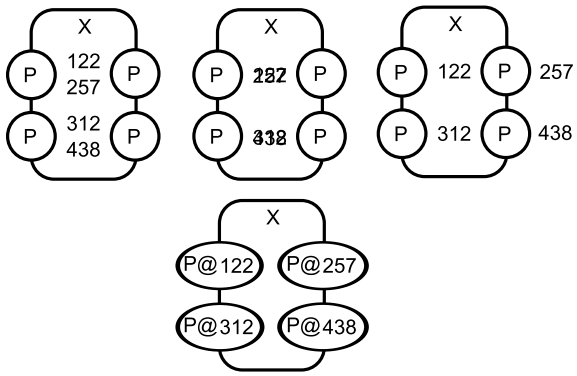


Figure 2.18: *(Top) Examples of incorrect* state variables. *(Bottom) Correct version.*

2.3.12 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 2.4.3). 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).

2.3.12.1 *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 2.19 illustrates this. The *clone marker* occupies the lower part of the EPN glyph and the marker's surface must not exceed 30% of the EPN's surface.

Label:

Not applicable.

Auxiliary items:

A *clone marker* does not carry any auxiliary items.

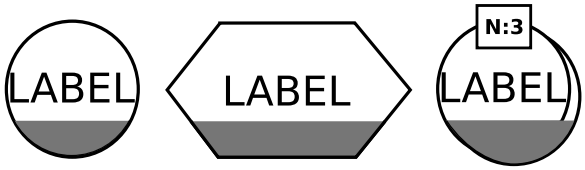


Figure 2.19: *The Process Diagram glyph for* clone marker.

2.3.12.2 *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

diagram. 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 marker's surface must not exceed 30% of the EPN's surface.

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 font color of the label and the color of the clone marker should contrast with one another. The label on a labeled *labeled clone marker* is mandatory.

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

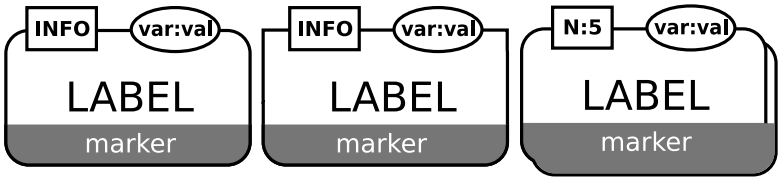


Figure 2.20: *The Process Diagram glyph for* labeled clone marker.

Figure 2.21 on the following page 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 diagrams by a reduction in the overall number of line crossings, but not always. In general, we advise that cloning should be used sparingly.

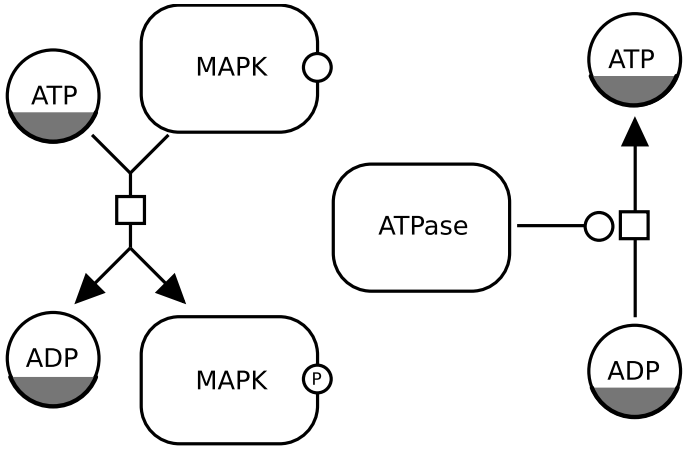


Figure 2.21: *An example of using cloning, here for the species ATP and ADP.*

2.3.13 Examples of complex EPNs

In this section, we provide examples of Entity Pool Node representations drawn using the SBGN Process Diagram Level 1 glyphs described above. First is a representation of the calcium/calmodulin kinase II, with phosphorylation on the sites threonine 286 and 306, as well as catalytic and autoinhibitory domains. This is shown in Figure 2.22. Note the use of *units of information* and *state variables*.

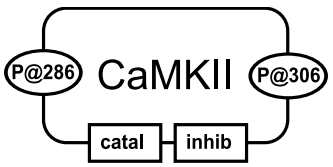


Figure 2.22: *An example representation of calcium/calmodulin kinase II.*

The next EPN example is a representation of 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. Figure 2.23 gives the diagram.

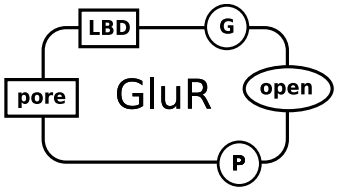


Figure 2.23: *An example of a glutamate receptor in the open state.*

2.4 Container nodes

Containers are SBGN constructions that contain one or several other SBGN constructs.

2.4.1 Glyph: *Complex*

A *complex* node represents a biochemical entity composed of other biochemical entities, whether macromolecules, simple chemicals, multimers, or themselves complexes.

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 octogonal 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.

Label:

The identification of a *named complex* is carried by a 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 2.3.11). 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 2.3.10). Those units of information can characterise a domain, such as a binding site. Particular *units of information* carry the material type and the conceptual type of the macromolecules. The center of the bounding box of a *unit of information* is located on the midline of the border of the complex. A *complex* may carry a *clone marker* (see section 2.3.12).

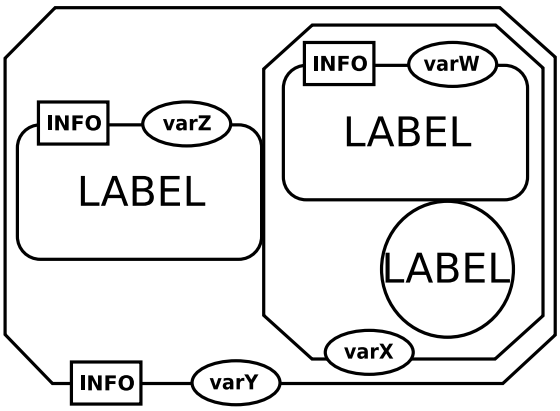


Figure 2.24: *The Process Diagram glyph for* complex.

2.4.2 Glyph: *Compartment*

In order to describe biochemical and cellular events, it is useful to define the notion of pools. A pool is an ensemble of participants that can be considered identical relatively to the events they are involved into. A compartment is a logical or physical subset of the event space that contains pools.

SBO Term:

SBO:0000289 ! functional 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 2.3.10. The center of the bounding box of a *unit of information* is located on the midline of the border of the compartment.

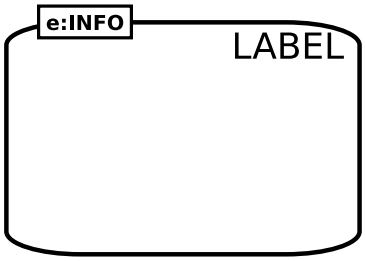
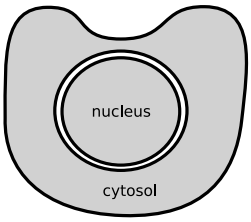
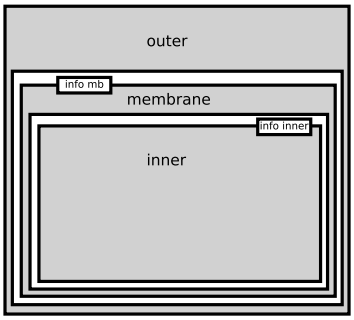


Figure 2.25: *The Process Diagram glyph for* compartment.

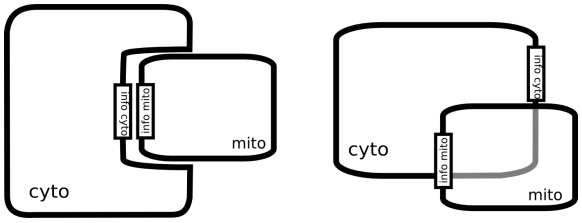
It is important to note that a compartment never contains another compartment, but may surround it. Two "adjacents" compartments are not separated by one line, but by **two** lines. The following example represents a cell made up of a nucleus surrounded by the cytoplasm.



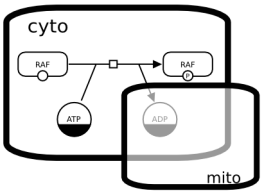
The following example represents three adjacent compartments. Two of the compartments carry units of information. Notice the fact that those units of information do not overlap several membrane boundaries.



To make drawing aestetically more pleasing and understandable, comparments are allowed to overlap each other visually, but it does not mean continment or surrounding. The following diagram represents two equivalent placement of compartments:



Overlapped (hidden) part of the compartment should not contain any object, which could be covered by overlapping compartment. The next diagram is not correct:



2.4.3 Glyph: *Submap*

A *submap* is used to encapsulate a certain number of processes (including all types of nodes and edges) within one glyph. The module hides its content to the users, and display only input terminals (or ports). A module is not equivalent to an omitted process (see section 2.5.2). In the case of an SBGN diagram available through a software tool, the content of a module may be available to the tool. A user could then ask the tool to expand the module, for instance by clicking on the module icon, that would expand the module within the same diagram (on the same canvas) or open it in a different canvas.

SBO Term:

to be determined

Container:

The *submap* is represented as a square box, to remind that it is fundamentaly 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 2.3) or containers (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 2.3) or containers (section 2.4).

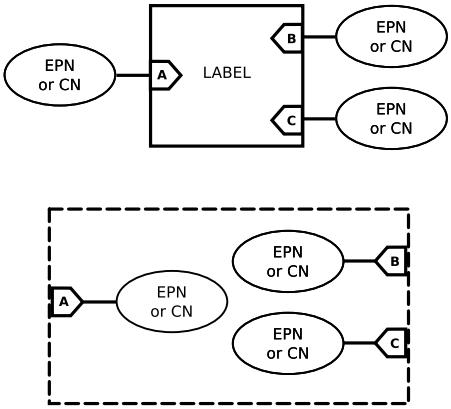
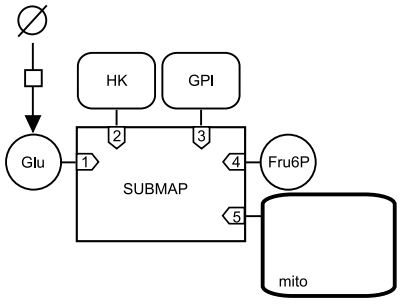
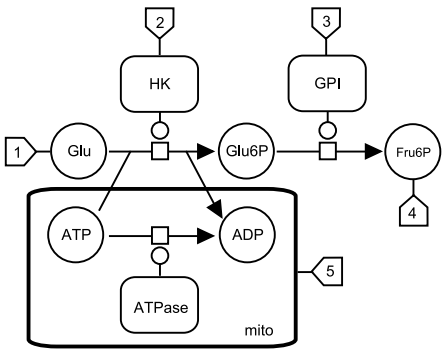


Figure 2.26: *The Process Diagram glyph for* submap.

The following diagram represents a *submap* that transforms glucose into fructose-6-phosphate. The *submap* carry 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.



The following diagram represents an unfolded version of the 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 to the compartment "mito" outside the submap. The compartment containing Glu6P is implicitely defined as the same than the compartment containing Glu and Fru6p. There is no ambiguity because if Glu and Fru6 were in different compartments, one of them should have been defined within the submap.



2.5 Process nodes

Process nodes represent processes that transform one or several EPNs into one or several different EPNs.

2.5.1 Glyph: *Transition*

A transition is a process transforming a set of states or entities in another set of states or entities.

SBO

SBO:0000167 ! reaction

origin

One or several *consumption* arcs (section 2.6.1) or one or several *production* arcs (section 2.6.2).

target

One or several *production* arcs (section 2.6.2).

node

A transition is represented by a square box. The arcs linking it to the state/entity nodes are attached to the center of opposite sides. The modulatory arcs point to the other two sides. A *transition* connected to *production* arcs on opposite sides is a reversible transition.

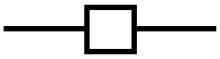
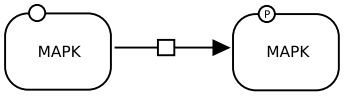


Figure 2.27: *The Process Diagram glyph for* transition.

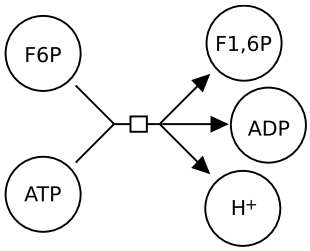
A transition is the basic process node in SBGN. It describes a process that transforms a given set of biochemical entities — macromolecules, simple chemicals or unspecified entities — into another set of biochemical entities. Such a transformation might imply modification of covalent bonds (conversion), modification of the relative position of constituents (conformational transition) or movement from one compartment to another (translocation).

A cardinality label may be associated with *consumption* (section 2.6.1) or *production* (section 2.6.2) arc indicating the stoichiometry of a process. This is required to eliminate ambiguity when the exact composition of the number of copies of the inputs or outputs to a reaction are ambiguous from the diagram.

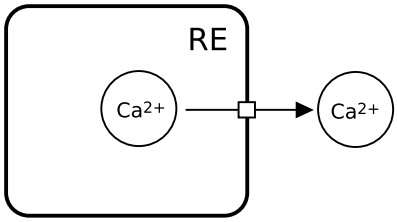
The following example illustrates the use of a *transition* node to represent the phosphorylation of a protein in a state transition diagram.



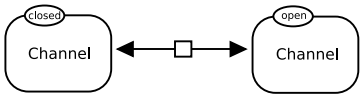
The following example illustrates the use of a *transition* node to represent an reaction between two reactants that generates three products.



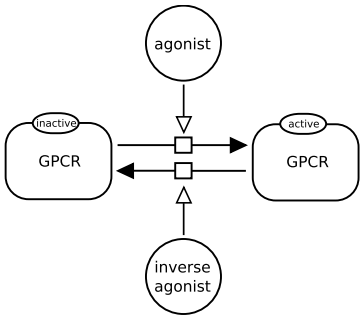
The following example illustrates the use of a *transition* node to represent a translocation. The large round-cornered rectangle represents a compartment border (see section 2.4.2).



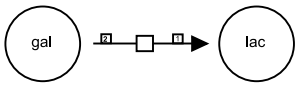
The following example illustrates the use of a *transition* node to represent the reversible opening and closing of an ionic channel in a state transition diagram.



When such a reversible process is assymetrically modulated, it should be represented by two different processes in State Transition. The following example illustrates the use of two *transition* 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.



The following example 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.



2.5.2 Glyph: *Omitted process*

Omitted processes are processes that are known to exist, but are omitted from the diagram 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 module.

SBO

To be determined

origin

One or several *consumption* arcs (section 2.6.1) or one or several *production* arcs (section 2.6.2).

target

One or several *production* arcs(section 2.6.2).

node

Omitted processes are represented as a transition containing a double slanted NW-SE line separated by an empty space.

Figure 2.28: *The Process Diagram glyph for* omitted.

2.5.3 Glyph: *Uncertain process*

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

SBO

To be determined

origin

One or several *consumption* arcs (section 2.6.1) or one or several *production* arcs (section 2.6.2).

target

One or several *production* arcs (section 2.6.2).

node

Uncertain processes are represented as a transition containing a questionmark.

Figure 2.29: *The Process Diagram glyph for* uncertain.

2.5.4 Glyph: *Association*

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

SBO

SBO:0000177 ! binding.

origin

More than one *consumption* arcss (section 2.6.1).

target

One *production* arc (section 2.6.2).

node

An *association* between several entities is represented by a filled disc.

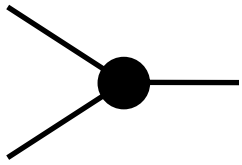
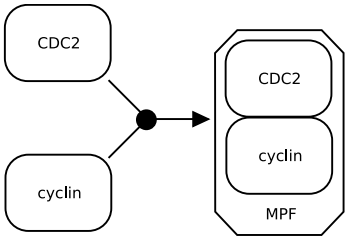
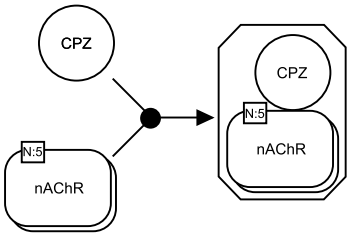


Figure 2.30: *The Process Diagram glyph for* association.

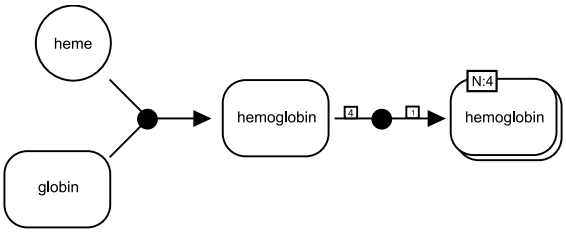
The following example illustrates the association of two cyclin and CDC2 kinase into the Maturation Promoting Factor in ST.



The following example illustrates the association of a pentameric macromolecule (a nicotinic acetylcholine receptor) with a simple chemical (the local anesthetic chlorpromazin) in an unamed complex, in an ST diagram.



An association does not obligatory results in the formation of a *complex*, but can also produce a *multimer*. See the formation of the hemoglobin below.



2.5.5 Glyph: *Dissociation*

The dissociation of a EPN into two EPNs represents the rupture of a non-covalent binding between the biological entities represented by those EPNs.

SBO

SBO:0000180 ! dissociation

origin

One *consumption* arc (section 2.6.1).

target

More than one *production* arc (section 2.6.2).

node

A *dissociation* between several entities is represented by two concentric empty discs. A simple empty disc could be, in some cases, confused with the *modulation* (section 2.6.3).

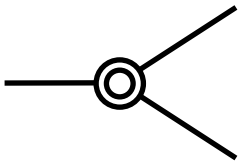


Figure 2.31: *The Process Diagram glyph for* dissociation.

2.6 Connecting arcs

Connecting arcs are lines that link EPNs and PNs together. The symbols attached to their extremities precise their semantics.

2.6.1 Glyph: *Consumption*

Consumption is the arc used to represent the fact that an entity only affects a process, but is not affected by it.

SBO

To be determined.

origin

Any EPN (section 2.3).

target

Any process node (section 2.5).

end-points

No particular symbol is used to represent a source.

A cardinality label may be associated with *consumption* (section 2.6.1) or *production* (section 2.6.2) arc 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 diagram. 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 of transition node, cardinality should be represented on all *consumption* and *production* arcs connected to that transition node to avoid misinterpretation.

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

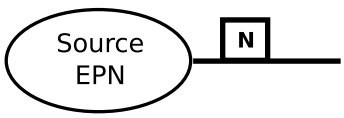


Figure 2.32: *The Process Diagram glyph for* consumption.

2.6.2 Glyph: *Production*

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

SBO

To be determined.

origin

Any process node (section 2.5), or module (section 2.4.3).

target

Any EPN (section 2.3).

end-points

The target extremity of a *production* carries a filled arrowhead.

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

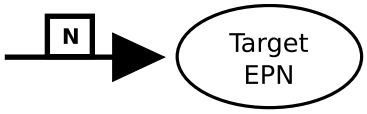
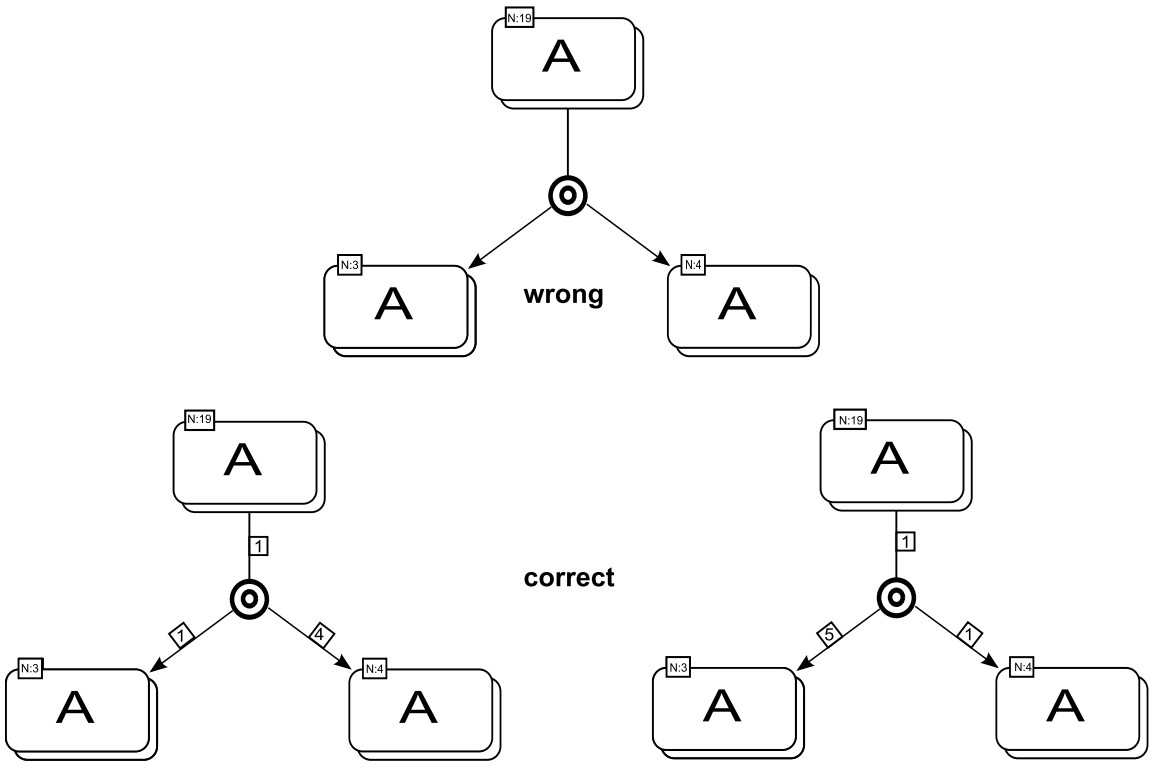


Figure 2.33: *The Process Diagram glyph for* production.

The following example illustrate the use of consumption/production arc cardinality labels to represent the stoichiometry of a process.



2.6.3 Glyph: *Modulation*

A modulation affects the flux of a process represented by the target transition. 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

SBO:0000168 ! control

origin

Any EPN (section 2.3) or any logical operator (section 2.7).

target

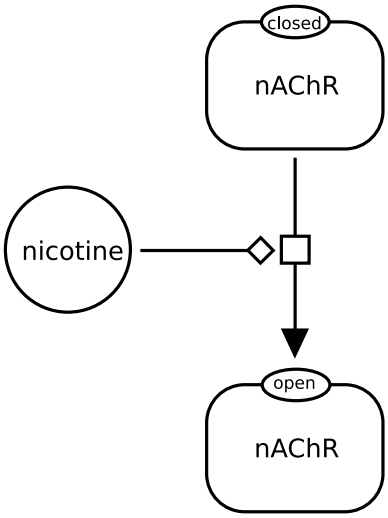
Any transition node (section 2.5).

end-points

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

Figure 2.34: *The Process Diagram glyph for* modulation.

The following example represents the effect of nicotine on the transition between closed and open states of nicotinic acetylcholine receptor. High concentrations of nicotine open the receptor while low concentrations can desensitize it without opening.



2.6.4 Glyph: *Stimulation*

A stimulation affects **positively** the flux of a process represented by the target transition. This stimulation can be for instance a catalysis or a positive allosteric regulation. Note that *catalysis* exists independently in SBGN, see section 2.6.5.

SBO

SBO:0000170 ! stimulation

origin
Any EPN (section 2.3) or any logical operator (section 2.7).

target
Any transition node (section 2.5).

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

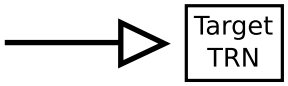


Figure 2.35: *The Process Diagram glyph for* stimulation.

2.6.5 Glyph: *Catalysis*

A catalysis is a particular case of stimulation, where the effector affects positively the flux of a process represented by the target transition. The positive effect on the transition is due to the lowering of the activation energy of a reaction.

SBO
SBO:0000172 ! catalysis

origin
Any EPN (section 2.3) or any logical operator (section 2.7).

target
Any transition node (section 2.5).

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

Figure 2.36: *The Process Diagram glyph for* catalysis.

2.6.6 Glyph: *Inhibition*

An inhibition affects **negatively** the flux of a process represented by the target transition. This inhibition can be for instance a competitive inhibition or an allosteric inhibition.

SBO
SBO:0000169 ! inhibition

origin
Any EPN (section 2.3) or any logical operator (section 2.7).

target
Any transition node (section 2.5).

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

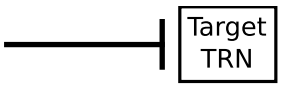


Figure 2.37: *The Process Diagram glyph for* inhibition.

2.6.7 Glyph: *Trigger*

A trigger effect, or absolute stimulation, is a stimulation that is necessary for a process to take place. A process modulated by a trigger can only occur when this trigger is active.

SBO

SBO:0000171 ! necessary stimulation

origin

Any EPN (section 2.3) or any logical operator (section 2.7).

target

Any transition node (section 2.5).

node

The target extremity of a *trigger* carries an open arrow (to remind that it is a *stimulation*) coming after a larger vertical bar.

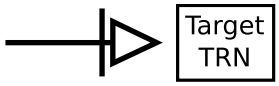
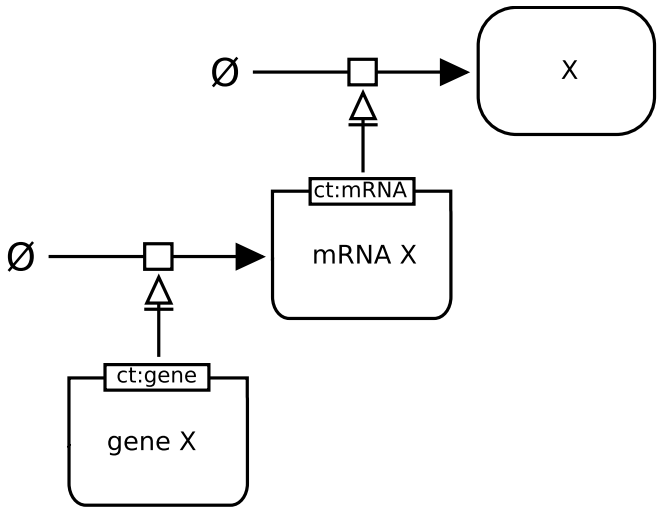
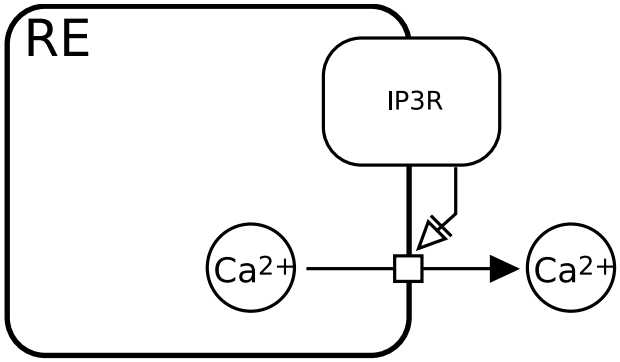


Figure 2.38: *The Process Diagram glyph for* trigger.

The example 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).



The example below describe 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 trigger intead represents this absolute stimulation.



2.6.8 Glyph: *Logic arc*

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

SBO
To be determined.

origin
Any EPN (section 2.3) or logical operator (section 2.7).

target
Any logical operator (section 2.7).

end-points
No particular symbol is used to represent a source.

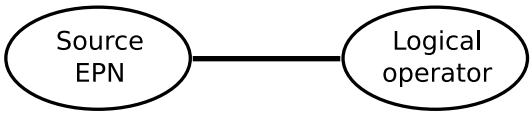


Figure 2.39: *The Process Diagram glyph for* logic arc.

2.6.9 Glyph: *Equivalence arc*

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

SBO
To be determined.

origin
Any EPN (section 2.3).

target
Tag (section 2.3.9).

end-points
No particular symbol is used to represent an *equivalence arc*.

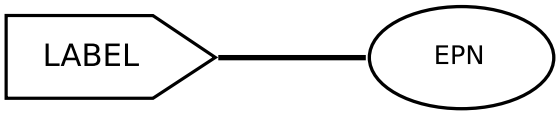


Figure 2.40: *The Process Diagram glyph for* Equivalence arc.

2.7 Logical operators

2.7.1 Glyph: *And*

SBO

SBO:0000173 ! and.

origin

More than one EPN (section 2.3) or logical operator (section 2.7).

target

Modulation (section 2.6.3), stimulation (section 2.6.4), catalysis (section 2.6.5), inhibition (section 2.6.6) or trigger (section 2.6.7) arcs.

node

and is represented by a circle carrying the word "AND".

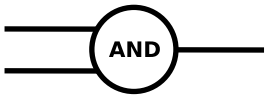


Figure 2.41: *The Process Diagram glyph for* and.

2.7.2 Glyph: *Or*

SBO

SBO:0000174 ! or.

origin

More than one EPN (section 2.3) or logical operator (section 2.7).

target

One modulation (section 2.6.3), stimulation (section 2.6.4), catalysis (section 2.6.5), inhibition (section 2.6.6) or trigger (section 2.6.7) arcs.

node

or is represented by a circle carrying the word "or".

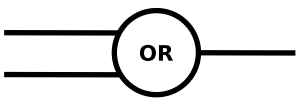


Figure 2.42: *The Process Diagram glyph for* or.

2.7.3 Glyph: *Not*

SBO

SBO:0000238 ! not.

origin

One EPN (section 2.3) or logical operator (section 2.7).

target

One modulation (section 2.6.3), stimulation (section 2.6.4), catalysis (section 2.6.5), inhibition (section 2.6.6) or trigger (section 2.6.7) arcs.

node

not is represented by a circle carrying the word "not".

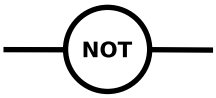


Figure 2.43: *The Process Diagram glyph for* not.

Chapter 3

Process Diagram grammar

3.1 Overview

In this chapter, we describe how the glyphs of SBGN can be combined to make a valid SBGN diagram. 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 semantcs, and it is essential if a reader is to understand a SBGN diagram without external help, and a writer is to create one that reflects his understanding of a biologiocal system.

In this section we start off by describing the concepts of the Process Diagram notation. 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 Diagram 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.

The Process Diagram in SBGN describes biological processes involving biological entities. An Entity Pool Node (section 2.3) such as a molecule is converted into other Entity Pool Nodes via a process.

It may be convenient to think of an SBGN Process Diagram as describing the flow of a fluid. In this analogy each Entity Pool Node represents a tank of fluid which may be emptied via a pipe (the consumption arc, section 2.6.1) that is connected to a valve (the process node, section 2.5), which in turn is connected to another pipe (the production arc, section 2.6.2) that fills another tank (Entity Pool Node). 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 2.3.6) and somewhere for it to drain away (the Sink).

3.3 The Conceptual Model

In order to formalise the conceptual representation described above and the semantic rules 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 diagram.

The model is separated into two layers. The Visual Layer (figure 3.1) 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). It is important in defining concepts such as Entity types and state variables that are implicit in the Visual Layer.

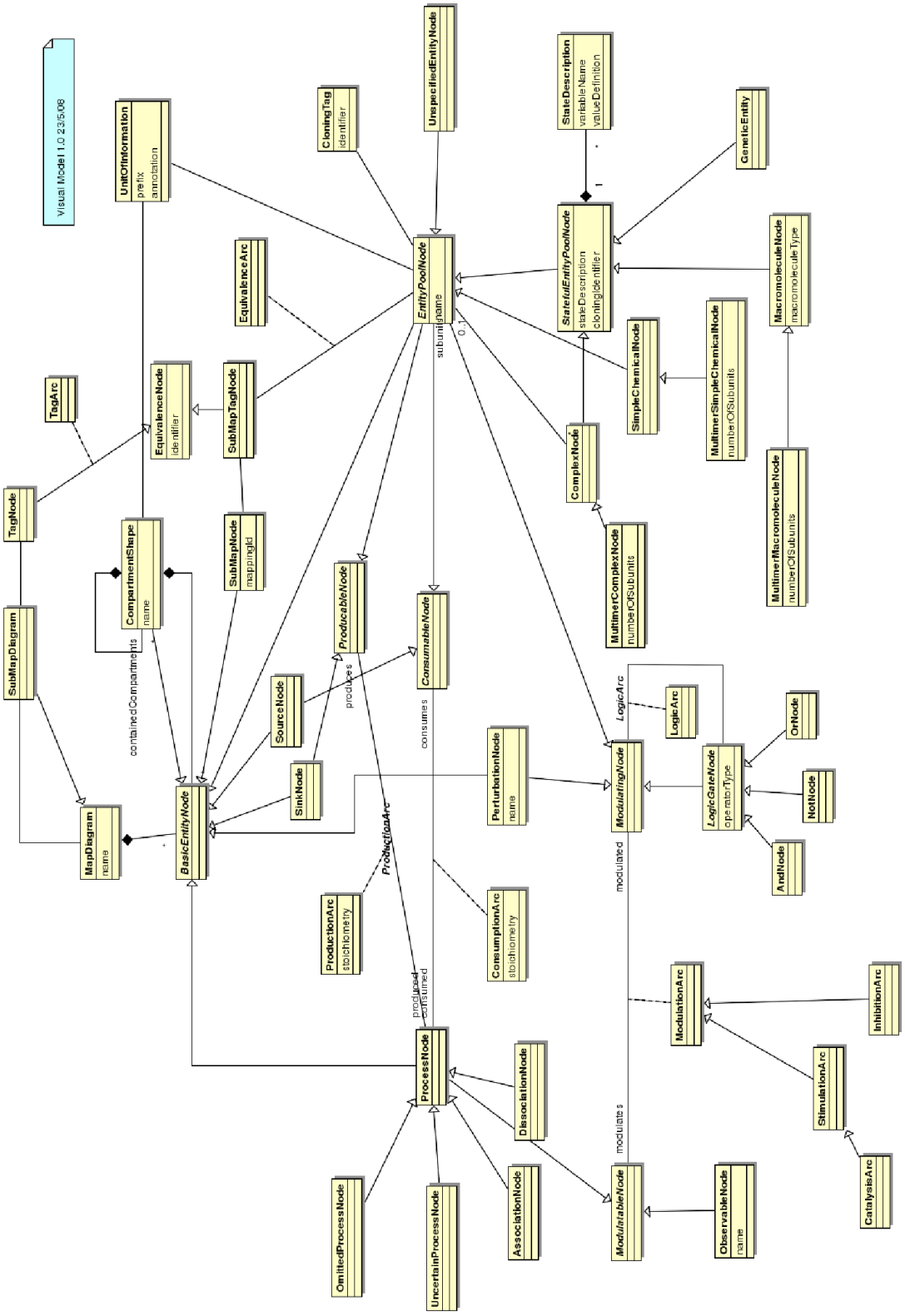


Figure 3.1: *UML model of SBGN visual representation. Class names shown in italics are Abstract classes and do not correspond to a Glyph. Classes in non-italicised script with the suffixes "Node" and "Arc" correspond to node and arc glyphs repectively.*

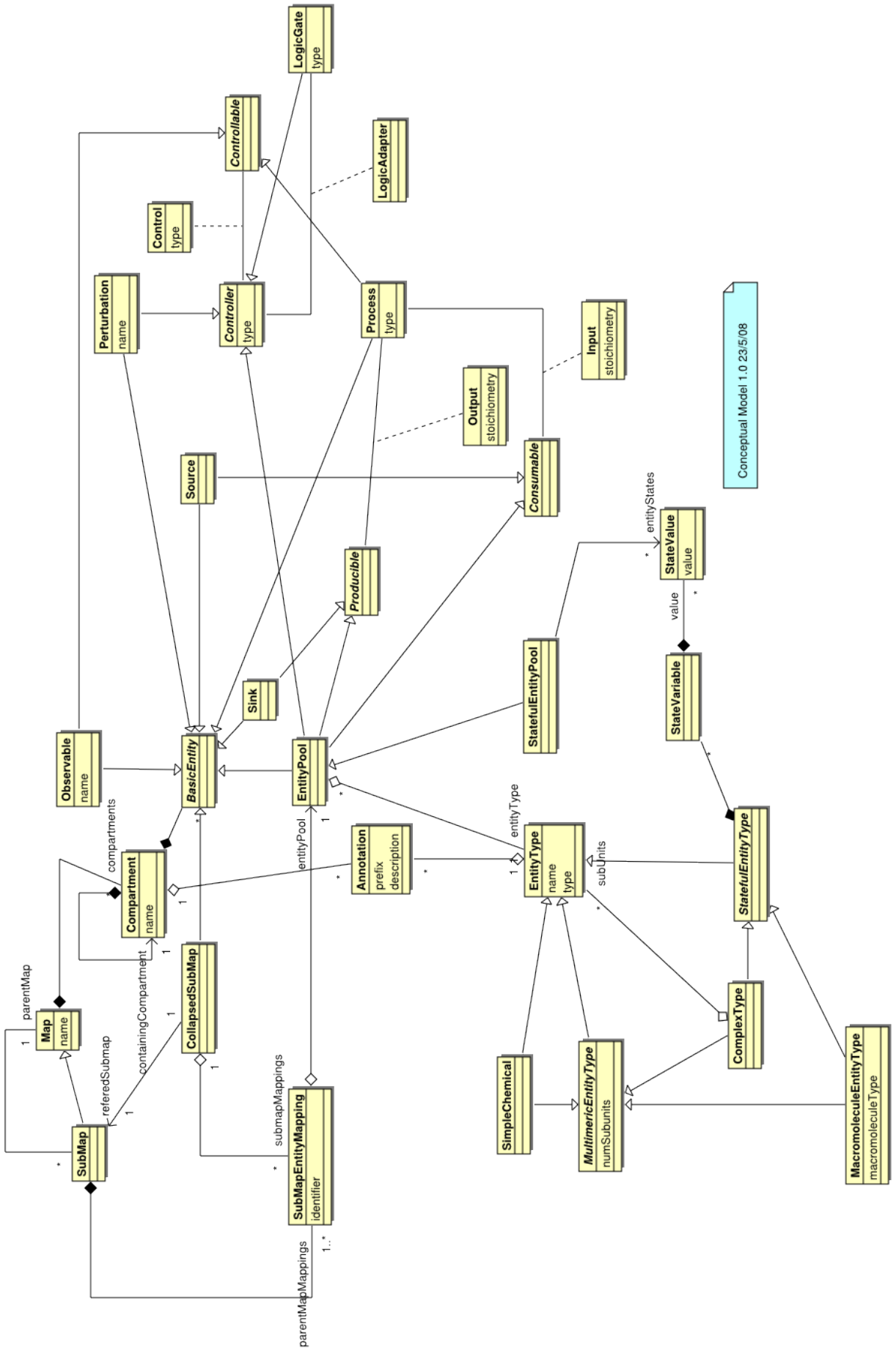


Figure 3.2: *UML model of SBGN conceptual representation. The main difference here is that the model is non-redundant with only one instance of an EntityPool with a given identity. Each class has a mapping back to the Visual Layer.*

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 module 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		
MapDiagram	name (I)	Name of the map.	N/A	The diagram is the visualisation of an SBGN map.
SubmapDiagram	mainMapName (I) name (I)	Name of the main map this this is a sub-map of. Name of the sub-map.	N/A	An SBGN diagram used to define an SBGN module.
BasicEntityNode			Abstract	The most fundamental node type. Any node that can be drawn on a MapDiagram inherits from this.
ConsumableNode			Abstract	Nodes that can be consumed and are connected at the start of a consumption arc.
ProducableNode			Abstract	Nodes that can produced and are linked to the end of a production arc.
ModulatableNode	modulatingNodes (I) {0 . . . *}	Any number of nodes that modulate this one.	Abstract	Nodes that can be modulated and so are linked to the end of a modulation arc.
ModulatingNode			Abstract	Nodes that can modulate modulateable nodes. They are linked to the start of a modulating arc.
CompartmentShape	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.
ObservableNode	name (I)	Name of the phenotype.	Observable	
PerturbationNode	name (I)	Name of the perturbation.	Perturbation	
MacromoleculeNode	As StatefulEntityPoolNode macromoleculeType (I)	 The type of the macromolecule. Currently there are no constraints on permitted types.	MacromoleculeNode	
GeneticEntityNode	As StatefulEntityPoolNode		GeneticEntity	
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	
Mulitmer-MacromoleculeNode	As MacromoleculeNode numberOfSubunits (I)	 The number of identical sub-units of this macromolecule.	Multimer-Macromolecule	Note that all sub-units have the *same* StateDescription values.
MultimerSimple-ChemicalNode	As SimpleChemicalNode numberOfSubunits (I)	 The number of identical sub-units of this simple chemical.	MultimerSimple-Chemical	
MultimerComplex-Node	As ComplexNode numberOfSubunits (I)	 The number of identical sub-units of this complex.	MultimerComplex	
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 value	Identity of the owning StatefulEntityPoolNode (not displayed, but implied by “ownership” of the glyph). The name of the “state variable” associated a StatefulEntityPoolNode. The definition of a set of values for the given variable.	State Variables	It is important to note that the StateDescription always defines a *set* of values.

continued on next page

continued from previous page

Name	Attributes		Glyph	Comment
	Name	Description		
ProcessNode	As ModulatableNode producableNodes (I) {1 . . . *} consumableNodes (I) {1 . . . *}	 The ProducableNodes linked to the process by a ProductionArc. The ConsumableNodes linked to the process by a ConsumptionArc.	Process	
OmittedProcessNode	As Process Node		Omitted Process	
UncertainProcessNode	As Process Node		Uncertain Process	
AssociationNode	As Process Node		Association	
DissociationNode	As Process Node		Dissociation	
LogicGateNode	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 LogicGateNode connected via a LogicArc.	Abstract	Not a glyph but a placeholder for the common behaviour of all logic gates.
ANDNode	As LogicGateNode		AND	
ORNode	As LogicGateNode		OR	
NOTNode	As LogicGateNode		NOT	
SubMapNode	subMapName (I) compartment (R)	The name of the sub-map 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	
SubMapTagNode	subMapName (I) identifier (I)	Name of submap that owns submap tag. A name that uniquely identifies the tag on the submap.	SubMapTag	
ProductionArc	processNode (I) producableNode (I)	The ProcessNode that this is the output of. The ProducableNode that is the product of the process.	Production	It is assumed that the arc is unidirectional.
ConsumptionArc	consumableNode (I) processNode (I)	The ConsumableNode that is the input to the process. The ProcessNode that this is the input to.	Consumption	It is assumed that the arc is unidirectional.
ModulationArc	modulatingNode (I) modulatedNode (I)	The ModulatingNode that is the source of this link. The ModulatableNode that is controlled (modulated) by this arc.	Modulation	
StimulationArc	As ModulationArc		Stimulation	
CatalysisArc	As ModulatioArc		Catalysis	
InhibitionArc	As ModulationArc		Inhibition	

continued on next page

continued from previous page

Name	Attributes		Glyph	Comment
	Name	Description		
TriggerArc	As ModulationArc		Inhibition	
LogicArc	modulatingNode (I) logicGate (I)	The ModulatingNode that is the source of this link. The logicGateNode 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).
TagArc	tag (I) taggedEPN (I)	The tag assigned to with the EPN. The EPN assigned to the tag.	Tag	
EquivalenceArc	subMapTag (I) EPN (I)	The tag assigned to with the EPN. The EPN assigned to the submap 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 diagram.	MapDiagram
SubMap	name (I) mainMapName (I)	The name of the submap. The name of the map that refers to this submap.		SubMapDiagram
Compartment	name (I) containingCompartment	The name of the compartment. The compartment that contains this compartment, if any.		CompartmentShape
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
Producible			Something that can be produced by a process.	ProducibleNode
Consumable			Something that can be consumed by a process.	ConsumableNode
EntityType	Name (I) type (I)	The name of the EntityType. Type of the Entity. One of: SimpleChemical, UnspeciedEntity, Complex, Macromolecule, and GeneticEntity.	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		
	annotations {0 . . . *}	Annotation must be associated with the type and be general to all Entity Pool Node. If it were associated with the Sprecies then this would be synonomous with a state 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.		Implicitly defined by a MacromoleculeNode and MultimerMacromoleculeNode
StateVariable	name (I) domainValues {1 . . . * }	The name of the state variable. The domain of the StateVariable, i.e. the set of possible values it can hold. This is required because an StatefulEntityPool is assigned a set of state values for a given StateVariable.	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 “regexp” used in the StateDescription.
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 StateVariables of the associated StatefulEntityType.		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 EntityType 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, DossociationNode, AssociationNode, OmittedProcessNode, UndertainProcessNode.

continued on next page

continued from previous page

Name	Attributes		Description	Visual Mapping
	Name	Description		
Output	process (I) product (I)	The process the is the output of. The Entity Pool Node that is the product of the process.		ProductionArc
Input	substrate (I) process (I)	The Entity Pool Node that is the input to the process. The process that this is the input of.		ConsumptionArc ConsumptionArc
Control	type (R)	The type of the control. One of Modulation, Stimulation, Inhibition, Trigger, Catalysis or LogicGate.		ModulationArc
LogicGate	type (R)	The type of LogicGate. One of AND, OR, NOT.		LogicGateNode
LogicAdapter			The LogicGate 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
Perturbation	name	Name of the perturbation.		PerturbationNode
Observable	name	Name of the phenotype.		ObservableNode
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.	SubMapTagNode and TagNode

3.4 Syntax

The syntax of SBGN State Transition diagrams is defined in the form of an incidence matrix. An incidence matrix has arcs as rows and nodes as colums. 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 PNs.

3.4.1 SN connectivity definition

Arc\SN	*macromolecule*	*simple chemical*	*unspecified entity*	*tag*	*source/sink*	*perturbation*	*observable*	*multimer*	*complex*	*genetic entity*	*submap*
consumption	I	I	I		I			I	I	I	
production	O	O	O		O			O	O	O	
modulation	I	I	I			I	O	I	I	I	
stimulation	I	I	I			I	O	I	I	I	
catalysis	I	I	I			I	O	I	I		
inhibition	I	I	I			I	O	I	I	I	
trigger	I	I	I			I	O	I	I	I	
logic arc	I	I	I					I	I	I	
equivalence arc	I	I	I	O				I	I	I	O

3.4.2 PN connectivity definition

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

3.4.3 Containment definition

There are two EPN types which allows containment in SBGN: *compartment* and *complex*. Next table describe relationship between EPN/PN elements of SBGN and these two containers. Plus sign means that the element is able to be containde within container. Empty cell means containment is not allowed.

EPN/PN\Containers	*compartment*	*complex*
compartment	+	+
complex		+
macromolecule	+	+
simple chemical	+	+
multimer	+	+
complex	+	+
unspecified entity	+	
source/sink		
transition	+	
omitted process	+	
uncertain process	+	
module	+	
association	+	
dissociation	+	
and	+	
or	+	
xor	+	
not	+	

3.4.4 Syntactic Rules

Indicence matrix, defining main part of the syntax, is too permissive. Additional rules should be defined to make syntax definition more precise.

1. EPNs. That rules are applicable to all EPNs (2.3)
   (a) If *macromolecule* has more than one *state variable*, all *state variables* should have a name;
   (b) All *state variables* of the *macromolecule* should have different names;
   (c) *Complex* should consists of different EPNs. If two or more elementns of the complex are identical they should be replaced by multimer.
   (d) Identity includes not only name of the element but its state also.
2. Sink/Source.
   (a) *source/sink* is allowed to have only one link attached to it. It could be either *consumption* or *production* link.
3. Process. That rules are applicable to all PNs (2.5)
   (a) PN should have nonzero number of *consumption* links.
   (b) PN should have nonzero number of *production* links.
   (c) All substrates of the transition should be different. If several copies of the same EPN are involved in the process, cardinality label of *consumption* arc should be used.
   (d) All producs of the transition should be different. If several copies of the same EPN are produced in the process, cardinality label of *production* arc should be used.
   (e) Once cardinality label set to one arc of the PN all other arcs should make their cardinality visible, even if it is undefined. In a case cardinality is undefined or unknown "?" should placed as cardinality label.
   (f) The EPN is not allowed to be a substrate and product of the same process. This rule is applicable to cloned EPNs as well.

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

(h) PN should have only one *Trigger* arc connected to it. If there more than one trigger is known for the process several PNs should be drawn.

4. *Association*

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

   (b) If *asociation* produces *complex*, than number of *consumption* arcs should be two or more.

   (c) If *association* produces *multimer*, than the only one *consumption* arc is allowed. 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*.

5. *Dissociation*

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

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

   (c) If *multimer* is splitted by *dissociation* process, than one or more *production* arcs should be connected to the process node.

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 figure 3.3.

Table 3.3: *Namespace scope definitions.*

Namespace Scope	Entities affected	Notes
MapDiagram	CompartmentShape, SubMapDiagram, EquivalenceNode	
CompartmentShape	BasicEntityNode	If no Compartment is drawn then all BasicEntityNodes are assumed to belong to an invisible "default" compartment.
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 CloningTag 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
CompartmentShape	N		
SimpleChemicalNode	Y	Clone Marker	
UnspecifiedEntityNode	Y	Clone Marker	
SourceNode	N		
SinkNode	N		
Perturbation	Y		CloneMarker
Observable	Y		CloneMarker
MultimerChemicalEntity	Y	Clone Marker	
StatefulEntityPoolNode	Y	Clone Label	
MacromoleculeNode	Y	As StatefulEntityPoolNode	
MultimerMacromoleculeNode	Y	As StatefulEntityPoolNode	
GeneticEntityNode	Y	As StatefulEntityPoolNode	
ComplexNode	Y	As StatefulEntityPoolNode	
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	As ProcessNode	
ANDNode	Y	As LogicalOperatorNode	
ORNode	Y	As LogicalOperatorNode	
NOTNode	Y	As LogicalOperatorNode	

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 diagram 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 WRITE-ME.

3.5.5 Compartments

The layout of compartments in an SBGN diagram 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 affect is therefore additive.
7. At most one trigger can be assigned to a process. Two triggers would imply an implicit Boolean AND or OR operator. For clarity only one trigger can be assigned to a process and such cmbinations must explicitly used 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 meaning fully 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 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.

3.5.7.1 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 SubMapTagNode 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 SubMapTagNode must be identical.
2. The EntityPoolNodes must be identical.

3.5.7.2 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.

3.5.7.3 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.

Chapter 4

General layout of a graph in SBGN

4.1 Introduction

The previous chapters describe the appearance and meaning of objects. Objects are, depending on the diagram (state transition, activity flow, entity relationship): state, entity, transition and container nodes as well as connecting arcs (edges). The objects of a diagram 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 to give a congenial appearance of the diagrams may reveal new insights. The arrangement of objects in a diagram is called a *layout*.

SBGN diagrams should be easily recognisable not only by the used glyphs but also by the general style of the layout. However, the arrangement of the objects 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 SBGN diagrams, 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 the appendix there is a list of additional suggestions which may help in producing aesthetically more pleasant layouts which may be easier to understand.

The layout guidelines are independent of the method used to produce the diagram and apply to both manually drawn diagrams as well as diagrams produced by a dynamic layout algorithm. The guidelines do not deal with interaction aspects (e. g. the effect of zooming) but apply to diagrams in standard view.

Please note that the color of objects do not carry any meaning in SBGN. Although one can use colors to emphasize part of a diagram or encode additional information, the meaning of the diagram should not depend on the colors. Furthermore, objects can have different sizes and size is also meaningless in SBGN. For example, a transition 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 diagram.

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. 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. Compartment borders are not allowed to overlap and are only allowed to touch each other if this is meaningful, e. g. ER and ER membrane. Modules are not allowed to overlap. Compartments and modules can contain other nodes including compartments and/or modules.

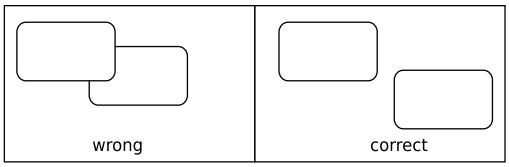


Figure 4.1: *Nodes must not overlap.*

4.2.1.2 Node-edge crossing

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

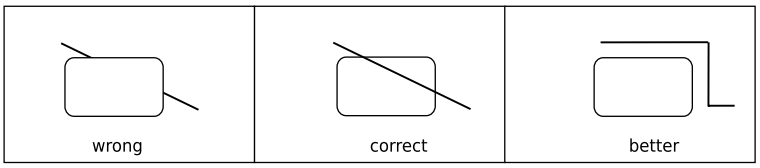


Figure 4.2: *If an edge crosses a node, the edge must be drawn on top of the node.*

4.2.1.3 Node border-edge overlaps

Edges are not allowed to overlap the border lines of nodes.

4.2.1.4 Edge-edge overlaps

Edges are not allowed to overlap. This include touching of edges. Furthermore, an edge is neither allowed to cross itself nor to cross a boundary of nodes 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.

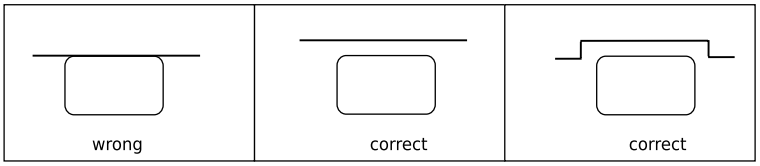


Figure 4.3: *Edges must not overlap node borders.*

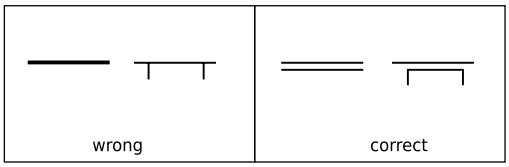


Figure 4.4: *Edges must not overlap.*

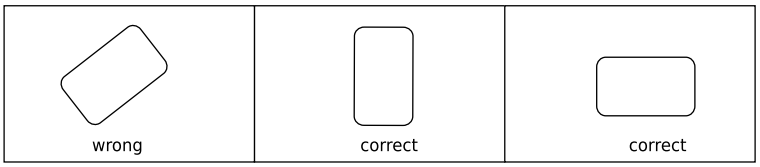


Figure 4.5: *The node orientation must be horizontally or vertically.*

4.2.1.6 Node-edge connection

The arcs linking the square box of a transition to the state/entity nodes are attached to the center of opposite sides. The modulatory arcs point are attached to the other two sides, but not necessarily all to the center as several modifiers can touch the same transition node. A transition connected to glyph-production arcs on opposite sides is a reversible transition.

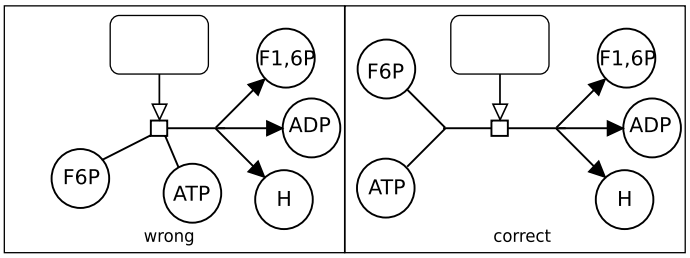


Figure 4.6: *Arcs between a transition and the state/entity nodes must be attached to the center of opposite sides, modulatory arcs must be attached to the other two sides.*

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 transition has all participants in the same compartment the transition node and all edges/arcs have to be in this compartment. If a transition has participants in at least two different compartments, the transition node has to be either in a compartment where the transition has at

least one participant or in the empty space.

4.2.1.10 Interaction edge

Other arcs ending at an interaction edge must end in the interaction zone. The dots representing the connection of an arc with the interaction zone should be equally distributed. The interaction zone consisting of two connected edges is either a circle or an enclosed area with the two edges parallel.

4.2.2 Recommendations

Recommendations are rules which should be followed if possible to produce layouts which look similar (better formulation) and 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

Crossings between edges should be minimized.

4.2.2.4 Units of information

Units of information should not hide the structure of the corresponding node.

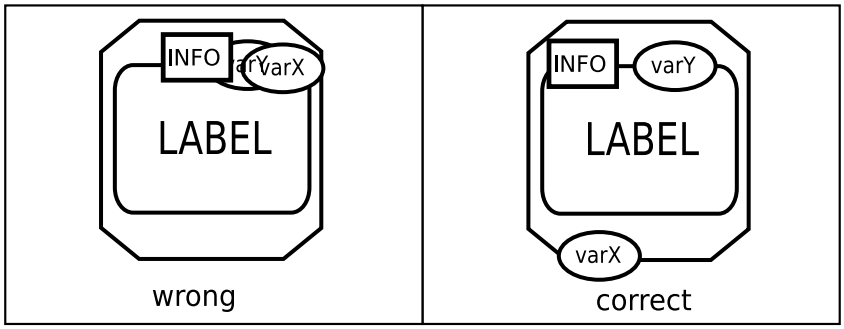


Figure 4.7: *Units of information should not overlap with any other element.*

4.2.3 Additional suggestions

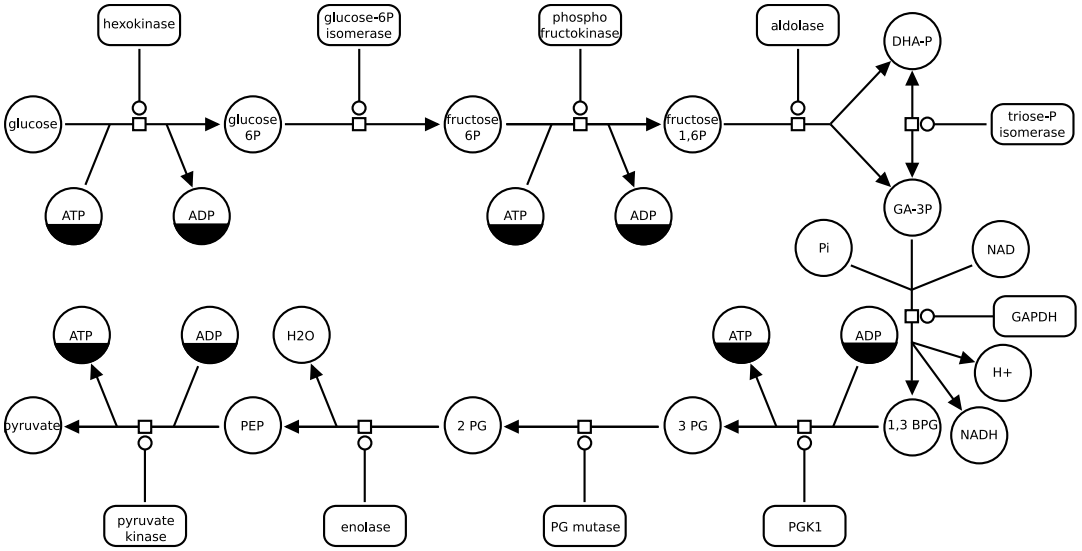
Here is a list of additional layout suggestions which may help in producing aesthetically more pleasing layouts which may be easier to understand.

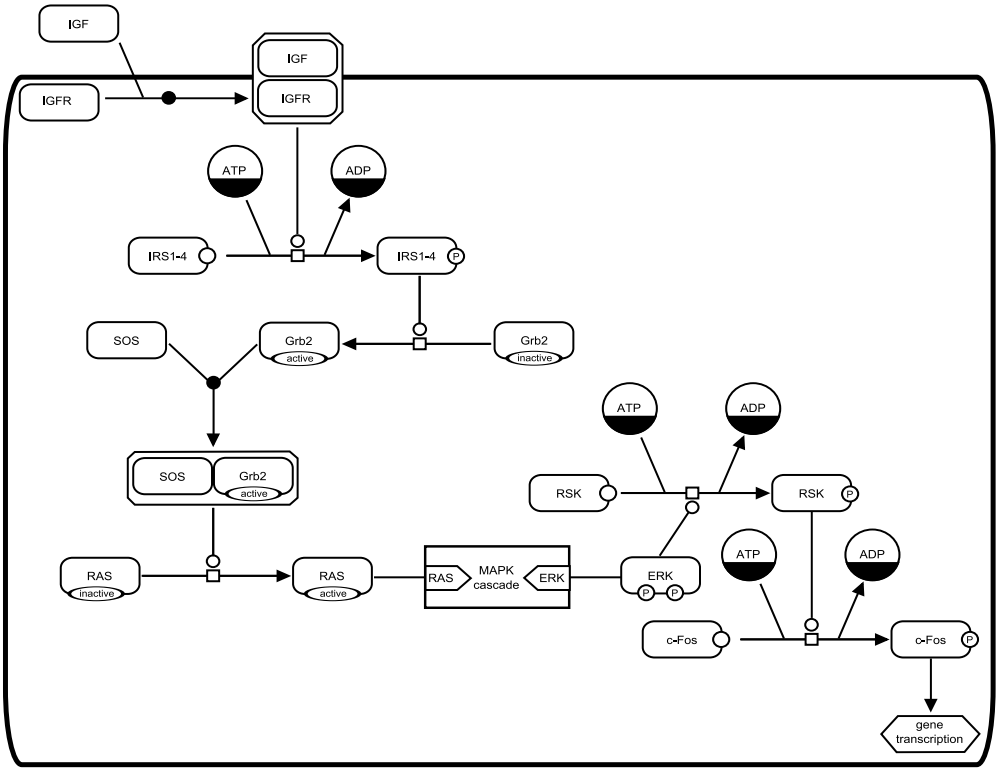
- Angle of edge crossings: If edge crossings are not avoidable edges should cross with an angle close to 90 degrees.
- Placement of substrates and products of a transition: Substrate and product nodes should be placed on different sides of the transition node.
- Drawing area and width/height ratio: The 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 and the edge length should be minimized.
- Number of edge bends: Edges should be drawn with as few bends as possible.
- Similar and symmetric parts: Similar parts of a diagram should be drawn in a similar way, and symmetric parts should be drawn symmetrically.
- Proximity information: Related elements (e. g. nodes connected by a transition 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: Different compartments should have different background shade or color.

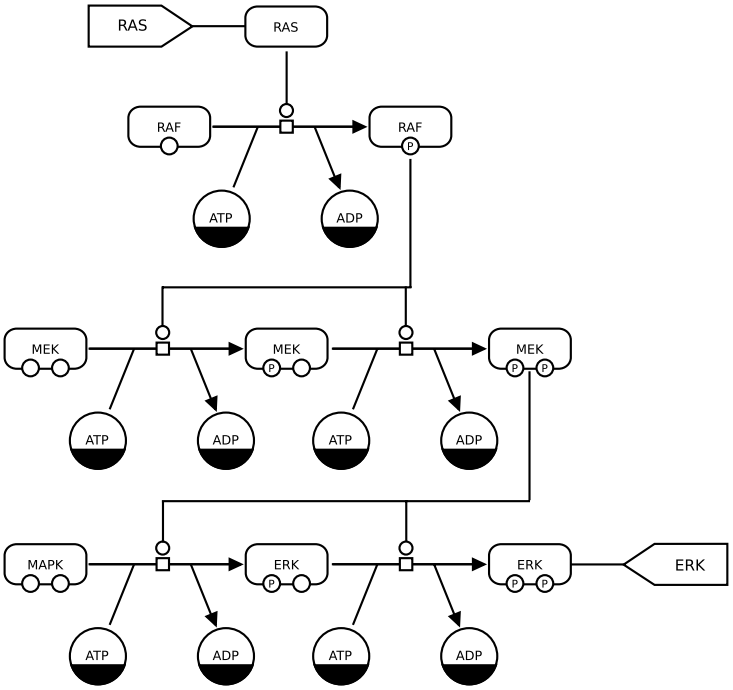
Appendix A

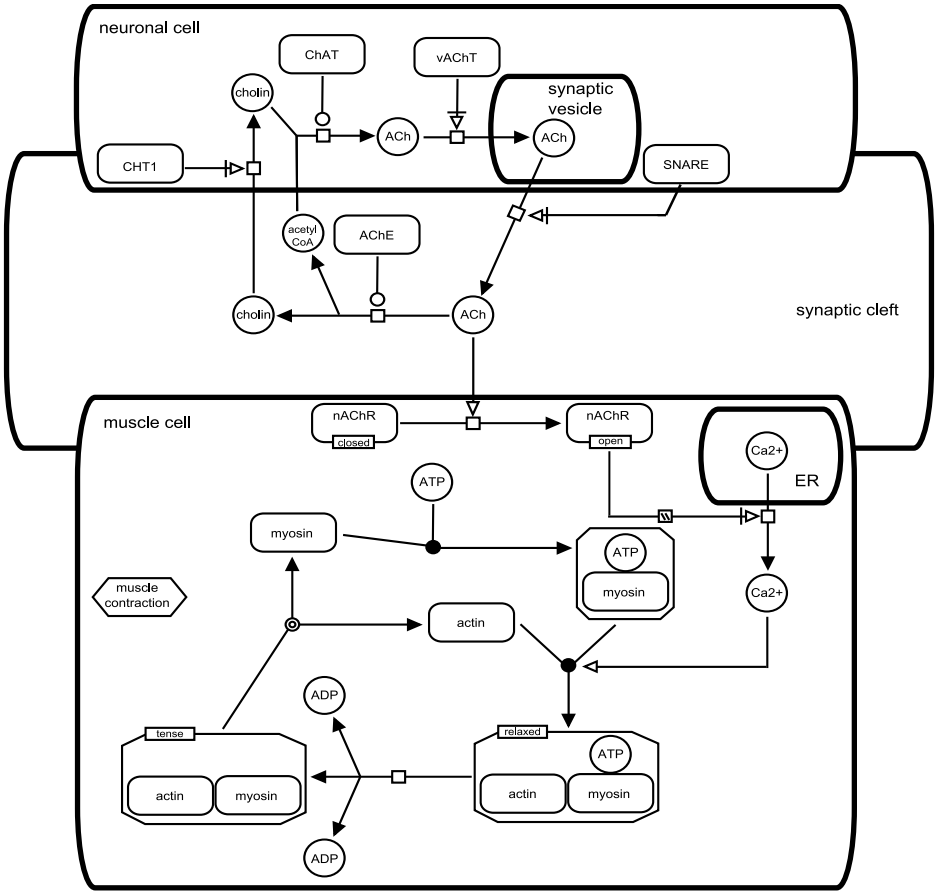
Complete examples

The following diagrams present complete examples of SBGN Process Diagrams. [NB: We are working on the legends of those maps. Another example is coming, describing a Gene Regulatory Network]




IGF
IGF
IGFR
IGFR
ATP
ADP
IRS1-4
IRS1-4
P
SOS
Grb2
active
Grb2
inactive
SOS
Grb2
active
ATP
ADP
RSK
RSK
P
RAS
inactive
RAS
active
RAS
MAPK
cascade
ERK
ERK
P
P
ATP
ADP
c-Fos
c-Fos
P
gene
transcription


RAS
RAS
RAF
RAF
P
ATP
ADP
MEK
MEK
MEK
P
P
P
ATP
ADP
ATP
ADP
MAPK
ERK
ERK
ERK
P
P
P
ATP
ADP
ATP
ADP

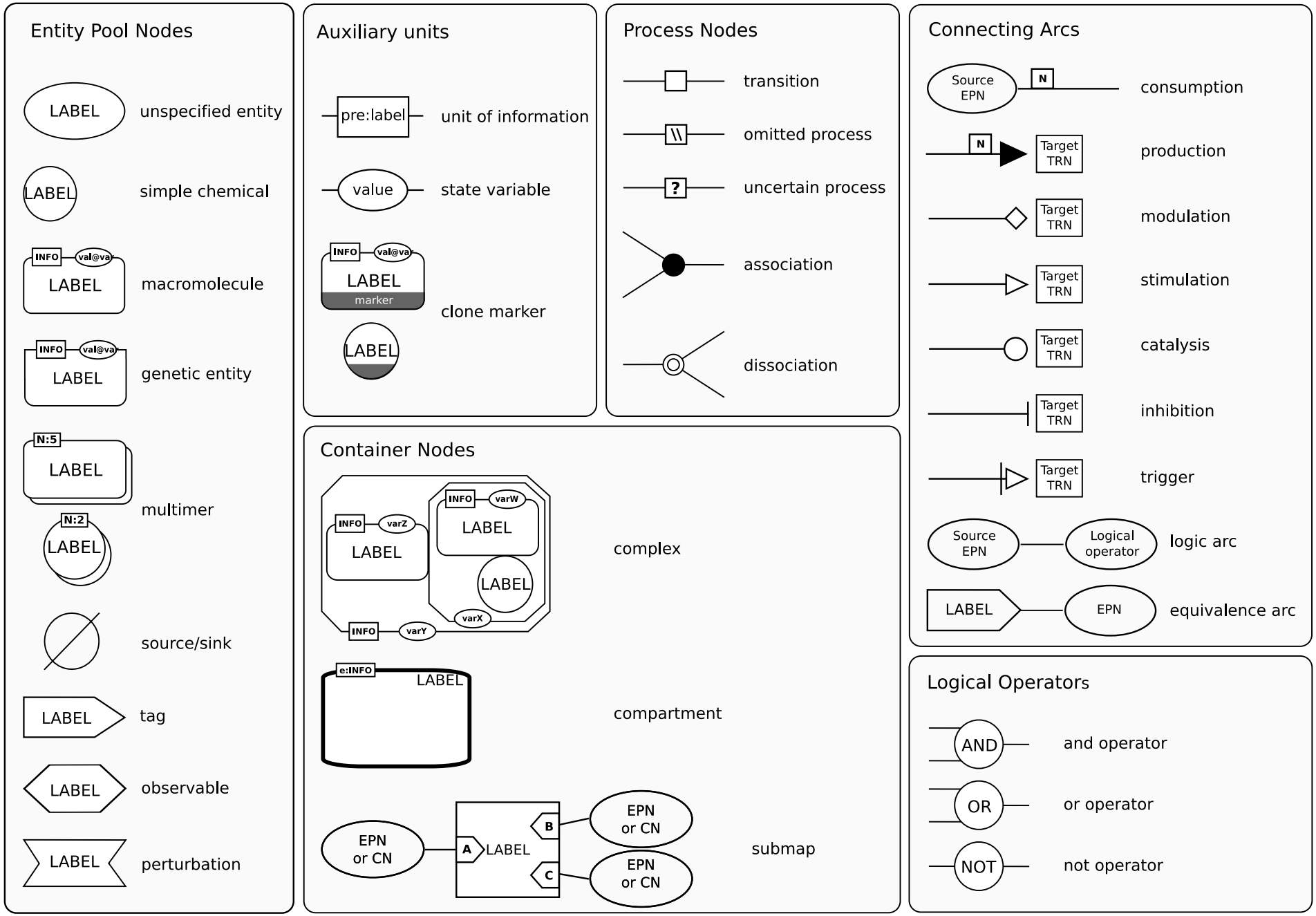

neuronal cell
ChAT
vAChT
cholin
synaptic vesicle
ACh
ACh
CHT1
SNARE
acetyl CoA
AChE
cholin
ACh
synaptic cleft
muscle cell
nAChR
closed
nAChR
open
Ca2+
ER
ATP
myosin
ATP
myosin
muscle contraction
actin
Ca2+
ADP
tense
relaxed
ATP
actin
myosin
actin
myosin
ADP

Appendix B

Reference card

Print this summary of SBGN symbols for a quick reference.

SYSTEMS BIOLOGY GRAPHICAL NOTATION REFERENCE CARD



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 Diagram 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 artifitial 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 Diagram 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 processus 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.

Bibliography

[1] Kurt W. Kohn. Molecular interaction map of the mammalian cell cycle control and dna repair systems. *Molecular Biology of the Cell*, 10(8):2703–2734, 1999.

[2] I. Pirson, N. Fortemaison, C. Jacobs, S. Dremier, J. E. Dumont, and C. Maenhaut. The visual display of regulatory information and networks. *Trends in Cell Biology*, 10(10):404–408, 2000.

[3] Daniel L. Cook, J. F. Farley, and S. J. Tapscott. A basis for a visual language for describing, archiving and analyzing functional models of complex biological systems. *Genome Biology*, 2(4):research0012.1–research0012.10., 2001.

[4] Ron Maimon and Sam Browning. Diagrammatic notation and computational structure of gene networks. In Hiroaki Kitano, editor, *Proceedings of the 2nd International Conference on Systems Biology*, pages 311–317, Madison, WI, 2001. Omnipress.

[5] Hiroaki Kitano. A graphical notation for biochemical networks. *BioSilico*, 1:169–176, 2003.

[6] Hiroaki Kitano, Akira Funahashi, Yukiko Matsuoka, and Kanae Oda. Using process diagrams for the graphical representation of biological networks. *Nature Biotechnology*, 23(8):961–966, 2005.

[7] Stuart L. Moodie, Anatoly A. Sorokin, Igor Goryanin, and Peter Ghazal. A graphical notation to describe the logical interactions of biological pathways. *Journal of Integrative Bioinformatics*, 3:36, 2006.

[8] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, A. P. Arkin, B. J. Bornstein, D. Bray, A. Cornish-Bowden, A. A. Cuellar, S. Dronov, E. D. Gilles, M. Ginkel, V. Gor, I. I. Goryanin, W. J. Hedley, T. C. Hodgman, J.-H. Hofmeyr, P. J. Hunter, N. S. Juty, J. L. Kasberger, A. Kremling, U. Kummer, N. Le Novère, L. M. Loew, D. Lucio, P. Mendes, E. Minch, E. D. Mjolsness, Y. Nakayama, M. R. Nelson, P. F. Nielsen, T. Sakurada, J. C. Schaff, B. E. Shapiro, T. S. Shimizu, H. D. Spence, J. Stelling, K. Takahashi, M. Tomita, J. Wagner, and J. Wang. The Systems Biology Markup Language (SBML): A medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, 2003.