# BCML Documentation

***Release 1.0***

**Luca Beltrame, Enrica Calura, Razvan Popovici**

August 04, 2010

# CONTENTS

This document describes the Biological Connection Markup Language, a data format and a framework to describe and analyze biological pathways.

For further information, see the *Introduction section*.

Contents:

CHAPTER

ONE

# INTRODUCTION

The **Biological Connection Markup Language** is an XML based data format to represent, annotate, filter and analyze biological pathways. It is implemented as an XML schema coupled with a group of utilities to manipulate and export the format. The use of an XML schema ensures flexibility from a computational perspective, and also the possibility of manipulation, representation, and adjustments of the data in a consistent way.

Pathways described in BCML can be used for computational analysis, or converted into a graphical representation useful for publication and data interpretation.

BCML is fully compliant to the Systems Biology Graphical Notation specification, as proposed by Le Novere *et al.* [LeNovere2009]. It is an implementation of the Process Description (PD) 1.1 specification.

BCML was developed as a joint effort of the Duccio Cavalieri group at the University of Firenze (Italy) and the Sorin Draghici group at the Wayne University (USA).

CHAPTER

TWO

# INSTALLATION

The BCML manual is divided into two parts: the XML schema which is needed to write files that are compliant to the specification, and the utilities that are needed to manipulate, validate and export the data.

## 2.1 XML Schema

To use the XML Schema, there is no need of installation. Simply copy the XSD and XML files from the schema archive in a directory of your choice and you will be already ready to use them.

## 2.2 Utility programs

BCML also offers a series of utilities to perform various operation on BCML schema compliant XML files. In order to use them, you need to have the Sun Java Runtime Environment (JRE) installed already, at least version 1.6. For Linux systems, the tools were only tested for Sun Java Virtual Machine.

Once this prerequisite is met, you need to download the utility archive (a ZIP file or a TGZ file, depending on your operating system), unpack it to a directory of your choice. The utility files are in the `bin` directory and can be used directly through the command prompt (Windows), a shell (Linux) or the Terminal (OS X). To uninstall the utilities, simply delete the folder.

## 2.3 Adding new annotations

Sometimes you may want to add additional annotations to the BCML-supplied definitions to build more complete pathway definitions: for example, adding new cell types. This is done through an additional program, `bcml_add_ontology`. `bcml_add_ontology` is a small program written in Python that simplifies the addition of new terms to the BCML schemas.

`bcml_add_ontology` has the following requirements:

- Python (2.5) or greater (http://www.python.org)
- The lxml Python module, 2.0 or greater (http://codespeak.net/lxml)
- The datamatrix Python module 0.9 or greater (http://pypi.python.org/pypi/datamatrix/0.9)

Installers for the major platforms are available on the linked web sites. Once the depenencies are satisfied, execute `bcml_add_ontology` as follows:

```
bcml_add_ontology -o <ontology file> -s <type> <source> <destination>
```

where `ontology file` refers to a text file containing the new terms to add, one per line, `type` indicates the category you want to add items to (e.g., ProvenIn, CellType), source is the full path to your `SBGN-ext.xsd` file (which contains the definitions) and `destination` the new file to be written.

To use the new XSD file, replace the old `SBGN-ext.xsd` file with the new one (make a backup copy first). Now you can use any XML editor to build your pathway with the new annotations.

To allow the new schema to be used in the tools (for example validation), you need to rebuild them (see below).

## 2.4 Rebuilding the utility programs

> **Warning:** The following instructions are for experienced users only.

If you make any modification to the schemas, you cannot use the pre-built utilities, and you need to rebuild them. Rebuilding them requires Apache Maven. Once you have installed Maven and its dependencies, download the source archive, and unpack it to a directory of your choice. Add your modified schema files (e.g. `SBGN-ext.xsd`) to the `sbgnpdschema/src/main/xsd/` directory, overwriting the existing ones. Notice that the names must be the same or rebuilding will fail Notice that the names must be the same or rebuilding will fail.

Lastly, Open a command prompt to that directory and type `make` (Linux/Unix) or `make.bat` (Windows) to build the new utilities (you willl find them in the `bin` directory once building is complete).

CHAPTER

THREE

# CREATING PATHWAYS IN BCML

As a pathway described in BCML is simply a text file, writing a BCML pathway requires no more than a common text editor. However, to ensure the writing of proper files and to increase efficiency, an XML editor is recommended. Also, an XML editor will make use of the BCML schema directly, preventing the creation of invalid elements. This will have the added advantage of reducing, if not eliminating completely, typing and validation errors.

Examples of XML editors that can be used to write BCML pathways include:

- XMLSpy (commercial)
- XMLSpear (freeware)
- Eclipse IDE (open source)
- Microsoft Visual Studio (free Express version, commercial professional and better).

Describing how to write a SBGN compliant pathway is outside the scope of the document: in any case, you should use the SBGN Process description specification as a reference on how to structure your pathway. More detailed information on BCML itself and what it supports are in the schema documentation. Each of the objects of SBGN PD has a homonym correspondent in BCML.

## 3.1 Writing annotations

Annotations are an extension of the schema that permit to associate specific findings and other facts to an element of a pathway. For example, you may want to associate the type of experiment where a specific interaction was recorded, or the Entrez Gene IDs related to a macromolecule.

The SBGN extension schema provides support for these kinds of annotations. For example, to annotate a macromolecule with a cell type in *Homo sapiens*:

```
<Macromolecule ID="my_gene">
...
        <Finding>
                <Organism>Homo sapiens</Organism>
                <CellType>Cancer cells</CellType>
        </Finding>
</Macromolecule>
```

Annotating findings enables the description of generic pathways. The generic pathways can be transformed to specific pathway; the elements not belonging to an organism or cell type of interest, can be removed with the bcml_filter tool.

Other valid information that can be added in Findings include for example PubMed IDs (using the `PMID` tag).

Additional annotations outside Findings can be used to indicate gene or protein identifiers, for example:

```
<Macromolecule ID="my_gene">
...
        <Organism name="HS">
                <annotation DB="EntrezGeneID" ID="148022"/>
        </Organism>
</Macromolecule>
```

In this case, we indicate the `EntrezGeneID` 148022 associated to `HS` (which is *Homo sapiens*) for this macromolecule. Annotation of the macromolecules with known database symbols is a facilitator of analysis techniques and enables representation of experiments within the pathway. The full list and description of available annotation features is in the schema documentation.

CHAPTER

FOUR

# VALIDATION, VISUALIZATION AND MANIPULATION OF BCML PATHWAYS

Once a BCML compliant file has been written, it needs to be checked. This process, called *validation* is used to ensure that:

1. The XML file itself is well-formed;
2. There are no elements that violate the SBGN specification;

Once a BCML file is valid, it can manipulated and used in different ways. These include converting a BCML file to a format suitable for graphical representation, filtering the file according to specific criteria, and exporting the entities to list.

## 4.1 Validating a BCML file

BCML files are checked for validity using the `bcml_check` utility. The program will ensure first that the file is valid from the point of XML consistency, then it will output any errors regarding SBGN compliance.

A validation run is triggered by running `bcml_check` as follows:

```
bcml_check --srcSBGN <source BCML file>
```

Where `source BCML file` is your BCML file. Any errors will be outputted to your screen, along with a brief description of where the error occurred, or what kind of violation was encountered.

`bcml_check` will also verify that the annotations have been inserted correctly and that the terms match the controlled vocabularies of the schema.

## 4.2 Viewing a BCML file

BCML files cannot be viewed directly. However, you can convert them to a graphical format (GraphML) and then visualize them with the yED graphics editor, that supports such a format. From there, you can export to widely-used graphics formats.

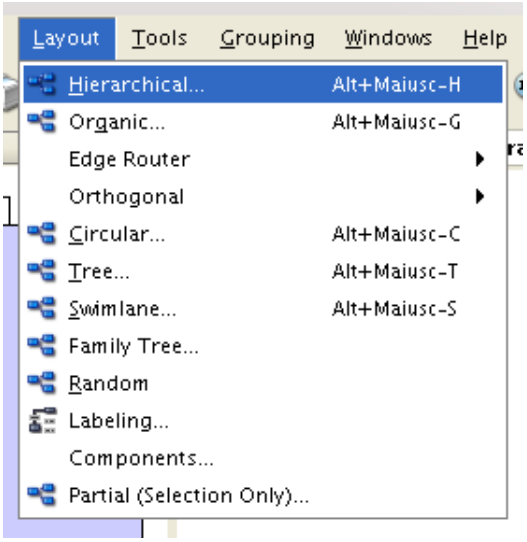
The first step involves converting a BCML file to GraphML, a task accomplished by running `bcml2graphml`:

```
bcml2graphml --srcSBGN <source BCML file>
             --targetGraphML <destination file>
```

where `source BCML file` is your BCML file and `destination file` the name of the file where the converted GraphML will be stored. `bcml2graphml` can work recursively: if you supply a path instead of a single file, every valid file in that path will be converted. This is useful if you have many BCML files.

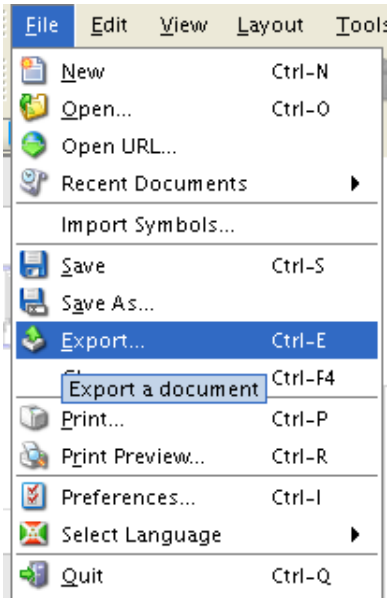
The second step involves loading the yED editor. You can download it or run it directly from the web. Upon launching it, select "Open file" from the greeting screen, and point the file browser to your GraphML file.

You will notice that all the entities in the pathway are overlapping. This is because they need to be *layouted*. From the Layout menu in yED, select "Hierarchical" (see image) and in the next window, click OK, accepting default parameters.



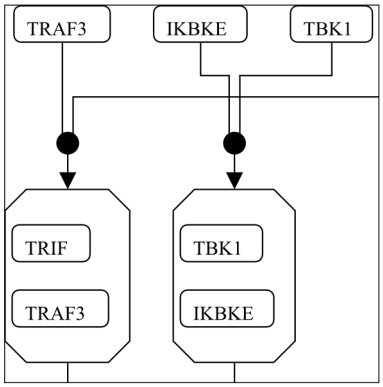

The pathway will then assume its right layout. By selecting "Save" from the "File" menu you will be able to save the layouting changes to the GraphML file.

Lastly, you will need to export the file. You can do so by choosing "Export" from the "File" menu:




It is recommended to use SVG or other vector format, which is scalable and can be converted to other formats with the appropriate software (such as the freely available Inkscape). yEd offers some other exporting options, but SVG is the one with best quality. yEd provides the capability to publish the results online, by exporting to the HTML / Adobe flash format.

An example section of a graphically converted BCML file will look like this:




## 4.3 Filtering a BCML file

BCML files can be "filtered" to include or exclude elements and reactions according to user-supplied criteria. For example, one may want to exclude all reactions that do not occur in fibroblasts. This is important because often interactions and pathway entities have been described and proven in some biological contexts but not others.

Filters can be applied on any feature of a BCML file. The most common use case is on the Findings, because the specific annotation is kept there, but in general any kind of element is usable. To actually perform filtering, you will have to use the `bcml_filter` program. It is typically invoked like this:

```
bcml_filter --srcSBGN <source BCML file> --outFile <destination BCML file>
            --filterExpr <expression>
```

As the command line says, the original file (`source BCML file`) is filtered and then written to the destination (`destination BCML file`) using a specific expression (`expression`). The filter expression must be enclosed in double quotes (").

### 4.3.1 Filter expressions

Filter expressions follow a simple logic. Criteria are specified with `element name='value'`, where `element name` can be any element that can be filtered, for example *Organism*, *CellType*, or *ProvenIn*. The criterion must be enclosed in single quotes. Multiple slections can be concatenated with `and` (all the conditions must be true) or `or` (any of the conditions must be true).

You can nest expressions with brackets, and the ones inside the brackets will be evaluated first than the ones outside.

For example, to filter a pathway for elements only described in *Mus musculus* and in macrophages, a typical filter expression would be:

```
"Organism='Mus musculus' and CellType='Macrophage'"
```

To include instead interactions proven in mouse and in human in dendritic cells, an example would be:

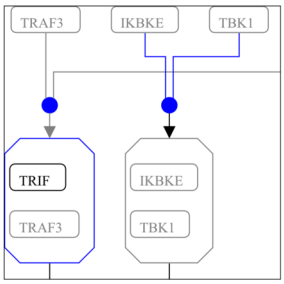
```
"(Organism='Mus musculus' or Organism='Homo sapiens') and
 CellType='Dendritic cells (DC)'"
```

### 4.3.2 Viewing filtered pathways

The file produced from the filtering can be then converted to GraphML for viewing. When viewing the graphical representation, items that do not match the filter expression are shown in light gray.

If a "disabled" element participates an interaction of any kind with another entity of the pathway, the interaction will be shown in blue: this indicates that with the current filter selection, the observed interaction may not have sense from a biological point of view.

An example is shown in the figure below.




## 4.4 Exporting an entity list from a BCML file

Entities (such as genes) in BCML files can be exported to a plain text format for use in ananalysis (see the relevant section) or for other uses. The `bcml_export` program was made for this purpose: extract a plain-text file with identifier information from a BCML file.

A typical invocation of `bcml_export` is as follows:

```
bcml_export --db <database> --method GeneList --organism <organism>
            --srcSBGN <source SBGN file> --outFile <destination file>
```

Like other tools, `source SBGN file` and `destination file` can be files or file paths: in the latter case, all files will be processed and exported. The `database` parameter indicates which database to extract entity information from, and depends on what was annotated inside the various `<Organism name=...>` tags in the BCML file. By default, it exports Entrez Gene IDs. `organism` indicates the two letter code of the organism to extract identifier information from, and as `database`, it should be present in the source file.

The result from the export is a plain-text file with the identifiers found, one per line. If the file has been filtered, only the identifiers that match the filter will be outputted: this behavior can be suppressed by adding the `--disableFilter` option.

CHAPTER

FIVE

# ADDING EXPERIMENTAL MEASUREMENTS TO A BCML FILE

Experimental measurements coming from various sources, such as fold changes, p-values, concentrations, SNPs, phosphate binding points of the corresponding proteins, can be easily added to a BCML file for viewing or analysis. The only requirement is that there must be a matching set of identifiers between the BCML file and the experimental measurements.

The mapping, or annotation, operation is divided into two main steps:

1. Preparation of an experimental measurement file
2. Incorporation of the measurements into the BCML file

## 5.1 Preparing the experimental measurement file

The experimental measurement file is a simple two-column tab-delimited text file. In the first column there is a list of identifiers, and in the second column a list of experimental measurements. Other lines (such as a header) can be present, but will be ignored during the annotation.

The identifiers in the file must be of the same type that are present in the BCML file, under the `<Organism name=...>` sub-tags (for example Entrez Gene IDs).

**Note:** If you are using Excel to export the file, pay attention that numerical values are outputted with US locale, with decimal points, and not commas as in other locales.

## 5.2 Incorporating the experimental measures

The experimental measurements are incorporated in the BCML file with the `bcml_annotate` program:

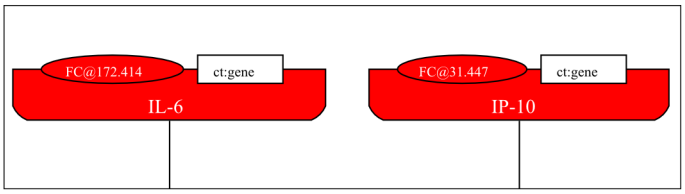
```
bcml_annotate --srcSBGN <source SBGN file> --db <database>
              --varName <measurement name>
              --varFile <experimental measurement file>
              --organism <organism> --outFile <destination BCML file>
```

`measurement name` indicates the name of the variable that will be shown in the graphs (corresponds to SBGN's StateVariable), while `database name` indicates which identifier to use for the mapping (and values should be present in the BCML file). `organism` is specified to tell `bcml_annotate` which species to annotate for in case of multiple values.

In case no match is found, no BCML file is produced and `bcml_annotate` issues a warning. There are also other less common options to specify the colors, how to handle multiple values for the same entity in the measurements, and so on. Execute `bcml_annotate` without arguments for a full list.

The resulting BCML file can then be either manipulated (filtered, etc.) or viewed after conversion to GraphML. In the latter case, entities affected by experimental measurements will appear colored (depending on the degree of the measurement) along with their actual value. The colors can be controlled from the command line or from the configuration file.

As a reference, here is an example of a BCML file where up-regulated genes were added:




SBGN's state variable holds the fold change, in this example, so not only the state of the gene can be known, but also its associated value.

## 5.3 Exporting measurements from a BCML file

BCML files with existing measurements can be exported to entity lists along with their measurements, as a two-column tab-delimited file. To do, `bcml_export` is invoked, like *when exporting entity lists* but adding the *–var* option:

```
bcml_export --db <database> --method GeneList --organism <organism>
            --srcSBGN <source SBGN file> --outFile <destination file>
            --var <variable name>
```

`variable` must be the same name as the one that was used when originally annotating the file, otherwise only an identifier list will be outputted. It is also convenient to use `---disableFilter` when exporting measurements data, so that filters will not exclude part of it.

The generated file can be opened and used with any program capable of loading tab-delimited data (including spreadsheet programs).

CHAPTER

SIX

# USING BCML PATHWAYS FOR ANALYSIS

BCML files can be converted into formats suitable for use with common pathway analysis algorithms such as the Fisher's Exact Test [Grosu2002], Gene Set Enrichment Analysis [Subramanian2005], and impact analysis [Tarca2009].

For most of the analysis algorithms, an export to gene list will be sufficient; for impact analysis there is a specialized workflow.

## 6.1 Exporting data to gene lists

Gene lists (which can then be used for Fisher's Test or GSEA) can be simply exported using `bcml_export` as *following the procedure to export identifiers*, without ising the *–var'* argument, and keeping or disabling the filter as appropriate. Once the gene list has been obtained, it can be used according to the selected analysis procedure.

## 6.2 Exporting data for impact analysis

The impact analysis format supported by BCML is the one implemented in the SPIA R package.

Exporting data for use with impact analysis requires different arguments to be passed to `bcml_export`. Specifically, it must be invoked with different `--db` and `--method` switches:

```
bcml_export --method SPIA --organism <organism> --srcSBGN <source SBGN file>
            --outFile <destination file> --var SPIA --disableFilter
```

Notice also the use of `--disableFilter`, as SPIA requires the full pathway to work effectively.

The export will produce an R file. This needs to be sourced in R before starting the analysis, as follows:

```
>>> source("mypathwayfile.r")
```

> **Warning:** SPIA needs to be installed, or the operation will fail. Also, the operation will write a file to SPIA's installation directory: make sure your user account has sufficient rights to write there.

**Note:** Because the standard spia has a fixed list of reaction types which cannot be harmonized with the dynamical nature of BCML, you will need the BCML-bundled version of SPIA (spia.r) in order to run analysis on the BCML-exported pathways. In order to use the standard spia version, you will have to manually replace the reaction types with the one defined by the orginal SPIA.

CHAPTER

SEVEN

# REFERENCES

CHAPTER

EIGHT

# INDICES AND TABLES

- *Index*
- *Search Page*

# BIBLIOGRAPHY

[Tarca2009] Tarca AL, Draghici S, Khatri P, Hassan SS, Mittal P, Kim JS, Kim CJ, Kusanovic JP, Romero R (2009) A novel signaling pathway impact analysis. Bioinformatics 25: 75-82.

[Grosu2002] Grosu P, Townsend JP, Hartl DL, Cavalieri D (2002) Pathway Processor: a tool for integrating whole-genome expression results into metabolic networks. Genome Res 12: 1121-1126.

[Subramanian2005] Subramanian A, Tamayo P, Mootha VK, Mukherjee S, Ebert BL, Gillette MA, Paulovich A, Pomeroy SL, Golub TR, Lander ES, Mesirov JP (2005) Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. Proc Natl Acad Sci U S A 102: 15545-15550.

[LeNovere2009] Le Novere N, Hucka M, Mi H, Moodie S, Schreiber F, Sorokin A, Demir E, Wegner K, Aladjem MI, Wimalaratne SM, Bergman FT, Gauges R, Ghazal P, Kawaji H, Li L, Matsuoka Y, Villeger A, Boyd SE, Calzone L, Courtot M, Dogrusoz U, Freeman TC, Funahashi A, Ghosh S, Jouraku A, Kim S, Kolpakov F, Luna A, Sahle S, Schmidt E, Watterson S, Wu G, Goryanin I, Kell DB, Sander C, Sauro H, Snoep JL, Kohn K, Kitano H (2009) The Systems Biology Graphical Notation. Nat Biotechnol 27: 735-741.