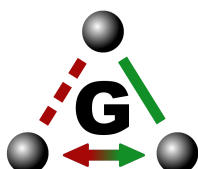


GGL Tutorial: Molecular Groups



Christoph Flamm^{1,*} and Martin Mann²

¹ Institute for Theoretical Chemistry, Vienna University



² Bioinformatics Group, University of Freiburg



<http://www.tbi.univie.ac.at/software/GGL/>

Version April 8, 2013

Built for GGL version 4.0.1

*Send comments to xtof@tbi.univie.ac.at or mmann@informatik.uni-freiburg.de

Contents

1	Molecular groups	1
1.1	Introduction to the specification language GML	1
1.2	BNF of GML	1
1.3	Keys for rule specification	2
1.4	Visualization	3
2	Molecular groups within SMILES	3
3	Molecular groups within graph rewrite rules	6

1 Molecular groups

This chapter explains how to define molecular groups in term of the Graph Modeling Language (GML) and how to use them. Therein, molecular groups are defined by an undirected graph where each node represents a single atom and edges correspond to bonds of a given valence. Within the GGL, we assume node and edge labels to be conform with the SMILES notation.

1.1 Introduction to the specification language GML

Within the GGL, graphs and graph rewrite rules are specified in a language called GML (Graph Modeling Language). Essentially, GML is composed of hierarchical **key**–**value** pairs. Keys are usually strings (some identifiers) and values specify the value of the corresponding key. Values are either single values (numbers, strings, etc) or lists of key–value pairs. Lists **must always** be enclosed in opening '[' and closing ']' brackets in GML. Nesting of lists to arbitrary depth is allowed in GML. The general structure of a GML specification looks as follows:

```
key1 [  
  key2 value2  
  key3 [  
    key4 value4  
    key5 value5  
  ]  
  key6 value6  
]
```

In the above code snippet keys 1 and 3 have both a list as value (hence the brackets). Keys 2, 4–6 are key–value pairs where the corresponding values are single values such as numbers or strings.

1.2 BNF of GML

Following is the grammar specification of GML in *Boyes Normal Form (BNF)*.

```

gml      ::= keyvalues
keyvalues ::= keyvalue (keyvalue)*
keyvalue ::= key value
key       ::= ['a'-'z','A'-'Z']['a'-'z','A'-'Z','0'-'9']
value     ::= real | integer | string | list | operator
real      ::= sign? digit '.' digit+ mantissa?
integer   ::= sign? digit+
operator  ::= '<' | '=' | '>' | '!'
string    ::= '"' instring '"'
list      ::= '[' keyvalues ']'
sign      ::= '+' | '-'
digit     ::= ['0'-'9']
mantissa  ::= ('E' | 'e') sign? digit+
instring  ::= ASCII-{'&', '"'} | '&' ['a'-'z','A'-'Z'] ';'

```

The GML-parser in GGL can parse any well formed GML file that conforms to the above BNF grammar specification. However the parser interprets only a subset of “known” key-value pairs (see according section) all other well-formed key-value pairs are **silently ignored** (Note: a source of errors is misspelling of known keys since the parsing is case-sensitive).

1.3 Keys for rule specification

The following table lists the relevant keys for molecular group specification in alphabetic order. For lists the optional enclosed keys are given in brackets.

key	type	keys in list	comment
compIDs	list	id	the node identifier that define the component
description	string	–	the textual ID of the group
edge	list	source, target, label	define a <i>bond</i> .
id	int	–	defines a numerical identifier for a <i>vertex</i> .
label	string	–	defines a textual label for a <i>vertex</i> or an <i>edge</i> .
node	list	id, label	define an <i>atom vertex</i>

key	type	keys in list	comment
molcomp	list	description, node, compIDs, (energy), (priority), (constrainXXX)	defines a molecule component
source	int	–	define the <i>source-vertex</i> of an edge
target	int	–	define the <i>target-vertex</i> of an edge

1.4 Visualization

The GML definition of molecular components can become quite large and hard to read. To ease their creation and to allow for a simple evaluation, the GGL sports the visualization script `molcomp2svg.pl` within its Perl module.

Given a molecular component in GML notation, the script produces a graphical depiction in Scalable Vector Graphics (SVG) format. Therein, compID nodes are highlighted in red. An example is given in Fig. 2c).

The `molcomp2svg.pl` script uses the `OpenBabel` package to create the 2D depictions of the molecules and thus requires its presence.

2 Molecular groups within SMILES

One field of application for the definition and use of molecular groups is the specification of molecules that differ only in a few atoms or bonds. In such cases, it can be convenient to specify only the dissimilar parts of the molecules and to use group placeholders for the equal parts. That way, the similarity becomes easy to see and the SMILES easier to read.

As an example, we use the molecules NADH and NAD^+ depicted in Fig. 1 sporting 66 and 65 atoms, respectively. The difference basically comprises only an additional proton within NADH and a charge change within the lower ring while the rest of the molecules are identically. Note, these two changes alter the ring from non-aromatic (NADH) to aromatic (NAD^+).

Minimal SMILES encodings of the molecules (highlighting the differing ring in red) are

```
NC(=O)C1[CH2]C=CN(C=1)C2OC(COP(O)(=O)O)O..
..P(O)(=O)OCC3OC(C(O)C3O)n4cnc5c(N)ncnc54)C(O)C2O
```

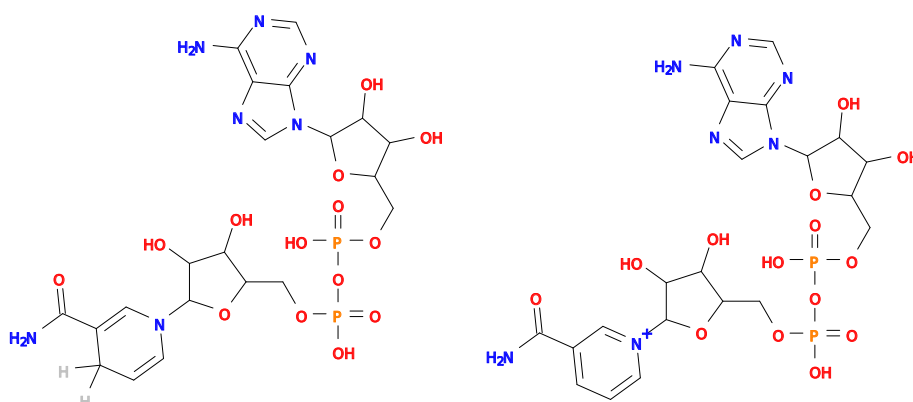


Figure 1: The molecules NADH (left) and NAD⁺ (right).

for NADH and

```
NC(=O) c1ccc[n+](c1)C2OC(COP(=O)(O)O)O . .
. . P(=O)(O)OCC3OC(C(O)C3O)n4cnc5c(N)ncnc54)C(O)C2O
```

for NAD⁺.

In contrast, when using group declarations for the identical parts, namely the CONH2 group and the ribo-adenosine, the SMILES shrinks to

```
[{CONH2}] c1[CH2]C=CN(C=1)[{Ribo-ADP}]
```

for NADH and

```
[{CONH2}] c1ccc[n+](c1)[{Ribo-ADP}]
```

for NAD⁺, both depicted in Fig. 2a) and b).

In the following, we give the GML encoding of the {CONH2} group. An according depiction, using the `molcomp2svg.pl` visualization script, is given in Fig. 2c).

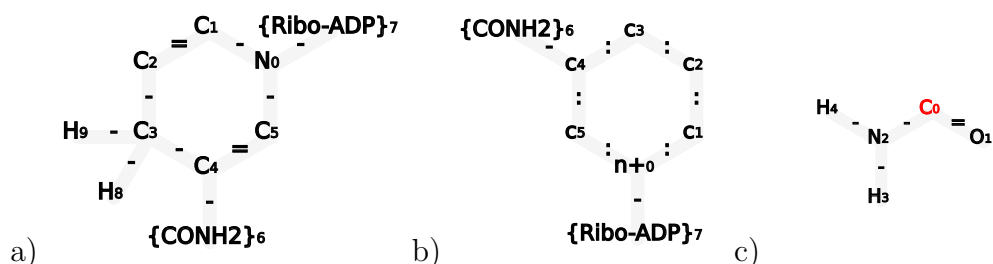


Figure 2: The molecules NADH (a) and NAD⁺ (b) described using group identifiers. Note, only for C_3 within NADH explicit proton information is given; all other carbons have one not depicted adjacent proton. (c) Depiction of the {CONH2} group using the script `molcomp2svg.pl`. Note, the proxy node is highlighted in red.

```
molcomp [
  description "{CONH2}"
  compIDs [ id 0 ]
  node [ id 0 label "C" ]
  node [ id 1 label "O" ]
  node [ id 2 label "N" ]
  node [ id 3 label "H" ]
  node [ id 4 label "H" ]
  edge [ source 0 target 1 label "=" ]
  edge [ source 0 target 2 label "-" ]
  edge [ source 2 target 3 label "-" ]
  edge [ source 2 target 4 label "-" ]
]
```

Note, both the {CONH2} as well as the {Ribo-ADP} group are molecule components that are interfacing via a single node with the remaining molecule. This enables the representation within SMILES as a complex pseudo atom in bracket notation namely [{CONH2}] or [{Ribo-ADP}], resp., as done within the shorted SMILES strings presented above. This node is specified using the `compIDs` entry within the GML definition. Therefore, a molecular group to be used within molecule representations is only allowed to sport *exactly one compID entry* naming the proxy node of the component. During the SMILES parsing, the complex pseudo atom is replaced with the proxy node of the according group and the remaining group atoms and bonds are added to the molecule.

3 Molecular groups within graph rewrite rules

The specification of (bio)chemical reactions often requires the representation of large (unchanged) parts of molecules in order to make the rule as specific as the chemical reaction. A classic example is the involvement of helper molecules like ATP, NADH, etc. that are only slightly changed but have to be represented completely to avoid the application of the rule using similar molecules.

To this end, the GGL supports the specification of molecular groups as pseudo-atoms within chemical rule definitions. They allow for a much easier and compact rule definition and avoid potential typos and mistakes.

As an example consider the lactat-dehydrogenase from the citrat-cycle given by $\text{NAD}^+ + \text{lactate} \rightarrow \text{NADH} + \text{pyruvate}$. NADH is a large molecule comprising 66 atoms. Thus, a complete specification would require the definition of all NADH atoms and bonds together with the according parts of lactate and pyruvate incorporating 76 atoms in total. Furthermore, this would be the case for all other NADH-dependent reactions as well.

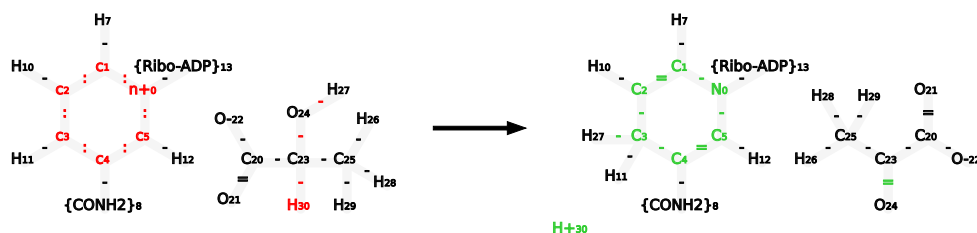


Figure 3: Lactat-dehydrogenase : $\text{NAD}^+ + \text{lactate} \rightarrow \text{NADH} + \text{pyruvate}$. The picture exemplifies the use of group identifiers to compact the rule specification. The colors indicate if specified as context (black), left (red), or right (green). Note, such a representation reduces the rule specification from 76 to only 23 atoms.

Using group identifiers, the definition of the lactat-dehydrogenase becomes much more compact as exemplified in Fig. 3. With only 23 atoms, the whole reaction is described. Note, the rule specification uses two group descriptors. Each is replaced during the rule GML parsing with according molecule components/subgraphs, i.e. $\{\text{CONH2}\}_8$ is replaced with a CONH_2 group (see Fig. 2) and $\{\text{Ribo-ADP}\}_{13}$ with a ribose and attached adenosine.

Note: group identifier are only allowed within context or for label changes within both left/right at once. The reason is: each group shows as interface

exactly one proxy node that will replace the specified pseudo atom labeled with the group ID. Thus, a rule can only change bonds with the proxy node! The rest of the group is statically added to the *rule context*. It is possible to specify label changes of the proxy node atom but these are restricted to charge changes as exemplified below. An explicit change of the proxy node label (e.g. make it aromatic “C” \rightarrow “c”) is not possible!

```
rule [
# example of charge change of a groups proxy node
# the rest of the group is added to the rule's context
left [
  node [ id 0 label "{GROUP}+" ]
  ..
]
..
right [
  node [ id 0 label "{GROUP}" ]
  ..
]
]
```