Biochemfusion

Protein Line Notation

Specification

version 1.3

2013-05-08

Table of Contents

Changes since version 1.2

Inline definitions of named modified terminal and residue structures are now supported. This allows PLN to become a self-contained structural representation of a molecule.	See the section "The inline-mod properties".

Introduction

The Biochemfusion Protein Line Notation format is a compact text representation of a protein that includes chemically significant annotations.

Although the name implies a single line of text the text may be broken into an arbitrary number of lines to ease e-mail transmission and enhance readability.

Line notation regions

The line notation consists of three distinct regions: the Sequence region, the Properties region, and the optional end-of-entry region:

```
H-ASDF-OH.H-CGTY-OH name="Simple protein" id=P00001 **
<--- Sequence ----><--- Properties ---------------><->
                                                   |_End-of-entry
```

The sequence region

The Sequence region cannot contain white space. Once a white space is encountered the Properties region is assumed to start. The region may contain linefeeds but all linefeeds must be ignored.

The Sequence region contains chains delimited by periods, '.'. A chain consists of an N-terminal specification, followed by a hyphen, '-', followed by a list of residues, followed by a hyphen, followed by a C-terminal specification.

Terminals

Standard unmodified terminals must be written as "H" (N-terminal) or "OH" (C-terminal).

Modified terminals must be written as the terminal structure name (modification name) in square brackets, e.g. "[biotin]". Terminal modification names must follow the format and constraints detailed in the section "Modification names".

The mechanism for mapping modification names to terminal structures will be determined by the actual implementation.

A terminal may also be the endpoint of a cyclization, in which case the terminal must be written as "(<cycle identifier>)". The syntax and representation of cyclizations is described in the section "Cyclizations".

Residues

The unmodified residues in a chain are single-letter uppercase or three-letter residue codes. [1] A small peptide that in IUPAC notation would be [2]

```
Ala-Cys-Asp-Glu-Phe-Gly
```

can be written as either

```
H-ACDEFG-OH
```

or

```
H-Ala-Cys-Asp-Glu-Phe-Gly-OH
```

in Protein Line Notation. Strictly speaking there are several possible forms since you may also mix single-letter and three-letter residue codes, see below.

Three-letter residue codes

Three-letter residue codes consist of a single uppercase letter followed by two lowercase letters. Three-letter codes may be mixed with single-letter codes.

The following sequences thus all represent the same molecule:

```
H-AYS-OH
H-Ala-Tyr-Ser-OH
H-AY-Ser-OH
H-Ala-YS-OH
```

Each three-letter residue code must be followed by a hyphen. The following sequence is therefore invalid (missing hyphen between "Ser" and "Glu"):

```
H-Ala-SerGlu-OH
```

Non-standard residues

Post-translationally or chemically modified residues are indicated by replacing the modifed residue's single-letter or three-letter code by a residue structure name (modification name) in square brackets.

An example would be that a carboxyl variant of a Glu residue is described by the modification name "4-carboxyglutamate". A chain containing a Glu/E residue

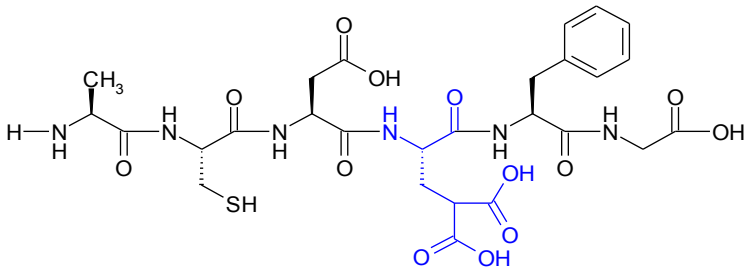
```
H-ACDEFG-OH
```

would then have a carboxyl variant by

```
H-ACD[4-carboxyglutamate]FG-OH
```

The modified sequence above corresponds to the following chemical structure

where the modified residue has been highlighted in blue.



How modification names are mapped to actual residue structures will be implementation-dependent. One way of mapping names to actual structures is via the use of *inline modification* properties. See "The inline-mod properties" section.

D-form residues

All residues can be transformed into their D-forms by prefixing the residue one-letter code or modification name with "{d}", e.g.

```
H-AS{d}EF-OH
```

has a D-Glu in position 3.

Three-letter residue D-forms must be prefixed with a lowercase 'd', so the following sequences are identical:

```
H-Ala-dTyr-Ser-OH
H-A-dTyr-S-OH
H-A{d}YS-OH
```

Usage of hyphen as an optional delimiter

Hyphens are generally allowed to enclose any (set of) residues. Modification names may also be optionally enclosed in hyphens.

The following sequences are considered valid and identical:

```
H-Ala-dTyr-[Gla]-Ser-OH
H-A{d}YR[Gla]S-OH
H-A{d}YR-[Gla]-S-OH
H-A{d}YR[Gla]-S-OH
H-A{d}YR[Gla]S-OH
```

The following sequences are legal and identical too:

```
H-AD[Gla]SOH-OH
H-AD[Gla]S-OH-OH
H-AD-[Gla]SOH-OH
H-AD-[Gla]S-O-H-OH
```


Hyphens may not be inserted between a D-form modifier and a residue. The following three sequences are invalid:

```
H-AS{d}-YR-OH
H-AS{d}-[Gla]R-OH
H-ASd-Tyr-R-OH
```

Bridges and cycles

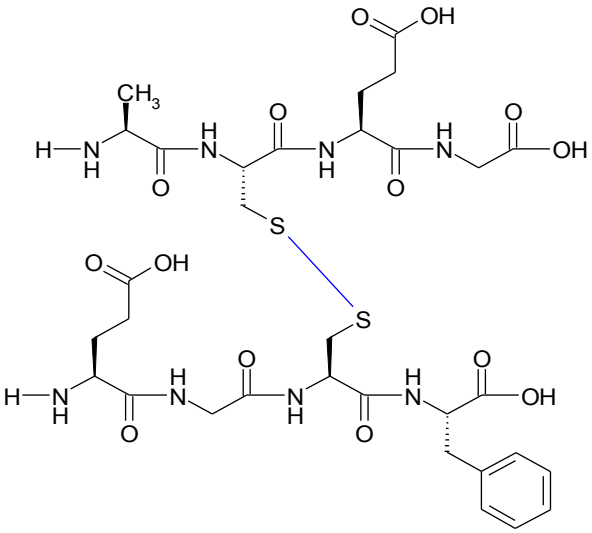
Bridges and cycles may form between residue pairs, terminal pairs, and residue-terminal pairs. Each bridge or cycle endpoint is indicated by a parenthesis pair containing a unique identifier.

Disulfide bridges

Disulfide bridges are indicated by pure-numeric identifiers that may suffix cysteine or selenocysteine residues. An example is

```
H-AC(1)EG-OH.H-EGC(1)F-OH
```

which corresponds to the following chemical structure where the disulfide bond has been highlighted in blue.



The identifiers chosen do not have to be consecutive or follow any numeric sequence; they only serve to uniquely identify the bridge. This means that the following example is equivalent to the one above.

```
H-AC(5)EG-OH.H-EGC(5)F-OH
```


Cyclizations

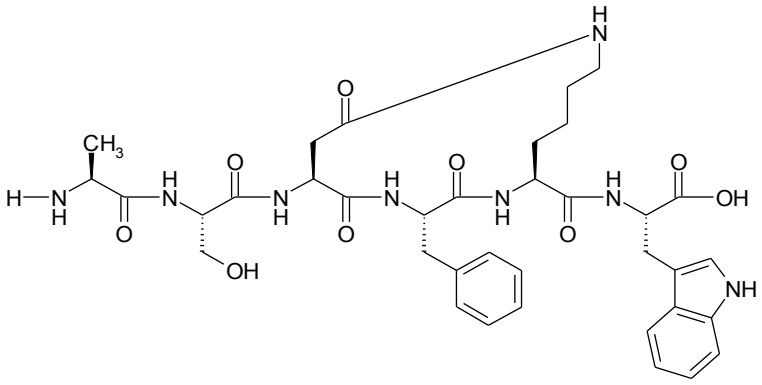
A cyclization differs syntactically from a disulfide bridge by the form of its identifier. A cyclization identifier is formed by the keyword "cyclo" followed by a unique integer.

A cyclization may only form between a reactive amino group and a reactive acid group. How the actual reaction sites are found will be implementation-dependent.

Cyclizations may form between residue sidechains, e.g.

```
H-ASD(cyclo1)FK(cyclo1)W-OH
```

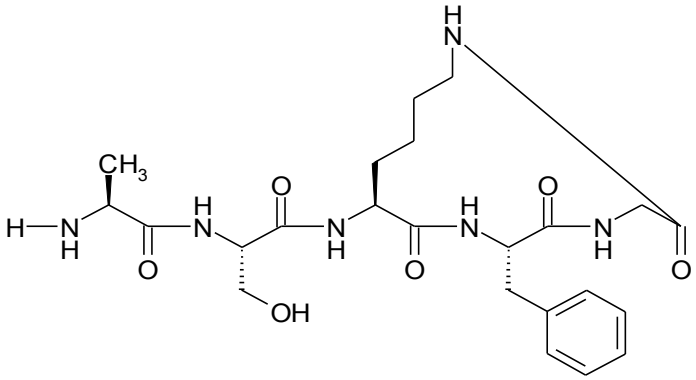
which corresponds to the chemical structure:



Cyclizations may also form between terminals and side chains, e.g.

```
H-ASK(cyclo1)FG-(cyclo1)
```

corresponding to



or between terminals as in the fully cyclic peptide below

```
(cyclo1)-ASDEF-(cyclo1)
```

The numbering obeys the same rules as the numbering of disulfide bridges.

The numbering does not have to be consecutive, it only serves to uniquely identify a particular cycle.

Unnumbered cyclo tags

When using "cyclo" to indicate a fully cyclic chain you may use it without numeric identifiers since it is clear where the endpoints of the cyclization are: Connecting terminals from different chains by using a "cyclo" tag is equivalent to forming a normal peptide bond and is not allowed.

The cyclic peptide above can thus also be written as

```
(cyclo)-ASDEF-(cyclo)
```

Note that the unnumbered form of "cyclo" is only valid for terminal-terminal cyclizations. The following two examples are therefore invalid:

```
H-ASD(cyclo)EK(cyclo)L-OH
(cyclo)-ASD(cyclo)E-OH
```

while their numbered counterparts are valid:

```
H-ASD(cyclo1)EK(cyclo1)L-OH
(cyclo1)-ASD(cyclo1)E-OH
```

Inter-chain cyclizations

The "lactam" tag can be used to differentiate cross-chain links from intra-chain cyclizations. Since both "lactam" and "cyclo" specify the same reaction mechanism they should be allowed interchangeably in PLN input. The following two examples should be allowed and describe the same molecule:

```
H-ASD(cyclo1)EK(cyclo1)L-OH
H-ASD(lactam1)EK(lactam1)L-OH
```

It is recommended that implementations output PLN that uses "lactam" for inter-chain links only.

The cyclization tags are uniquely identified by the tag name and the following number. This means that "cyclo1" and "lactam1" specify two different lactam cycle endpoints. You must therefore match "lactam" tags with "lactam" tags and "cyclo" tags with "cyclo" tags.

The following PLN is therefore invalid:

```
H-ASD(cyclo1)EK(lactam1)S-OH
```

and an appropriate error message would be that there is no matching endpoint for "cyclo1".

Modification names

Allowed characters in modification names are

- All alpha-numeric characters `a..z, A..Z, 0..9`
- Square brackets and normal parentheses `[] ()`
- Comma, period, hyphen, plus sign `, . - +`
- Apostrophe and underscore `' _`

If square brackets and parentheses are used within a modification name the brackets and parentheses must be correctly paired.

Although periods are allowed within a name, modification names cannot end with a period.

The Properties region

The Properties region lists properties that are key-value pairs separated by equal signs, '='. Keys, values, and '=' characters may be delimited by any number of white spaces and linefeeds. Linefeeds should be ignored as in the Sequence region.

Property keys

A property key is an all-lowercase identifier. Valid property keys are

Key	Key description
name	Protein name
id	Protein id
inline-mod	Inline named terminal or residue modification structure.

The name property

A *name* value may only contain white space or double quote characters if it is enclosed in double quotes. A double quote within a *name* value must be encoded as two consecutive double quotes, e.g.

Name	Correctly quoted
Simple protein	"Simple protein"
Not so "small" protein	"Not so ""small"" protein"
"Nice"-protein	"""Nice""-protein"

The *name* property is optional. It may only be defined once in a PLN entry.

The id property

The value of the *id* property may contain alphanumeric and underscore characters only - no white space is allowed.

The *id* property is optional and may only be defined once in a PLN entry.

The inline-mod properties

Named modification structures can be embedded in the PLN as *inline-mod* properties. Any number of *inline-mod* properties are allowed.

Each *inline-mod* property defines one named structure in the following format.

```
inline-mod=Y-residue,[newTyr],C16H23N1O2,QkNGTR...aCGg==
           <--type-> <-name-> <--info--> <--structure-->
```

The property consists of four comma-separated values. None of the values may contain spaces or commas, not even if attempted quoted.

▪ The *type* value defines the type of the modification structure. It will be either a one-letter residue code followed by "-residue" or "N-terminal" or "C-terminal".

▪ The *name* value is the modification name enclosed in square brackets, so the modification name "Gla" will be listed as "[Gla]". The modification name must adhere to the standard requirements listed in the "Modification names" section.

▪ The *info* value can be any text that describes the structure in a compact form, e.g. the sum formula or a chemical name. Implementations are also allowed to simply write a blank *info* value, but the value cannot be skipped; e.g. using a blank info value is allowed:

```
inline-mod=Y-residue,[newTyr],,QkNGTR...aCGg==
```

but skipping it is not:

```
inline-mod=Y-residue,[newTyr],QkNGTR...aCGg==
```

▪ The *structure* is the modification structure embedded as a base64-encoded Biochemfusion binary molecule file.[3,4] This is a very compact way of representing a molecule with 2D coordinates included.

The base64 character set used is this list of 64 characters:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
```

The base64-encoded molecule file must use '=' characters for padding so the base64 string is always a multiple of 4 characters.

The end-of-entry region

An optional end-of-entry delimiter may be used to separate a PLN entry from plain text that follows or other PLN entries in a multi-entry text. If no other text follows the PLN entry an end-of-entry delimiter is not required.

An end-of-entry delimiter consists of at least one white space followed by two asterisk '*' characters.

Examples of correct usage:

PLN text	Comments
`H-ACDEFG-OH ** which is a fascinating entry...`	End-of-entry used to separate PLN from plain text.
`H-ACDEFG-OH name=1st_entry**H-QWER-OH name=2nd_entry**H-EFTYS-OH name=final_entry`	Three PLN entries. Note that the last entry does not need an end-of-entry delimiter.
`H-ACDEFG-OH name="Two stars **" ** which is...`	The protein name (here "Two stars **") can contain an end-of-entry delimiter as long as it is correctly quoted.

Examples of invalid end-of-entry delimiter usage:

PLN text	Comments
`H-ACDEFG-OH**`	Missing white space before '**'.
`H-ACDEFG-OH↵ **`	Missing white space before '**' (linefeeds are ignored).
`H-ACDEFG-OH name="1st_entry **"...and I am quoting here`	Incorrectly quoted protein name masks end-of-entry delimiter.

References

(1) IUPAC "Nomenclature and Symbolism for Amino Acids and Peptides"
http://www.chem.qmul.ac.uk/iupac/AminoAcid/AA1n2.html#AA1

(2) IUPAC "Nomenclature and Symbolism for Amino Acids and Peptides"
http://www.chem.qmul.ac.uk/iupac/AminoAcid/A1819.html#AA191

(3) Base64 encoding converts binary data to printable text.
http://en.wikipedia.org/wiki/Base64

(4) The specification for the Biochemfusion binary molecule format is available at the Biochemfusion web site.
http://wwwbiochemfusion.com/doc/Biochemfusion_BinaryMoleculeFormat_1.0_spec.pdf

