PSI Recommendation
PSI Mass Spectrometry and Proteomics Informatics Working Groups
Status: DRAFT

Douwe Schulte, Utrecht University
Juan Antonio Vizcaíno, EMBL-EBI
Eric W. Deutsch, Institute for Systems Biology
Ralf Gabriels, Ghent University
Joshua A. Klein, Boston University
Petr Novák, Institute of Microbiology, CAS
Dimitry Loginov, Institute of Microbiology, CAS

12 Sep 2025

ProForma 2.1 (Proteoform and Peptidoform Notation)

Status of this document

This document provides information to the proteomics community about a new extension (version 2.1) of the standard proteoform notation called ProForma. Distribution is unlimited. This document will be ratified via the HUPO Proteomics Standards Initiative (PSI) Document Process. Any alterations of this document MUST also follow the HUPO PSI Document Process.

Version Draft 1 of version 2.1

1. Abstract

The Human Proteome Organisation (HUPO) Proteomics Standards Initiative (PSI) defines community standards for data representation in proteomics to facilitate data comparison, exchange and verification. This document presents the updated specification (version 2.1) for a proteoform and peptidoform notation, which is based on the ProForma notation [1], published originally by the Consortium for Top-Down Proteomics, and then extended to version 2.0 by the PSI [2].

Further detailed information, including any updates to this document, implementations, and examples is available at http://psidev.info/proforma.

Contents

2. Introduction

2.1 Description of the need

Protein and peptide sequences are usually represented using a string of amino acids using a well-known one letter code endorsed by the IUPAC (see e.g. https://wissen.science-and-fun.de/chemistry/biochemistry/iupac-one-letter-codes-for-bioinformatics/). Representing all the possible variations of a protein or peptide primary structure, including both artefactual and post-translational modifications (PTMs) of peptides and proteins is less clear. For example, the Consortium for Top-Down Proteomics (CTDP) introduced a standard proteoform notation format called ProForma [1, 3] for writing the primary structures of fully characterized proteoforms [4]. Proteoforms comprise protein species that include variations arising from genetic, transcriptomic, translational, post-translational, and artefactual (e.g., during sample processing) sources. ProForma specifically focuses on representing PTMs of endogenous and artefactual sources. Briefly, ProForma describes proteoforms as the amino acid sequences (the one-letter code representation) complemented with information on any modifications (of a known identity or via unidentified mass shifts) given in brackets following certain amino acids.

Despite its suitability to support a wide range of possible use cases, the original ProForma notation had some limitations. Additionally, the Proteomics Standards Initiative (PSI) has developed a format called PEFF (PSI Extended FASTA Format, http://psidev.info/peff) [5]. Although PEFF's primary intended use is for representing search databases for optimising proteomics analyses, PEFF can also be used to represent proteoforms [4] (see more details in Section 4). Therefore, there are multiple ways of encoding protein modifications and extended discussion took place to achieve a consensus. A comprehensive standard notation for proteoforms, as well as for their peptidic counterparts –peptidoforms (term introduced in [6])– is then required for the community, so that it can enhance the current description or be newly embedded in many relevant PSI (and potentially other) file formats.

The format specification version presented here, ProForma 2.1, represents an updated version of ProForma 2.0 [2] which constituted the consensus between both groups, CTDP and PSI, for the enhanced standard representation of proteoforms and peptidoforms. Compared to the original ProForma notation, it aims to support a broader variety of peptidomics and proteomics approaches, including bottom-up (focused on peptides/peptidoforms) and middle/top-down (focused on proteins/proteoforms) approaches [7]. The name of the notation, ProForma 2.0 (now updated to version 2.1), derives from the original ProForma notation introduced by CTDP. For simplicity, going forward we will refer to this extended notation as ProForma.

2.2 Requirements

The main eight requirements to be fulfilled for a proteoform and peptidoform notation are:

- It MUST be a string that is human readable, so it can be generally understood by human individuals.
- It MUST be machine parsable. Other variants of this notation will not be supported computationally, although they could be 'human readable.'
- It MUST be able to support the encoding of amino acid sequences and protein modifications.
- It MUST be able to support the main use cases needed by the proteomics community as a whole, including both bottom-up (focused on peptides/peptidoforms) and middle/top-down (focused on proteins/proteoforms) approaches.
- It MUST be flexible to accommodate different "flavours" of notations, considering common current use.
- It MUST be compatible with existing PSI file formats, where it could be used.
- It MUST be able to capture ambiguity in the position of the modified sites.
- It MUST be able to evolve, so new use cases can be added iteratively in the future.

Several of these requirements, particularly the first three, coincide with those of the original ProForma notation [1, 3]. The fourth requirement was present in the ProForma notation description, but now includes support for the bottom-up proteomics-specific entities, i.e., peptides, whereas the original ProForma notation exclusively targeted whole proteoforms. The final four requirements are new. No new requirements have been introduced in version 2.1, when compared with version 2.0.

2.3 Issues to be addressed

The main issues to be addressed by ProForma are:

- It MUST be able to represent peptidoforms and proteoforms in a consistent and reproducible way, considering the different ways of representing protein modifications.
- It MUST be able to be used jointly with the Universal Spectrum Identifier (USI) [8], to represent peptide-spectrum matches (PSMs), and to represent proteoform-spectrum matches (PrSMs).

3. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119 (2).

4. The Proteoform and Peptidoform Notation Definition

4.1 The documentation

The documentation of the ProForma Notation for proteoform and peptidoforms is divided into several components. All components in their most recent form are available at the HUPO-PSI website (http://psidev.info/proforma) and at the ProForma GitHub page (https://github.com/HUPO-PSI/ProForma/).

- Main specification document (this document).

4.2 Relationship to other specifications

The format specification described in this document is not being developed in isolation; indeed, it is designed to be complementary to, and thus used in conjunction with, several existing and emerging models. Related specifications include the following:

1. *PSI Universal Spectrum Identifier* (http://psidev.info/USI). The PSI Universal Spectrum Identifier [8] is designed to provide a universal mechanism for referring to a specific spectrum in public repositories. It can optionally include an interpretation of the spectrum using the notation described in this specification. Displayers of USIs MAY use any of the supported ProForma notations.
2. *mzSpecLib, the PSI spectrum library format* (http://psidev.info/mzSpecLib). The PSI spectrum library format has been developed as a standard mechanism for storing spectrum libraries. Identified spectra of modified peptides can be represented using this ProForma notation. Furthermore, many spectrum library entries are derived from multiple spectra, and this provenance will be referenced using USIs.
3. *mzPAF* (http://psidev.info/mzPAF). The PSI peak annotation format (used within mzSpecLib). Can contain ProForma definitions of otherwise impossible to write fragments.
4. *PROXI* (http://psidev.info/proxi). The Proteomics Expression Interface being developed by the PSI is a standardized API by which mass spectrometry proteomics information can be exchanged. References to individual spectra will be made via USIs.
5. *PEFF* (http://psidev.info/peff). Although it is not its main intended use, the PSI Extended Fasta Format enables the representation of proteoforms [5]. However, PEFF was not designed for the representation of the (potentially much shorter) peptidoforms. Additionally, PEFF 1.0 supports formally only a subset of the use cases outlined in this specification. Another key difference is that each proteoform instance in PEFF requires a FASTA header, whereas this is not required in ProForma.
6. *ProForma* (http://psidev.info/proforma). ProForma Proteoform Notation version 1, which enables the representation of proteoforms (https://topdownproteomics.github.io/ProteoformNomenclatureStandard/), developed by the CTDP [1, 3]. This specification is subsumed by this new version 2.0 [2] (now updated to version 2.1) ProForma specification.

4.3 Changes from version 2.0 to 2.1

The core of this specification document remains mostly unchanged from version 2.0. But the specification has been restructured to fit the different levels of compliance and additionally, version 2.1 now adds support for the following use cases:

- Named entities (section 8.2)
- Custom monosaccharides in glycan compositions (section 10.2)
- Added charged formulas (section 11.1)
- Controlling placement of ambiguous modifications (section 11.2)
- Support for N and C-terminal modifications as global modifications (section 11.3)
- Moved chimeric spectra into the main specification (section 11.4)
- Moved charge states for peptidoforms into the main specification (section 11.5)
- Defining MSn precursor ions (Section 11.6)

Additionally, clarifications have been added:

- Section 5: Character encoding is UTF-8.
- Section 5.1: Formal grammar (and appendix Ⅰ)
- Section 5.2: Data schema (and appendix Ⅱ)
- Section 5.3: Features overview
- Section 6.2.1.3: Best practices on isomeric state of modifications
- Section 6.2.1.4: Best practices on the placement of modifications
- Section 9.3: An error in the representation of the branched peptides in examples a and b has been corrected.
- Section 12 *(Pending issues)* now includes two additional sections ('Representation of ambiguity of locations across linear peptides' and 'Representation of DNA and RNA').
- Some URLs have been updated.

Differently to version 2.0, this specification document has been structured to describe the different features included in the different levels of compliance. This is intended to help implementers understand the features needed for the different compliance levels better.

Starting in this version 2.1, ProForma also supports describing features in additional extension documents external to the main specification. These extensions would otherwise be treated like separate feature extensions that must be separately verified with the producer or consumer of ProForma sequences. These extensions are expected to be compatible with ProForma 2.1 and subsequent versions of the main specification.

5. Introduction

The ProForma notation is a string of characters that represent linearly one or more peptidoform/proteoform primary structures with possibilities to link peptidic chains together. It is not meant to represent higher order structures.

ProForma is case insensitive. However, within the data that follows the different keys, capitalisation may be important. In that case, capitalisation sensitivity is the decision of the supported CVs/ontologies.

Since ProForma MAY be used to represent both peptidoforms and proteoforms, there is currently no limit in its maximum length. Line breaks MUST NOT be used. However, non-ASCII characters are also allowed since non-ASCII characters can be included in the supported ontologies and controlled vocabularies (CVs). The character encoding MUST be UTF-8.

If implementers want to add any general metadata (e.g. date of creation, software, version of ontologies, etc) to ProForma entities, the way to do it in this version would be to use the INFO tag (section 6.6) or the name (section 8.2).

Due to the extensive list of use cases supported in this specification, it is not expected that all implementers can provide support to all the supported features from ProForma. To facilitate adoption and separate some of the use cases, there are multiple "levels of compliance" and extensions for ProForma. Differently to version 2.0, this specification document has been structured to describe the different features included in the different levels of compliance.

The base level 'Base-ProForma' (section 6) is designed to be simple enough to be easy to implement, any software that writes ProForma sequences is incentivised to limit the output to features supported in this level if possible. The full level 'level 2-ProForma v2.1' (section 7) adds support for ambiguity at the amino acid and modification level as well as adds some more complex use cases. The last four levels are each geared towards a particular field: top-down (section 8), cross-linking (section 9), glycans (section 10), and spectral (section 11). If one implementation supports more than one extension, multiple supported extensions can be indicated separated by '+'. Example: (level 2-ProForma + cross-linking + mass spectrum compliant).

Additionally, see Section 12 "Pending Issues - Future developments" for features not yet formally supported in this version of the specification. In particular the representation of non-linear peptides is NOT formalised in this version of ProForma. See section 12.1 *(Representation of cyclic peptides)*, for possible ways to represent them. In the future, there could be additional extensions, e.g., for lipid molecules and oligonucleotide fragments.

5.1 Formal grammar

The full notation can be used to write down a ‘compound peptidoform ion’ which can have global modifications based on the sample conditions. This compound peptidoform ion ties one or more peptidoforms together that might be present in the same spectrum. These peptidoforms are separated with a ‘+’ sign. The ‘*’ symbol indicates 0 or more repetitions of the element in brackets.

Compound peptidoform ion =

(>>>NAME)<GLOBAL_MOD>PEPTIDOFORM ION(+PEPTIDOFORM ION)*

A peptidoform ion is composed of one or more peptidoforms that, at some point in their lifespan, form a single molecule and together have a charge.

Peptidoform ion = (>>NAME)PEPTIDOFORM(//PEPTIDOFORM)*/CHARGE

A peptidoform is a linear sequence of amino acids with modifications. A peptidoform has a single mass, unless the ambiguous amino acids B or Z are used.

Peptidoform = (>NAME)[UNKNOWN_POS]?{LABILE}[N_TERM]-SEQUENCE-[C_TERM]

This overview also clarifies in what scope modification names are defined.

- Any global modification is placed on all peptidoforms and as such on all linear peptides in the whole compound peptidoform.
- Any cross-linking or branch named modification, for example a cross-link called ‘#XL1’ exists in its entire peptidoform. This means that multiple peptidoforms in the same compound peptidoform can reuse the same name for a cross-link while each refers to a different cross-link. See section 9.2.
- An ambiguous modification, a modification where its location is unknown but it is known to be present at that linear peptide, is scoped to its linear peptide. See section 7.6.
- A labile modification indicates a modification is present at a particular linear peptide, but it is lost in the mass spectrometer itself so it is not tied to a particular location. See section 6.4.

A full formal grammar of the full ProForma specification with all extensions is present at: https://github.com/HUPO-PSI/ProForma/blob/master/grammar/proforma.ebnf, as well as added as appendix I.

5.2 Data schema

A JSON data schema is defined. This schema defines which properties are needed for all data structures to fully store a ProForma sequence. This schema is intended to help implementers in designing the data types used in their parsers.

The schema is available as a JSON scheme at: https://github.com/HUPO-PSI/ProForma/blob/master/proforma.schema.json.
Additionally, it is available as a markdown document for easier human readability at: https://github.com/HUPO-PSI/ProForma/blob/master/proforma.schema.md. The markdown document is also added as appendix II.

5.3 Overview of features

An overview of all features available in the different compliance levels of ProForma are displayed below. Each shows an example and the defining section.

5.3.1 Base-ProForma

Feature	Example	Section
Amino acids (+UO)	AHAFCKUTO	6.1
Unimod names	PEM[Oxidation]AT	6.2.1
PSI-MOD names	PEM[monohydroxylated residue]AT	6.2.1
Unimod numbers	PEM[UNIMOD:35]AT	6.2.2
PSI-MOD numbers	PEM[MOD:425]AT	6.2.2
Delta masses	PEM[+15.995]AT	6.2.3
N-terminal modifications	[Carbamyl]-QPEPTIDE	6.3
C-terminal modifications	PEPTIDEG-[Methyl]	6.3
Labile modifications	{Glycan:Hex}EM[U:Oxidation]EV	6.4
Multiple modifications	MPGNW[Oxidation][Carboxymethyl]PESQE PEPTIDEG-[Methyl][Amidated] [Acetyl][Carbamyl]-QPEPTIDE	6.5
Information tag	ELV[INFO:AnyString]IS	6.6

5.3.2 Level 2-ProForma

Feature	Example	Section
Ambiguous amino acids	BZJX	7.1
Prefixed delta masses	PEM[U:+15.995]AT	7.2
Mass gap	PEX[+147.035]AT	7.3
Formulas	PEM[Formula:O]AT PEM[Formula:[17O1]]AT	7.4
Mass with interpretation	PEM[+15.995|Oxidation]AT	7.5
Unknown mod position	[Oxidation]?PEMAT	7.6.1
Set of positions	PEP[Oxidation#1]M[#1]AT	7.6.2

Range of positions	PE(PM)[Oxidation]AT	7.6.3
Position scores	PEP[Oxidation#1(0.95)]M[#1(0.05)]AT	7.6.4
Range position scores	(PEP)[Oxidation#1(0.95)]M[#1(0.05)]AT	7.6.5
Amino acid ambiguity	(?VCH)AT	7.7
Modification prefixes	PEPM[U:Oxidation]AS[M:O-phospho-L-serine]	7.8

5.3.3 Level 2-ProForma + top-down

Feature	Example	Section
RESID modifications	EM[R:L-methionine sulfone]EM[RESID:AA0581]	8.1
Names	(>Heavy chain)EVQLVESG	8.2

5.3.4 Level 2-ProForma + cross-linking

Feature	Example	Section
XL-MOD modifications	EVTK[X:Aryl azide]LEK[XLMOD:00114]SEFD	9.1
Cross-linkers (intrachain)	EVTK[X:Aryl azide#XL1]LEK[#XL1]SEFD	9.2.1
Cross-linkers (interchain)	EVTK[X:Aryl azide#XL1]L//EK[#XL1]SEFD	9.2.2
Branches	ED[MOD:00093#BRANCH]//D[#BRANCH]ATR	9.3

5.3.5 Level 2-ProForma + glycans

Feature	Example	Section
GNO modifications	NEEYN[GNO:G59626AS]K	10.1
Glycan compositions	NEEYN[Glycan:Hex5HexNAc4NeuAc1]K	10.2

5.3.6 Level 2-ProForma + advanced complexity

Feature	Example	Section
Charged formulas	SEQUEN[Formula:Zn1:z+2]CE	11.1
Controlling placement	PTI(MERMERME)[+32|Position:E]PTIDE [Oxidation|Limit:2]^4?PEPTIDE [Oxidation|CoMKP]?PEPT[Phospho]IDE PIE(MEME)[Dioxidation|CoMUP][Oxidation|CoMUP]PTE	11.2
Global isotope	<13C>CARBON	11.3.1
Fixed modifications	<[Oxidation]@M>ATPEMILTCMGCLK <[TMT6plex]@K,N-term>ATPEILTCNSIGCLK	11.3.2
Chimeric spectra	NEEYN+SEQUEN	11.4
Charges	SEQUEN/2 SEQUEN/[Na:z+1,H:z+1]	11.5
Ion notation	SEQUEN-[b-type-ion]	11.6

6. Level of Compliance: Base Level Support

Technical name: Base-ProForma compliant

6.1 The canonical amino acid sequence

Amino acid sequences are represented by strings of amino acids represented as characters using the one letter code endorsed by the IUPAC (http://publications.iupac.org/pac/1984/pdf/5605x0595.pdf and https://wissen.science-and-fun.de/chemistry/biochemistry/iupac-one-letter-codes-for-bioinformatics/). There are also letters for representing unusual amino acids (see http://www.insdc.org/documents/feature_table.html#7.4.3), which are used in some UniProt entries. These are:

- U: Selenocysteine
- O: Pyrrolysine

Additionally level 2-ProForma compliant adds support for ambiguous amino acids see 7.1.

6.2 Generic representation of protein modifications

Multiple formats and reference systems MUST be supported, because some flexibility is required. The same approach is followed for both artifactual protein modifications and natural PTMs. Square brackets MUST be used to represent them when the position is unambiguous. They are located after the character representing the modified amino acid. If there is ambiguity in the position of the protein modification, different rules apply (see section 7.6).

Two different generic reference systems for protein modifications are supported including the following CVs and/or ontologies:

- Unimod (http://unimod.org/).
- PSI-MOD (https://github.com/HUPO-PSI/psi-mod-CV).

Additionally, top-down support adds RESID as supported ontology, see 8.1; cross-linking support adds support for XLMOD, see 9.1; and glycan support adds GNOme as supported ontology, see 10.1.

Furthermore, if there is a need for a use case specific reference system the tag ‘Custom’ SHOULD be used to define modifications in an implementation specific reference system. This could for example be used to contain modifications that are being tested before adding them to a public reference system. Alternatively, this tag could be used to allow user specified modifications when ProForma strings are used by users of a software. Any information in this tag is not expected to be cross software compatible, or even compatible across different instances of the same software, if a user controlled reference system is used.

6.2.1 Controlled vocabulary or ontology modification names

The names from different CV or ontology terms MAY be used to represent protein modifications. The two main reference systems used are Unimod and PSI-MOD. However, to facilitate differentiation between reference systems for readers, the names coming from the other four supported CVs/ontologies MUST be preceded by a letter and colon, indicating the originating CV/ontology. In the case of Unimod and PSI-MOD, the use of prefixes is optional.

Examples of proper modification name usage:
- Unimod: U (use of the one letter prefix is optional)
- PSI-MOD: M (use of the one letter prefix is optional)
- RESID: R (use of the one letter prefix is mandatory) (only for top-down support)
- XL-MOD: X (use of the one letter prefix is mandatory) (only for cross-linking support)
- GNO: G (use of the one letter prefix is mandatory) (only for glycan support)
- Custom: C (use of the one letter prefix is mandatory)

EM[Oxidation]EVEES[Phospho]PEK (example using Unimod names)
EM[L-methionine sulfoxide]EVEES[O-phospho-L-serine]PEK (example using PSI-MOD names)
EM[R:L-methionine sulfone]EVEES[O-phospho-L-serine]PEK (see section 8.1)
EMEVTK[X:DSS#XL1]SESPEK (see section 9.2)

In the case of GNO (only needed for glycan support, see section 10.1), the use of accession numbers is preferred since accession numbers and names are often the same. For the definition of accession numbers see 6.2.2. Example:

NEEYN[GNO:G59626AS]K is preferred over NEEYN[G:G59626AS]K

Prefixes can still be used for Unimod and PSI-MOD names (but it is not included in basic support, see section 7.8):

EM[U:Oxidation]EVEES[U:Phospho]PEK
EM[M:L-methionine sulfoxide]EVEES[M:O-phospho-L-serine]PEK

If prefixes are not used for CV/ontology term names, different CVs/ontologies in the same ProForma instance SHOULD NOT be mixed:

EM[Oxidation]EVEES[O-phospho-L-serine]PEK -> Different CVs/ontologies SHOULD NOT be used.

Special characters do not need to be escaped. The only restriction is that unpaired bracket characters MUST NOT be used. Example of properly paired internal brackets:

EM[Oxidation]EVE[Cation:Mg[II]]ES[Phospho]PEK

6.2.1.1 Best practices on the use of protein modifications

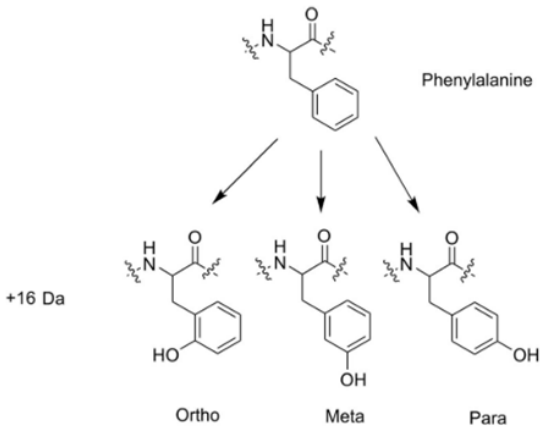
In the same sequence, the same reference system SHOULD be used to represent the protein modifications. However, the delta mass notation (section 6.2.3) MAY be combined with the other cases.

6.2.1.2 Definition of the Unimod modification name

The Unimod OBO file SHOULD be used: https://unimod.org/obo/unimod.obo. Within this file, term names are found in the "name" tag. These terms differ in the Unimod web interface (https://unimod.org/). There, the equivalent to the "name" field in the OBO file is the "PSI-MS Name" column, if not empty (if there is a value). If the "PSI-MS Name" field is empty, the "interim name" is used. Unimod synonyms are currently NOT supported, as they are provided inconsistently.

6.2.1.3 Best practices on isomeric modifications

For some protein modifications there are different isomers. For example, oxidation of phenylalanine leads to addition of oxygen in the ortho-, meta-, or para-position, generating multiple isoforms bearing the same mass shift. The correct reporting of isomeric and isobaric peptides is especially critical in structural mass spectrometry applications, such as hydroxyl radical protein footprinting, where precise localization of modifications directly impacts structural interpretation.



Adopted from [9]

Generally, if the exact isomer is not known the Unimod term or a general PSI-MOD term can be used. If the exact isomer is known the most specific PSI-MOD term MUST be used.

PEL[Oxidation]TIDE	General oxidation from Unimod
PEL[Monohydroxylated leucine]TIDE	Oxidation specific for Leucine from PSI-MOD
PEL[(2S,4R)-5-hydroxyleucine]TIDE	Specific oxidation isomer from PSI-MOD

6.2.1.4 Best practices on modification positions

If a modification defined in a reference system (CV/ontology) is defined to have specific positions on which it can occur, these SHOULD be followed in the sequence. For example in Unimod, since 'Deamidated' is allowed at the side chains of Q, N, and R anywhere, and at the side chain of F at the protein N-terminus, a notation with 'Deamidated' at A would not be strictly correct, although it would be allowed. Additionally, if the reference system describes a modification as being allowed on a certain amino acid only at a certain terminus, this SHOULD be interpreted as being allowed on the side chain of that amino acid, not the terminus itself. For example the following are valid:

E[Glu->pyro-Glu]KDTYL
HGWVRQAPG[Oxidation]
[Acetyl]-EKDTYL
HGW-[Methyl]

And the following are invalid according to Unimod:

[Glu->pyro-Glu]-EKDTYL
HG[Oxidation]WVRQAPG
E[Acetyl]KDTYL
HGW[Methyl]

In some cases it makes sense to use modifications on locations not allowed by the database. For example when seeing modifications on locations not (yet) described in the database, or to use a known wrong placement as decoy to calculate false localisation rate, for example A[Phospho] is used as a decoy residue by Ramsbottom *et al.* [10]. For this reason the decision to treat these cases as errors or warnings or even ignore them is left open for each implementation to decide for themselves.

6.2.2 Controlled vocabulary or ontology protein modification accession numbers

In case accession numbers from the supported CVs/ontologies are used to report protein modifications, full accession numbers MUST be used in all cases (no abbreviations in the names of the ontologies/CVs are allowed). The supported names are:
- Unimod: UNIMOD
- PSI-MOD: MOD
- RESID: RESID (only for top-down support)
- XL-MOD: XLMOD (only for cross-linking support)
- GNO: GNO (only for glycan support)

Examples of proper accession number usage:

EM[MOD:00719]EVEES[MOD:00046]PEK
EM[UNIMOD:35]EVEES[UNIMOD:56]PEK

The following examples are incorrect:

EM[M:00719]EVEES[M:00046]PEK
EM[U:35]EVEES[U:56]PEK

6.2.3 Delta mass notation

In addition to using CV/ontologies names and/or accession numbers, mass differences (delta masses) MAY be used to represent protein modifications.

Delta masses SHOULD only be used when the protein modification cannot be represented using a CV/ontology (e.g., if software does not use ontologies/CVs), when the modification (or combination of modifications) is ambiguous (e.g., coming from open modification searches or *de novo* approaches), or when it is unknown. Otherwise, protein modifications SHOULD be represented using Unimod, PSI-MOD, RESID, XL-MOD, or GNO CV parameters.

Mass differences MUST be expressed in Daltons between the coded amino acid and the observed mass. Positive mass shifts MUST be specified with a plus sign. Negative shifts MUST be specified with a negative sign. Monoisotopic masses MUST be used.

EM[+15.9949]EVEES[+79.9663]PEK
EM[+15.995]EVEES[-18.01]PEK

Interpretation of the actual delta masses is then left to the reader software.

Full ‘level 2-ProForma’ adds support for tagging masses with a CV; see section 7.2.

6.3 N-terminal and C-terminal modifications

The square brackets containing the modification MUST be located before the first amino acid in the sequence or after the last amino acid in the peptide sequence. In both cases, they are separated by a dash (-). Multiple modifications on the termini can be denoted with multiple modifications in square brackets. Examples:

[iTRAQ4plex]-EM[Oxidation]EVNES[Phospho]PEK

[iTRAQ4plex]-EM[Oxidation]EVNES[Phospho]PEK[iTRAQ4plex]-[Methyl]

PEPTIDEG-[Methyl][Amidated]

[Acetyl][Carbamyl]-QPEPTIDE

Disclaimer: these last two are syntactically valid, but these might not exist in biology.

6.4 Labile modifications

Labile modifications are those which are known to separate under certain experimental conditions during fragmentation and therefore are not visible in the fragmentation MS2 spectrum (i.e. the MS2 spectra are indistinguishable from spectrum not containing the modification). They are represented by curly brackets {}, not by square ones. Labile modifications MUST be located before the first amino acid sequence and before N-terminal modifications, if applicable. The examples below use the Glycan composition modifications which are only required in 'level-2 ProForma + glycan' section 10.2, but the concept of labile modifications is part of 'base-ProForma':

{Glycan:Hex}EM[Oxidation]EVNES[Phospho]PEK[iTRAQ4plex]

{Glycan:Hex}[iTRAQ4plex]-EM[Oxidation]EVNES[Phospho]PEK[iTRAQ4plex]

{Glycan:Hex}[iTRAQ4plex]-EM[Oxidation]EVNES[Phospho]PEK[iTRAQ4plex]-[Methyl]

One can also express multiple labile modifications using the following notation:

{Glycan:Hex}{Glycan:NeuAc}EMEVNESPEK

6.5 Representation of multiple modifications on the same amino acid residue

It is possible to represent two or more modifications on the same amino acid, terminal, or group of amino acids. The caret symbol (^), which can be used to represent multiple instances of the same unlocalised modification before the N-terminal end of the amino acid sequence (section 7.6.1) MUST NOT be used within the amino acid sequence. No extra character is required. Examples:

MPGLVDSNW[Oxidation][Carboxymethyl]PAPPESQE
PEPTIDEG-[Methyl][Amidated]
[Acetyl][Carbamyl]-QPEPTIDE
MPGLVDSNPAPPESQEKKPLK(PCCACPETKKARDACIIEKGEEHCGHLIEAHKECMRALGFKI)[Oxidation][Oxidation][half cystine][half cystine]

This is not be allowed because it uses the caret symbol:

MPGLVDSNPAPPESQEKKPLK(PCCACPETKKARDACIIEKGEEHCGHLIEAHKECMRALGFKI)[Oxidation]^2[half cystine][half cystine]

If a combination of modifications occurs together the combined term from any supported CV should be used. In the case this does not already exist a new modification should be created in advance.

If a modification occurs twice on a single site, combined modifications SHOULD be used instead of stacking single modifications. For example [U:Dimethyl] is preferred over specifying [U:Methyl] twice.

6.6 The information tag

General information or comments can be encoded using the ‘info’ tag like:
ELV[INFO:AnyString]IS
ELV[info:AnyString]IS

The information represented in an ‘info’ tag is considered non-standard (e.g. any text besides unpaired brackets) and does not need to be parsed.

“Info” tags can be split using the pipe character. Example of proper ‘info’ tag usage:

ELVIS[Phospho|INFO:newly discovered]K
ELVIS[Phospho|INFO:newly discovered|INFO:really awesome]K

The following comments would be invalid because of an unpaired bracket:

ELVIS[Phospho|INFO:newly]discovered]K
ELVIS[Phospho|INFO:newly[discovered]K

As a concrete example of its use, “Info” tags can be used to provide metadata about ProForma entities (e.g. date of creation, version of Unimod used, software used for creating it, and many others):

ELVIS[Phospho|INFO:newly discovered|INFO:Created on 2021-06]K
ELVIS[Phospho|INFO:newly discovered|INFO:Created by software Tool1]K

The ‘Info’ tag can also be used to add general information to a stretch of amino acids:

EDTAVYYC(SRWGGDGFAMDY)[Info:CDR3]WGQGTL

To provide general metadata on a full entity instead of on amino acids the peptidoform/proteoform names can be used as described in section 8.2.

7. Level of Compliance: Additional General Support (Level 2 ProForma)

Technical name: level 2-ProForma compliant

7.1 Ambiguous amino acids

In addition to the common and unusual amino acids defined in 6.1. There are some definitions for ambiguous amino acids (see http://www.insdc.org/documents/feature_table.html#7.4.3). These are:

- B: Aspartic Acid or Asparagine
- Z: Glutamic Acid or Glutamine
- J: Leucine or Isoleucine
- X: Any amino acid (see also section 7.3 *(Specifying a gap of known mass)*, for the use of X). We note that the character X itself is assigned zero mass in this notation, and should be followed by a modification to provide the residue's total mass.

In the case of B and Z there are two possible masses. Any implementation which uses masses in any way SHOULD handle the combinatorial expansion of these amino acids. So for example a peptide with two Bs has three possible masses: DD, DN/ND, and NN.

7.2 Delta masses with prefixes

This level of support extends the delta mass notation, see 6.2.3, with support for using prefixes for CVs/ontologies to provide more information.

If "canonical" delta masses are directly taken from a CV/ontology, the corresponding abbreviation to that CV/ontology MAY be used.

- Unimod: U
- PSI-MOD: M
- RESID: R (only for top-down support)
- XL-MOD: X (only for cross-linking support)
- GNO: G (only for glycan support)
- Custom: C

Examples of delta masses corresponding to CV/ontology entries:

EM[U:+15.9949]EVEES[U:+79.9663]PEK
EM[U:+15.995]EVEES[U:+79.966]PEK

The notation also supports the encoding of experimentally observed delta masses. In those cases, the prefix "Obs" MUST be used. The number of significant figures included in the delta mass depends on the accuracy of the available data and SHOULD be used as is by interpreters. Example:

EM[U:+15.995]EVEES[Obs:+79.978]PEK

7.3 Specifying a gap of known mass

This mechanism can be used to express a gap in the sequence of an unknown number of amino acids, but the corresponding mass difference is known. This is supported by the use of the unknown amino acid, section 7.1, indicated with the character X, followed by brackets indicating the total mass of the gap, meaning that the mass of X is actually zero.

RTAAX[+367.0537]WT

7.4 Support for elemental formulas

A modification representing a small molecular substructure or a functional group can be described by a chemical formula. The descriptor "Formula" MUST be used. Only elemental formulas are supported. Example of proper chemical formula usage:

SEQUEN[Formula:C12H20O2]CE
SEQUEN[Formula:[13C2]CH6N]CE

As no widely accepted specification exists for expressing elemental formulas, we have adapted a standard with the following rules (taken from https://github.com/rfellers/chemForma):

Formula Rule 1
A formula will be composed of pairs of atoms and their corresponding cardinality (two Carbon atoms: C2). Pairs SHOULD be separated by spaces but are not required to be. Atoms and cardinality SHOULD NOT be. Also, the Hill system for ordering (https://en.wikipedia.org/wiki/Chemical_formula#Hill_system) is preferred, but not required.

Example: C12H20O2 or C12 H20 O2

Formula Rule 2
Cardinalities must be positive or negative integer values. Zero is not supported. If a cardinality is not included with an atom, it is assumed to be +1.

Example: HN-1O2

Formula Rule 3
Isotopes will be handled by prefixing the atom with its isotopic number in square brackets. If no isotopes are specified, previous rules apply. If no isotope is specified, then it is assumed the natural isotopic distribution for a given element applies.

Example: [13C2][12C-2]H2N
Example: [13C2]C-2H2N

SEQUEN[Formula:[13C2][12C-2]H2N]CE
(here 2 ^{12}C atoms are replaced by 2 ^{13}C atoms)

See in section 12 (*Pending Issues*) how this mechanism could be extended in the future to support more complex molecular formulas.

7.5 Support for the joint representation of experimental data and its interpretation

The pipe character "|" is used to represent protein modifications simultaneously with CV/ontology names and/or accession numbers, and delta masses. As explained in section 4.2.6 *(Delta mass notation)* it is possible to represent both canonical delta masses and experimental observations, allowing the representation of both interpretation (using CV/ontology names/accession numbers) and experimental observations (delta masses).

Example:
ELVIS[U:Phospho|+79.966331]K

Showing both the interpretation and measured mass:
ELVIS[U:Phospho|Obs:+79.978]K

Other combinations between CV/ontology names, accession numbers, and delta masses using synonyms are allowed, though they MUST be synonymous terms. Some examples:

ELVIS[Phospho|O-phospho-L-serine]K
ELVIS[UNIMOD:21|MOD:00046]K
ELVIS[UNIMOD:21|Phospho]K
ELVIS[Phospho|O-phospho-L-serine|Obs:+79.966]K

Ambiguous cases are also allowed because they can be used to represent "comparable" information.

ELVIS[Obs:+79.966|Phospho|Sulfo]K

Highly different modifications SHOULD NOT be joined as it would be difficult for readers to correctly interpret. It is however acknowledged that readers can choose to implement the parsing in different ways. Some tools may always take CV terms, others could take delta masses, and so on.

7.6 Support for the representation of ambiguity in the modification position

This notation is used to represent ambiguous modified sites, associated positions and associated probabilities or scores.

This notation is not yet supported for crosslinking modifications (see section 12.9), except for the case of disulfide cross-linkers which may be represented with ambiguous positions using the PSI-MOD term for "half-cystine" (MOD:00798), as noted in section 9.2.3.iii.

7.6.1 Unknown modification position

The positions of some modifications may be unknown. In this case, protein modifications are represented using square brackets that MUST be located on the left side of the amino acid sequence. The symbol '?' is used to indicate that the actual position of the modification is unknown.

[Phospho]?EM[Oxidation]EVTSESPEK

In case of multiple modifications with an unknown location, two options are possible to represent them:

(i) Listing them separately as in this example of two phosphorylations:

[Phospho][Phospho]?[Acetyl]-EM[Oxidation]EVTSESPEK

(ii) Indicating the concrete modification only once but using the caret (^) symbol to represent the number of occurrences of the modification.

[Phospho]^2?[Acetyl]-EM[Oxidation]EVTSESPEK

N-terminal modifications MUST be the last ones written, just next to the amino acid sequence. For example:

Wrong: [Acetyl]-[Phospho]^2?EM[Oxidation]EVTSESPEK
Right: [Phospho]^2?[Acetyl]-EM[Oxidation]EVTSESPEK

7.6.2 Indicating a possible set of modification positions

The position of a modification may be unknown but belong to a known set of possible sites. In this case, the possible positions for the modifications may be indicated. The rules that MUST be followed are:

(i) Groups of possible sites for a modification are represented immediately following the modification notation using the symbol #, followed by an arbitrary label consisting of alphanumeric characters ([A-Za-z0-9]+ in regular expression notation). Note that the label prefix #XL is a special case that MUST be reserved for crosslinkers only.

(ii) A single preferred location for the modification MUST be specified, so that the sequence can be easily rendered in visualization tools. The preferred location for the modification is indicated by the position of the modification notation in the amino acid sequence.

In this example, '#g1' is used as the arbitrary label:

EM[Oxidation]EVT[#g1]S[#g1]ES[Phospho#g1]PEK

This is read as a named group 'g1' indicates that a phosphorylation exists on either T5, S6 or S8, and S8 is the preferred location because the notation 'Phospho' is placed at this position.

The following example is not valid because a single preferred location must be chosen for a modification:

EM[Oxidation]EVT[#g1]S[Phospho#g1]ES[Phospho#g1]PEK

7.6.3 Representing ranges of positions for the modifications

Ranges of amino acids as possible locations for the modifications may be represented using parentheses within the amino acid sequence. Some examples:

PRT(ESFRMS)[+19.0523]ISK
PRT(EC[Carbamidomethyl]FRMS)[+19.0523]ISK

The caret symbol (^), which can be used to represent multiple instances of the same unlocalised modification before the N-terminal end of the amino acid sequence (Section 6.3), is not allowed within the amino acid sequence.

Overlapping ranges represent a more complex case and are not yet supported, and so, the following example would NOT be valid:

P(RT(ESFRMS)[+19.0523]IS)[+19.0523]K

7.6.4 Indicating modification position preference and localisation scores

There are two options to represent this type of information. The values of the modification localisation scores can be indicated in parentheses within the same group and brackets.

Example of proper localisation score usage:

EM[Oxidation]EVT[#g1(0.01)]S[#g1(0.09)]ES[Phospho#g1(0.90)]PEK

Scores for the modification position can be expressed as probabilities and/or FLR (False Localisation Rate), but the actual meaning of the scores is not reported. The preferred location of the modification notation reflects the value of the scores. If there is a tie in the value of the localisation scores, one preferred position needs to be chosen by the writer.

An additional option to represent localisation scores is to leave the position of the modification as unknown using the '?' notation but report the localisation modification scores at specific sites.

Example of proper usage of localisation scores with unknown modification site notation:

[Phospho#s1]?EM[Oxidation]EVT[#s1(0.01)]S[#s1(0.09)]ES[#s1(0.90)]PEK

7.6.5 Representing scoring for ranges of positions for a modification

Ranges of amino acids as possible locations for the modifications may also be accompanied by scoring using the same notation. Some examples:

PRT(ESFRMS)[+19.0523#g1(0.01)]ISK[#g1(0.99)]
PR[#g1(0.91)]T(EC[Carbamidomethyl]FRMS)[+19.05233#g1(0.09)]ISK

7.7 Representation of amino acid sequence ambiguity

Ambiguity in the amino acid sequence needs to be represented in some cases, e.g. to represent sequence changes that do not change the mass of the peptidoform/proteoform, but are not known. One concrete example is the need to encode the results of *de novo* sequencing tools. The way to encode this information is to use a parenthesis and a question mark '?' including the ambiguous sequence represented in a preferred way. Examples:

(?DQ)NGTWEM[Oxidation]ESNENFEGYM[Oxidation]K

In this example the ambiguous amino acid sequence could be any of GGE, GAD, NE, or DQ, in any order.

(?N)NGTWEM[Oxidation]ESNENFEGYM[Oxidation]K

In this example the ambiguous amino acid sequence could be either N or GG.

7.8 Using prefixes for Unimod and PSI-MOD named modifications

Prefixes are allowed to be used for Unimod and PSI-MOD names. Using U: for Unimod and M: for PSI-MOD allows precisely defining which ontology was used for the determination of the modification. Usage of this prefix is not mandatory as outlined in section 6.2.1. Supporting the optional specification of this prefix in this compliance level.

EM[U:Oxidation]EVEES[U:Phospho]PEK
EM[M:L-methionine sulfoxide]EVEES[M:O-phospho-L-serine]PEK

8. Level of Compliance: Top-Down Extension

Technical name: level 2-ProForma + top-down compliant

8.1 Support for the additional ontology RESID

Although RESID is included in PSI-MOD, this reference system is still used in the top-down proteomics community. So this extension allows modifications to be used from the RESID ontology. The RESID ontology can be found on this url: https://proteininformationresource.org/resid/. This is allowed both in named modification, section 6.2.1, indexed modification, section 6.2.2, and tagged mass modifications, section 7.2.

EM[RESID:AA0581]EVEES[RESID:AA0037]PEK

8.2 Peptidoform/proteoform names

Peptidoforms and proteoforms can be named to organise complex definitions. This is done by prepending the peptidoform by '(>' followed by the name and closed with ')'. Any use of parentheses '(', ')' within a name MUST be balanced. The name cannot start with an angled bracket '>'. In the below examples the names are used to give human meaning to the two peptidoforms in a chimeric spectrum. Supporting chimeric spectra is part of advanced complexity section 11.4.

(>Trypsin)AANSIPYQVSLNS+(>Keratin)AKEQFERQTA

There is no format imposed on the name, but it is recommended to follow FASTA format principles/conventions. The following example uses the name to record the full fasta identifier in a cross-linked peptidoform ion. Cross-linked peptidoforms are part of the cross-linker extension section 9.2.

(>P07225 Vitamin K-dependent protein S OS=Homo sapiens OX=9606 GN=PROS1 PE=1 (SV=1) RANGE=12..42)GGK[xlink:dss[138]#XLDSS]IEVQLK//(>P07225 Vitamin K-dependent protein S OS=Homo sapiens OX=9606 GN=PROS1 PE=1 SV=1)KVESELIK[#XLDSS]PINPR/4

Additionally the other layers in a compound peptidoform can be named as well, using ‘(>’ for peptidoforms, ‘(>>’ for peptidoform ions, and ‘(>>>’ for compound peptidoform ions. Below is an example of an antibody Fab where the heavy chain and light chain are cross-linked together and the names are used to distinguish the chains as well as distinguish the Fab from the Fc, which is the cleaved part of the heavy chain to produce the Fab. If multiple levels of name tags are specified they MUST appear with the highest level first. So the following example shows the correct ordering of ‘(>>>A)(>>B)(>C)`

(>>>Trastuzumab Fab and coeluting Fc)(>>Fab)(>Heavy chain)EVQLVESGGGLVQPGGSLRLSC[M:l-cystine (cross-link)#XL1]AASGFNIKDTYIHWVRQAPGKGLEWVARIYPTNGYTRYADSVK GRFTISADTSKNTAYLQMNSLRAEDTAVYYC[#XL1]SRWGGDGFYAMDYWGQG TLVTVSSASTKGPSVFPLAPSSKSTSGGTAALGC[M:l-cystine (cross-link)#XL2]LVKDYFPEPVTVSWNSGALTSGVHTFPAVLQSSGLYSLSSVVTV PSSSLGTQTYIC[#XL2]NVNHKPSNTKVDKKVEPKSC[M:l-cystine (cross-link)#XL3]DKT//(>Light chain)DIQMTQSPSSLSASVGDRVTITC[M:l-cystine (cross-link)#XL4]RASQDVNTAVAWYQQKPGKAPKLLIYSASFLYSGVPSRFSGSRS GTDFTLTISSLQPEDFATYYC[#XL4]QQHYTTPPTFGQGTKVEIKRTVAAPSVFIFPP SDEQLKSGTASVVC[M:l-cystine (cross-link)#XL5]LLNNFYPREAKVQWKVDNALQSGNSQESVTEQDSKDSTYSLSS TLTLSKADYEKHKVYAC[#XL5]EVTHQGLSSPVTKSFNRGEC[#XL3]+(>Fc)HTCP PCPAPELLGGPSVFLFPPKPKDTLMISRTPEVTCVVVDVSHEDPEVKFNWYVDGV EVHNAKTKPREEQYNSTYRVVSVLTVLHQDWLNGKEYKCKVSNKALPAPIEKTI SKAKGQPREPQVYTLPPSREEMTKNQVSLTCLVKGFYPSDIAVEWESNGQPENNY KTTPPVLDSDGSFFLYSKLTVDKSRWQQGNVFSCSVMHEALHNHYTQKSLSLSP GK

9. Level of Compliance: Cross-Linking Extension

Technical name: level 2-ProForma + cross-linking compliant

9.1 Support for additional controlled vocabulary XL-MOD

This extension allows modification to be used from the XL-MOD CV. The CV can be found on this url: https://github.com/HUPO-PSI/xlmod-CV. This is allowed both in named modification, section 6.2.1; indexed modification, section 6.2.2; and tagged mass modifications, section 7.2.

9.2 Support for cross-linkers

Crosslinked sites MUST be represented by immediately following the modification notation using the prefix #XL, followed by an arbitrary label consisting of alphanumeric characters ([A-Za-z0-9]+ in regular expression notation). Cross-linker modification notations MUST be mentioned once only.

Any annotation made with the symbol # represents a way of linking different locations within the amino acid sequence. From ProForma version 2.0 it is used for representing cross-linkers, branched peptides and for grouping protein modifications (including glycans) to represent ambiguity (section 7.6).

While commonly cross-linkers are attached to one (hydrolysed) or two sites the specification also allows the definition of cross-linkers attached to more sites.

It is acknowledged that the current version of ProForma does not provide support for all possible use cases involving cross-linked peptides. In the future, it is expected that a specific extension for this type of information can be developed.

9.2.1 Crosslink notation (within the same peptide)

Cross-linker modification notations MUST be mentioned once only. This example shows a DSS crosslink between two lysines:

EMEVTK[XLMOD:02001#XL1]SESPEK[#XL1]

This second example shows a DSS crosslink between two lysines and an EDC cross-link between two other lysines:

EMK[XLMOD:02000#XL1]EVTKSE[XLMOD:02010#XL2]SK[#XL1]PEK[#XL2]AR

A “dead end” crosslink happens regularly with bifunctional crosslinkers when one side attaches and the other hydrolyses before attaching. These modifications are annotated at only one site.

EMEVTK[XLMOD:02001#XL1]SESPEK

EMEVTK[XLMOD:02001]SESPEK

9.2.2 Representing inter-chain crosslinks

Inter-protein or inter-chain connections are supported using // to separate the crosslinked peptides. This notation is similar to IUPAC condensed notation for inter-protein connections.

SEK[XLMOD:02001#XL1]UENCE//EMEVTK[XLMOD:02001#XL1]SESPEK

SEK[XLMOD:02001#XL1]UENCE//EMEVTK[#XL1]SESPEK

It is acknowledged by the authors that more complex scenarios are possible when representing inter-chain crosslinks, including a higher number of linked peptides, directionality, etc. It is envisioned that when these use cases become a clear requirement in the future, a dedicated working group can extend these guidelines.

9.2.3 Representing disulfide linkages

Disulfide bonds may be represented using five possible notations:

(i) Using the PSI-MOD term for "L-cystine (cross link)" (MOD:00034) to explicitly describe the cross-link using the cross-linking notation:

EVTSEKC[MOD:00034#XL1]LEMSC[#XL1]EFD
EVTSEKC[L-cystine (cross-link)#XL1]LEMSC[#XL1]EFD

There are more complex examples that are possible. For instance, another example with inter-chain disulfide bonds is insulin:

FVNQHLC[MOD:00034#XL1]GSHLVEALYLVC[MOD:00034#XL2]GERGFFYTPKA
\\GIVEQC[MOD:00034#XL3]C[#XL1]TSIC[#XL3]SLYQLENYC[#XL2]N

As mentioned above, more complex scenarios are possible which will need to be resolved in future versions.

(ii) Using the XLMOD term XLMOD:02009 similarly to case (i) above:

EVTSEKC[XLMOD:02009#XL1]LEMSC[#XL1]EFD
EVTSEKC[X:Disulfide#XL1]LEMSC[#XL1]EFD

(iii) Using the Unimod term for "Xlink:Disulfide" [UNIMOD:2020] similarly to case (1) above:

EVTSEKC[UNIMOD:2020#XL1]LEMSC[#XL1]EFD
EVTSEKC[Xlink:Disulfide#XL1]LEMSC[#XL1]EFD
EVTSEKC[Xlink:Disulfide#XL1]LEK//MSC[#XL1]EFDR

(iv) Using the PSI-MOD term for "half cystine" (MOD:00798) if the pairing is not known. Since the term is only for half the link, it must be specified on all involved sites with no group tag:

EVTSEKC[half cystine]LEMSC[half cystine]EFD
EVTSEKC[MOD:00798]LEMSC[MOD:00798]EFDEVTSEKC[MOD:00798]LEMSC[MOD:00798]EFD

(v) Using the Unimod term for "Dehydro" [UNIMOD:374] similarly to case (iv) above:

EVTSEKC[UNIMOD:374]LEMSC[UNIMOD:374]EFD
EVTSEKC[Dehydro]LEMSC[Dehydro]EFD

This SHOULD NOT be done unless there is a separate modification of the second cross-linked residue:
EVTSEKC[Dehydro#XL1]LEMSC[#XL1][Dehydro]EFD

This may produce a peptidoform that has the correct intact mass, but does not semantically produce the correct association of the linker.

9.3 Representation of branched peptides

Branched peptides can be expressed using the same notation used for representing two cross-linked peptides, but using the term #BRANCH (see above). Examples:

a) ETFGD[MOD:00093#BRANCH]//D[#BRANCH]ATER

b) Cross-linked via a sidechain:

AVTKYTSSK[MOD:00134#BRANCH]//AGKQLEDGRTLSDYNIQKESTLHLVLRLRG-[#BRANCH]
Where a sidechain of a Lysine from peptide 1 is linked to the C-term of the peptide 2 via amidation (-H2O)

Taken from:
https://www.news-medical.net/whitepaper/20180329/Synthesizing-Unsymmetrically-Branched-Peptides.aspx

10. Level of Compliance: Glycan Extension

Technical name: level 2-ProForma + glycans compliant

10.1 Representation of glycans using the GNO ontology as CV

Glycans that are currently included in Unimod or PSI-MOD (individual or very short chains) MAY be represented that way. If the glycans are not included in either PSI-MOD or Unimod, the GNO ontology SHOULD be used. The use of accession numbers is preferred since accession numbers and names are often the same.

- GNO (Glycan Naming Ontology, https://www.ebi.ac.uk/ols/ontologies/gno).

Examples of proper glycan notation:

Encoding Hex 5 HexNAc 4 NeuAc 1:
NEEYN[GNO:G59626AS]K

Encoding Hex 8 HexNAc 2 and Hex 5 HexNAc 2:
YPVLN[GNO:G62765YT]VTMPN[GNO:G02815KT]NSNGKFDK

The same mechanisms for expressing labile modifications and ambiguity in the modification position applicable to other types of modifications SHOULD be used for glycans as well (see following sections, e.g. sections 6.4 and 7.6).

There are more complex cases, where ambiguity can be caused by multiple combinations between labile and non-labile glycans attached to the same amino acid sequence. A possible mechanism to represent these more complex cases is available in section 12 (*Pending issues*). A further limitation comes from the restricted set of glycans in GNO. We expect that these issues will be solved as the glyco(proteomics) community develops in the near future.

10.2 Representation of glycan composition

Glycan residues (generic monosaccharides) can be represented using the descriptor “Glycan”. If glycan symbols conflict with themselves or element symbols in such a way that ambiguities occur, we will consider requiring spaces between 'atoms' (see Formula Rule #1).

Example: Hex2HexNAc

SEQUEN[Glycan:HexNAc1Hex2]CE

The supported list of monosaccharides in ProForma is included below. It is worth noting that the masses and elemental compositions included below for each monosaccharide are those resulting after each of them are condensed with the amino acid chain.

Hex: Hexose, 162.0528 Da, C6H10O5

HexNAc: N-Acetyl Hexose, 203.0793 Da, C8H13N1O5
HexS: Hexose Sulfate, 242.0096 Da, C6H10O8S1
HexP: Hexose Phosphate, 242.0191 Da, C6H11O8P1
HexNAcS: N-Acetyl Hexose Sulfate, 283.0361 Da, C8H13N1O8S1
HexN: 161.0688 Da, C6H11N1O4
HexNS: 241.0256 Da, C6H11N1O7S1
dHex: Deoxy-Hexose, 146.0579 Da, C6H10O4
aHex: 176.0321 Da, C6H8O6
en,aHex: 158.0215 Da, C6H6O5
Neu: Neuraminic acid, 249.0849 Da, C9H15N1O7
NeuAc: N-acetyl Neuraminic Acid / Sialic Acid, 291.0954 Da, C11H17N1O8
NeuGc: N-glycolyl Neuraminic Acid, 307.0903 Da, C11H17N1O9
Sug: 42.0106 Da, C2H2O1
Tri: 72.0211 Da, C3H4O2
Tet: Tetrose, 102.0317 Da, C4H6O3
Pen: Pentose, 132.0422 Da, C5H8O4
Hep: Heptose, 192.0634 Da, C7H12O6
Oct: Octose, 222.0740 Da, C8H14O7
Non: Nonose, 252.0845 Da, C9H16O8
Dec: Decose, 282.0951 Da, C10H18O9
Fuc: Fucose, 146.0579 Da, C6H10O4 (a particular stereochemical assignment of dHex abundant in mammals)
Sulfate: 79.9568 Da, O3S1
Phosphate: 79.9663 Da, H1O3P1

The most up to date list of supported monosaccharides (in two different formats, obo and json) can be found at:

https://github.com/HUPO-PSI/ProForma/tree/master/monosaccharides

For other monosaccharides not listed here a molecular formula can be used to describe the sugar moiety. This is done by wrapping the formula in curly braces. These formulas can be defined to happen multiple times with the same occurrence specifier as normal monosaccharides.

This example encodes a HexNAc as a chemical formula instead of the name:
SEQUEN[Glycan:{C8H13N1O5}1Hex2]CE
SEQUEN[Glycan:{C8H13[15N1]O5}1Hex2]CE

Additionally, charged monosaccharides can be defined using the charged molecular formula definition from section 11.1:
SEQUEN[Glycan:{C8H13N1O5Na1:z+1}1Hex2]CE

This notation is only required to be supported for level 2-ProForma + glycans + advanced complexity supporting readers.

11. Level of Compliance: Advanced Complexity Support

Technical name: level 2-ProForma + advanced complexity compliant

11.1 Charged formulas

Charges can be specified on chemical formulas using the following suffix '<formula>:z<charge>'. This allows localised charges to be defined. The mass of a charged formula is the mass of its constituents plus -z times the mass of an electron.

SEQUEN[Formula:Zn1:z+2]CE

If the reader also supports the glycan composition notation, section 10.2, charged glycan compositions can be defined using the same syntax:
SEQUEN[Glycan:{C8H13N1O5Na1:z+1}1Hex2]CE

11.2 Controlling placement of modifications

Protein modifications of unknown position can result in many combinations of possible locations. To add more knowledge into the placement of these modifications, and limit the number of possible combinations, there are four ways to control the exact placement of modifications of unknown position:

- Position: a list of positions where this modification is allowed. The positions are written the same as global modifications, see 11.3.2. If this is not specified the allowed position SHOULD be the allowed positions as defined in the CV/ontology, see 6.2.1.4.
- Limit: the maximal number of this modification of unknown position on one position. If this value is not specified a default value of 1 is used. This setting can only be applied on modifications of unknown position with a specified occurrence, see 7.6.1 ii.
- CoMKP or ColocaliseModificationsOfKnownPosition: this controls if this modification of unknown position can be placed on a location where a placed modification is located. By default this is not allowed.
- CoMUP or ColocaliseModificationsOfUnknownPosition: this control if this modification of unknown position can coexist at the same location as any other modifications of unknown position. By default this is not allowed.

When the database defines many positions for a modification, or the modification has no defined positions because it is a mass or formula the 'Position' rule can be used to limit the positions to the interesting ones:

[Formula:Zn:z+2|Position:N-term,C-term]^5[Carbamidomethyl|Position:C]^5?MDPETC PCPSGGSCTCADSCKCEGCKCTSCKKSCCSCCPAECEKCAKDCVCKGGEAAEAE AEKCSCCQ

PEPTI(MERMERMERM)[Oxidation|Position:M][Oxidation|Position:M]DE

PEPTI(MERMERMERM)[+32|Position:E]PEPTIDE

The 'Limit' rule can be used to allow a modification to be placed multiple times on the same amino acid, in the first case the oxidation is only allowed once per amino acid, while in the second example it is allowed up to two times.

[Oxidation]^4?PEPTIDE
[Oxidation|Limit:2]^4?PEPTIDE

The 'CoMKP' rule in this examples allows the co-occurrence of the Oxidation together with the Phospho on the T:

[Oxidation|CoMKP]?PEPT[Phospho]IDE

In these examples the modifications are never allowed both on the same position, because of the default value of 'CoMUP':

PETIEM[Dioxidation#1][Oxidation#2]REM[#1][#2]REM[#2]RM[#1]PEPTIDE

[Oxidation#1][Oxidation#2]?PTIM[#1][#2]ERM[#1][#2]ERM[#1][#2]ERM[#1][#2]DE

So if 'CoMUP' is applied to both modifications they can be cooccur:

PETIE(MEME)[Dioxidation|CoMUP][Oxidation|CoMUP]P

One peptidoform that fits this definition is: PEPTIEM[Dioxidation][Oxidation]EMEP

11.3 Representation of global modifications

This mechanism MAY be used for modifications that apply to all relevant residues in the peptide/protein amino acid sequence. These modifications MAY be represented by the use of the characters "<" and ">" on the left side of the sequences. A couple of use cases are envisioned:

11.3.1 Representation of global isotope replacement

This might be used in the case of synthetic peptides with 100% incorporation.

Example: Consider extension for 13C on all residues:
Carbon 13: <13C>ATPEILTVNSIGQLK
Nitrogen 15: <15N>ATPEILTVNSIGQLK
Deuterium: <D>ATPEILTVNSIGQLK

The representation of multiple isotopes is also possible. They can be located in any order.

Both Carbon 13 and Nitrogen 15: <13C><15N>ATPEILTVNSIGQLK

Distributions of isotope masses could be supported in future work.

11.3.2 Fixed protein modifications

This mechanism can be useful especially in the case of full proteoforms. The affected amino acid MUST be indicated using @. If the modification affects (N or C) terminal positions this MUST be indicated with the terms 'N-term' or 'C-term', respectively. Additionally if the modification only affects a certain amino acid at a terminal location this SHOULD be added after the terminal using a colon ":" followed by the amino acid. If more than one residue is affected by the same modification, they MUST be comma separated. Examples:

<[S-carboxamidomethyl-L-cysteine]@C>ATPEILTCNSIGCLK
<[MOD:01090]@C>ATPEILTCNSIGCLK
<[Oxidation]@C,M>MTPEILTCNSIGCLK
<[TMT6plex]@K,N-term>ATPEILTCNSIGCLK
<[Oxidation]@W,C-term:G>QATPEILTWCNSIGCLKG
<[Gln->pyro-Glu]@N-term:Q><[Oxidation]@W,C-term:G>QATPEILTWCNSIGCLKG
<[Amidated]@C-term>QATPEILTWCNSIGCLKG

While this last example is allowed it SHOULD be written as follows:

QATPEILTWCNSIGCLKG-[Amidated]

Similarly, while it is allowed to list fixed modifications that do not occur in the specified peptide users SHOULD remove these:

<[Oxidation]@C,M>EPELVESGGGLAQP

SHOULD be written as:

EPELVESGGGLAQP

11.4 Representation of multiple peptidoform assignments in chimeric spectra

In bottom-up approaches, in the case of chimeric spectra, more than one peptidoform sequence MAY be potentially assigned to a single mass spectrum. In this case, multiple peptidoform sequences MUST be separated by the plus sign (+). Example:

EMEVEESPEK/2+ELVISLIVER/3
PEPTIDE+OTHERPEPTIDE

11.5 Representation of charges

The charge of a peptidoform can be specified after the definition. This specifies the charge of the peptidoform, excluding any local charges of elemental formulas (section 11.1). Any positive charges are defined as addition of protons H+, any negative charges are defined as addition of electrons e-.

PEPTIDE/2

If the charge carriers are different, these can be defined instead of the global charge using the formula with charge notation (section 11.1). The charge carriers have to be enclosed by square brackets and if there are multiple they have to be separated by commas.

PEPTIDE/[Na:z+1]
PEPTIDE/[Na:z+1,H:z+1]

If a charge carrier is present multiple times the occurrence specifier from the unknown location modifications can be used.

PEPTIDE/[Na:z+1^2]

If both localised charges and charge carriers are used all of these should be accounted for in the total charge of the peptide. The following example has a total charge of +4.

PEPT[Formula:Zn:z+2]IDE/[Na:z+1^2]

11.6 Ion notation

A MSn precursor can be defined using the corresponding entries in Unimod, such as ‘a-type-ion’ (UNIMOD:140) and ‘b-type-ion’ (UNIMOD:2132).
PEPTID-[a-type-ion]
PEPTID-[b-type-ion]

More complex ion types can be defined as their full chemical composition, for example for a d ion from the above a ion this could be the definition:
PEPTID[Formula:H-1C-1O-2|Info:d-ion]-[a-type-ion]

This notation can be used to indicate MSn precursors:

MS2 precursor: SFFLYSKLTVDKSRWQQGNV

MS3 precursor: SFFLYSKLTV-[b-type-ion]

12. Pending Issues - Future developments

Additionally, there are several use cases that are NOT currently supported in the current version of the specification. These complications are left open and will ideally be addressed in future versions, after the community has gained more experience with the common cases. The objective here is to document those cases appropriately and propose some possible solutions for representing the information in future versions of ProForma.

12.1 Representation of cyclic peptides

Cyclic peptides are only currently supported if they can be represented using the supported CVs/ontologies for protein modifications. The following examples represent possible ways to represent cyclic peptides, but these solutions need to be formalised and PSI-MOD modifications created.

1) Cyclic peptide with C- and N-termini bound together at the peptide backbone level
Kalata B1 (PubChemID: 46231131, UniProtKB: P56254)
[*MOD:nnnnnn*#XL1]-RNGLPVCGETCVGGTCNTPGCTCSEPVCT-[#XL1]

where *MOD:nnnnnn* would be a new PSI-MOD term to represent backbone cyclisation involving the amidation between a C-terminal carboxylate and a N-terminal amine, with mass difference of O-1H-2 (-18 Da).

2) Cyclic peptide with C- and N-termini bound together at the peptide backbone level with 3 disulfide bonds
Retrocyclin 1 (PubChem ID 16130540). The exact structure is the following:
https://pubchem.ncbi.nlm.nih.gov/compound/Retrocyclin-1#section=Biologic-Description&fullscreen=true
[*MOD:nnnnnn*#XL1]-RC[MOD:00798.DS1]IC[MOD:00798.DS2]GRGIC[MOD:00798.DS2]RC[MOD:00798.DS1]IC[MOD:00798.DS3]GRGIC[MOD:00798.DS3]-[#XL1]

where *MOD:nnnnnn* would be a new PSI-MOD term to represent backbone cyclisation involving the amidation between a C-terminal carboxylate and a N-terminal amine, with a mass difference of O-1H-2 (-18 Da).

3) Cyclic peptide with C-terminal COOH condensed to a sidechain NH2
3a) peptide with no other PTM
LEIK[N6-(L-asparagyl)-L-lysine#XL1]KIPHDN[#XL1]

3b) A real case scenario: Topitracin (PubChem ID:6474109)

[[N-[2-[1-amino-2-methylbutyl]-4,5-dihydro-4-thiazolyl]carbonyl]-Leucine]-LE[D-Glutamic
acid]IK[M:N6-(L-asparagyl)-L-lysine#XL1]K[M:D-Ornithine]I[M:D-alloisoleucine]P[D-Phenylalanine]HD[M:D-Aspartic acid]N[#XL1]

12.2 Representation of ambiguity when different glycans are attached to the same amino acid sequence

Multiply glycosylated peptides, especially under vibrational/collisional dissociation, may fragment in ways that allow sequencing the peptide backbone without completely characterizing the glycan sites. Instead, only the aggregate composition can be determined based on the precursor peptide mass. In such cases, only the glycosylation may be known, by motif for N-glycan or there may be several possible sites. Alternatively, the total number of glycosylation sites may be unknown (O-glycans), with the aggregate glycan composition may be spread across positions in unknown proportions.

There is a need to express that a site is a possible glycosylation site as well as a mechanism to express the total amount of glycan composition shared across these sites. The latter is achieved by using a labile modification to prefix the total composition. There are multiple proposals for expressing putative site assignment:

Proposal 1. Use PSI-MOD glycosylated residue modifications.
{Glycan:Hex 10 HexNAc 4}YPVLN[MOD:00006]VTMPN[MOD:00006]NSNGKFDK

This peptide hosts two N-glycans, where the glycan class is known from the required motifs on the sequence, and that it is multiply glycosylated because no single N-glycan with the aggregate composition is biosynthetically feasible. This proposal denotes the inferred glycosylation sites using the PSI-MOD “N-glycosylated residue” term. This forces the reader to treat this group differently, where the modification is inferred to be the labile glycan modification and that the modification may be split amongst each site, assigning zero or more monosaccharides to each group position.

Pros:

- Conveys extra metadata about the glycan type
- Uses an existing term

Cons:

- Introduces new semantics for a modification that is not explicitly conveyed notationally, namely that this modification is not observable, but just encodes positional information.
- For complex and ambiguous O-glycopeptides, this method would pull double-duty with ambiguity notation.

Proposal 2. Use ambiguity groups.
{Glycan:Hex 10 HexNAc 4}YPVLN[#g1]VTMPN[Glycan#g1]NSNGKFDK

The same case with Proposal 1, but instead of adding extra baggage to an existing term, this proposal uses ambiguity groups to denote possible positions, and mark one group with a new “Glycan” key, which adds the same labile modification inference step.

Pros:

- Uses an ambiguity-specific mechanism to signal ambiguity.

Cons:

- Adds a new component to ambiguity group interpretation that parsers must now be prepared to handle.
- No ability to communicate glycan type at the site level.

A complex O-glycopeptide example
{Glycan:Hex 5 HexNAc 5}PEPSTAT[Glycan#g1]IS[#g1]T[#g1]ICS[#g1]S[#g1]T[#g1]RIKES[#g1]IT[#g1]ES[#g1]

Fragmentation may demonstrate that some S/T residues are not putative sites, while the distribution of glycan composition is still not known amongst the remaining sites. The true solution might be:

PEPSTATISTICS[Glycan:HexNAc Hex]S[Glycan:HexNAc Hex]TRIKES[Glycan:HexNAc Hex]IT[Glycan:HexNAc Hex]ES[Glycan:HexNAc Hex]

Or PEPSTATISTICSS[Glycan:HexNAc 2 Hex 2]TRIKES[Glycan:HexNAc Hex]IT[Glycan:HexNAc Hex]ES[Glycan:HexNAc Hex], or any permutation thereof.

Proposal 3. Use cross-linking-like notation
{Glycan:Hex 10 HexNAc 4.G1}YPVLN[Glycan:#G1]VTMPN[Glycan:#G1]NSNGKFDK

The only differentiating feature of this proposal from Proposal 2 is that it isolates the notational change solely within the Glycan tag handling, which reduces the burden on implementers who do not want to support glycosylation.

12.3 Representation of rare amino acids not supported by the one letter code

This use case is currently not supported. These SHOULD be handled through their representations in one of the supported ontologies/CVs.

12.4 Representation of average masses

During the development of the format, it was acknowledged that, in the case of top-down proteomics approaches, there could be cases where monoisotopic masses are unknown, and then average masses need to be used. At the moment, monoisotopic masses are the only ones formally allowed, but this MAY have to change in future changes.

12.5 Representation of lipids

These SHOULD be handled through their representations in one of the supported ontologies/CVs. However, a similar mechanism to the one described in section 10.2, Representation of glycan composition, could be implemented for lipid molecules.
Examples:

SEQUEN[Lipid:OleicAcid]CE
SEQUEN[Lipid:PalmiticAcid]CE

It is envisioned that when this use case becomes a clear requirement in the future, a dedicated working group can extend these specific guidelines.

12.6 Distribution of isotopes in the sequence

The representation of the distributions of isotopes for global modifications (section 11.3.1) is not supported in the current version of the specification. A mechanism will need to be envisioned to support this use case in future versions.

12.7 Representation of molecular formula

Elemental formulas are supported by the current version of ProForma (section 7.4), and molecular formulas may be supported in the future if it would prove helpful. For example, specifying branching in a PTM structure. A molecular formula may include repeated (condensed) sections using parentheses and an extra cardinality.

Examples:
CH3(CH2)4CH3
SEQUEN[Formula:CH3(CH2)4CH3]CE

At the time of writing, an extension on using chemical structures (e.g., SMILES) to define protein modifications has been suggested.

12.8 Representation of an overlapping range of possible modification positions

Notation of ambiguous localization currently supports non-overlapping ranges. A possible representation of overlapping ranges, that may be considered in the future, uses a grouping tag for both parentheses.

Examples:
PROT([#g1]EOC[Carbamidomethyl]FORMS)[+19.0523#g1]ISK
PR([#g1]OT([#g2]EOC[Carbamidomethyl]FOR)[+19.0523#g1]MS)[+19.0523#g2]ISK
PROT([#g1]([#g2]EOC[Carbamidomethyl]FORMS)[+19.0523#g2]IS)[+19.0523#g1]K

12.9 Representation of ambiguous crosslinker modification positions

Notation for ambiguous crosslinker modification positions is not supported in this version of ProForma but may be supported in the future.

12.10 Metadata related to ProForma entries

At present, metadata related to ProForma entries (e.g. date of creation, software, version of ontology used, etc) cannot be provided in a standardised manner. The creation of an additional metadata file could be considered in future versions.

12.11 Representation of sequences coming from non-mass spectrometry-based proteomics approaches

The ProForma notation could be made compatible with non-mass spectrometry proteomics approaches, such as nanopore and Edman-based sequencing, and others that

will face the same notation challenges. A mechanism will need to be envisioned to support these use cases in future versions.

12.12 Representation of ambiguity of locations across linear peptides

A labile modification (see section 6.4) and modifications of unknown position (see section 7.6) have to be defined as part of a peptidoform. To allow for one of these modifications to be defined as ambiguous across multiple peptidoforms a new notation has to be devised. An example use case for this would be a glycan in a chimeric spectrum where a fragmentation scheme is used that does not allow for the localisation of the glycan. In this case both the glycan and the peptidoforms could be fully known based on the fragmentation but for the glycan it could still be unclear to which peptidoform it belongs.

12.13 Representation of DNA and RNA

Adding DNA and RNA to ProForma would enable easier writing and reporting of these molecules for MS experiments. For example this would allow the use of these in spectral libraries stored in the mzSpecLib format (as precursors, if needed mzPAF should be extended with DNA/RNA fragmentation naming). At a high level DNA, RNA, and proteins are similar: both built of distinct building blocks in a linear chain with a defined backbone. So writing DNA in ProForma could be done like so: ATGCTG[+34]A. The basic sequence block should be changed from amino acid to base pair and the modifications changed to a database of base pair modifications. This would still allow for all complexity of ProForma (modifications of unknown position, labile, global, chimeric, cross-linking etc). One issue that is left then is to be able to tell apart DNA/RNA sequences from peptides, especially if both are allowed to be mixed in the same definition. This should be done with some kind of tag before the now called 'peptidoform'. One proposal is to use <!DNA>, <!RNA>, and <!AA>. Examples:

<!DNA>ATGTCATCGT
<!RNA>AUGUCAUCGU
<!DNA>ATCG[Formula:OH#XL1]T//<!AA>HYTGC[#XL1]R

This tag is unambiguously and easily parse-able, as the only current options for things starting with < are global modifications and global isotope replacements. Global modifications <[mod]@location> are easily distinguished by the use of the [and global isotope replacements always use a number (except for <D> but that is also easily recognised). This tag should be placed before the peptidoform name tag <!DNA>(>My nice sequence 001)ATGCTAGT which means that any global modification comes before. If this tag is missing the sequence defaults to a protein/peptide which would make this addition easily backwards compatible.

- https://genesilico.pl/modomics/ could be used as a database for RNA modifications.
- https://www.insdc.org/submitting-standards/feature-table/#7.4.3 contains all base pairs also with ambiguous characters.

Open questions:

- How to make global modifications work: right now they assume AA.
- What is the best notation for double stranded DNA/RNA?
- How to differentiate 5’-to-3’ from 3’-to-5’?

13. Appendix Ⅰ: Formal grammar

The full formal grammar as described in section 5.1. A parser automatically generated from this definition together with tests based on the examples in the specification can be found on GitHub https://github.com/HUPO-PSI/ProForma/blob/master/grammar/. The used EBNF notation follows ISO/IEC 14977:1996.

```
(* Main entry point *)
proforma = compoundPeptidoformIon;
compoundPeptidoformIon = [nameCompoundPeptidoformIon],
{modGlobal}, {peptidoformIon, "+"}, peptidoformIon;

(* Global modifications *)
modGlobal = "<", modGlobalOption, ">";
modGlobalOption = ISOTOPE | (modDefined, "@",
modGlobalLocation, {",", modGlobalLocation});
ISOTOPE = D | (INT, ELEMENT);
modGlobalLocation = (C,"-",T,E,R,M, [":", aminoAcid]) |
(N,"-",T,E,R,M, [":", aminoAcid]) | aminoAcid;

(* Peptidoform ion: a single entity that is at some point
one molecule *)
peptidoformIon = [namePeptidoformIon], {peptidoform, "//"},
peptidoform, [peptidoformCharge];
peptidoformCharge = "/", (SIGNEDINT, | "[", adductIon,
{",", adductIon}, "]");
adductIon = formula, CHARGE, [occuranceSpecifier];

(* Peptidoform: a single linear sequence of aminoacids *)
peptidoform = [namePeptidoform], {globalModUnknownPos},
{modLabile}, [modNTerm], sequence, [modCTerm];
globalModUnknownPos = mod, [occuranceSpecifier], {mod,
[occuranceSpecifier]}, '?';
modLabile = "{", modInternalDefined, "}";

(* Main sequence elements *)
sequence = sequenceSection, {sequenceSection};
sequenceSection = ambiguousAminoAcid | modRange |
sequenceElement;
ambiguousAminoAcid = "(?", sequenceElement,
{sequenceElement}, ")";
modRange = "(", sequenceElement, {sequenceElement}, ")",
mod, {mod};
sequenceElement = aminoAcid, {modOrLabel};
aminoAcid = LETTER;
nameCompoundPeptidoformIon = "(>>>", NAMETEXT, ")";
namePeptidoformIon = "(>>", NAMETEXT, ")";
```

```
namePeptidoform = "(>", NAMETEXT, ")";

(* Modfications *)
modOrLabel = mod | ("[", modLabel, "]");
mod = "[", modInternal, "]";
modUnknownPos = "[", modSingle, [modLabel], , "]";
unknownPosSetting = ("|",P,O,S,I,T,I,O,N,":",
modGlobalLocation, {",", modGlobalLocation}) | ("|",
L,I,M,I,T, ":", INT) | ("|", C,O,M,K,P) | ("|", C,O,M,U,P);
modInternal = modSingle, [modLabel], {"|", modSingle,
[modLabel]};
modDefined = "[", modInternalDefined, "]";
modInternalDefined = modSingle, {"|", modSingle};
modSingle = modFormula | modGlycan | modMass |
modAccession| info | modName;

modFormula = F,O,R,M,U,L,A,":", formula, [CHARGE];
formula = WS, formulaElement, {WS, formulaElement}, WS;
formulaElement = ("[", WS, INT, WS, ELEMENT, WS,
[SIGNEDINT], WS, "]")|(ELEMENT, WS, [SIGNEDINT]);

modGlycan = G,L,Y,C,A,N,":", monosaccharide, [INT], WS,
{monosaccharide, [INT], WS};
monosaccharide = ("{",formula, [CHARGE],"}") |
MONOSACCHARIDE;

(* Defined to need a sign, which is not needed from a
parser perspective *)
modMass = [modMassCV, ":"], SIGN, NUMBER;
modMassCV =  CVAbbreviation | (O,B,S);
CVAbbreviation = U|M|R|X|G;

(* An accession number which is a full integer expect when
GNO is used *)
modAccession = (CVName, ":", INT) | (G,N,O,":",
ALPHANUMERIC, {ALPHANUMERIC});
CVName = (U,N,I,M,O,D) | (M,O,D) | (R,E,S,I,D) |
(X,L,M,O,D);

(* A named modification *)
modName = [CVAbbreviation, ":"], MODTEXT;

(* Info tag *)
info = I,N,F,O,":", MODTEXT;

modNTerm = modOrLabel, [modOrLabel], "-";
modCTerm = "-", modOrLabel, [modOrLabel];
```

```
(* Possible labels/named modifications *)
modLabel = "#", (modLabelXL | modLabelBranch |
modLabelAmbiguous);
modLabelXL = X,L, ALPHANUMERIC, {ALPHANUMERIC};
modLabelBranch = B,R,A,N,C,H;
modLabelAmbiguous = ALPHANUMERIC, {ALPHANUMERIC}, ["(",
SIGNEDNUMBER, ")"];

(* Reused syntactic elements *)
occuranceSpecifier = "^", INT;
CHARGE = ":", Z, SIGNEDINT;

MONOSACCHARIDE = (S,U,L,F,A,T,E)|(P,H,O,S,P,H,A,T,E)
|(F,U,C)|(N,E,U,A,C)|(N,E,U,G,C)|(N,E,U)|(D,H,E,X)
|(E,N,",",A,H,E,X)|(A,H,E,X)|(H,E,X,N,A,C,S)|(H,E,X,N,A,C)
|(H,E,X,N,S)|(H,E,X,S)|(H,E,X,P)|(H,E,X,N)|(H,E,X)|(S,U,G)
|(T,R,I)|(T,E,T)|(P,E,N)|(H,E,P)|(O,C,T)|(N,O,N)|(D,E,C);

ELEMENT = (H,E)|(L,I)|(B,E)|(N,E)|(N,A)|(M,G)|(A,L)|(S,I)
|(C,L)|(A,R)|(C,A)|(S,C)|(T,I)|(C,R)|(M,N)|(F,E)|(C,O)|(N,I
)
|(C,U)|(Z,N)|(G,A)|(G,E)|(A,S)|(S,E)|(B,R)|(K,R)|(R,B)|(S,R
)|(Z,R)|(N,B)|(M,O)|(T,C)|(R,U)|(R,H)|(P,D)|(A,G)|(C,D)|(I,
N)
|(S,N)|(S,B)|(T,E)|(X,E)|(C,S)|(B,A)|(L,A)|(C,E)|(P,R)|(N,D
)|(P,M)|(S,M)|(E,U)|(G,D)|(T,B)|(D,Y)|(H,O)|(E,R)|(T,M)|(Y,
B)
|(L,U)|(H,F)|(T,A)|(R,E)|(O,S)|(I,R)|(P,T)|(A,U)|(H,G)|(T,L
)|(P,B)|(B,I)|(P,O)|(A,T)|(R,N)|(F,R)|(R,A)|(A,C)|(T,H)|(P,
A)|(N,P)|(P,U)|(A,M)|(C,M)|(B,K)|(C,F)|(E,S)|(F,M)|(M,D)|(N
,O)|(L,R)|(R,F)|(D,B)|(S,G)|(B,H)|(H,S)|(M,T)|(D,S)|(R,G)|(
C,N)|(N,H)|(F,L)|(M,C)|(L,V)|(T,S)|(O,G)|U|W|I|Y|V|K|S|B|C|
N|O|F|H|P;

DIGIT = "0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9";
INT = DIGIT, {DIGIT};
SIGN = "+"|"-";
SIGNEDINT = [SIGN], INT;
SIGNORSIGNEDINT = SIGNEDINT | SIGN;
(* Is full floating point used? If so the following can be
added: [E, [SIGN], INT, [".", INT]]*)
NUMBER = INT, [".", INT];
SIGNEDNUMBER = [SIGN], NUMBER;
A = "A"|"a";
B = "B"|"b";
C = "C"|"c";
```

```
D = "D"|"d";
E = "E"|"e";
F = "F"|"f";
G = "G"|"g";
H = "H"|"h";
I = "I"|"i";
J = "J"|"j";
K = "K"|"k";
L = "L"|"l";
M = "M"|"m";
N = "N"|"n";
O = "O"|"o";
P = "P"|"p";
Q = "Q"|"q";
R = "R"|"r";
S = "S"|"s";
T = "T"|"t";
U = "U"|"u";
V = "V"|"v";
W = "W"|"w";
X = "X"|"x";
Y = "Y"|"y";
Z = "Z"|"z";
LETTER =
A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z;
ALPHANUMERIC = LETTER|DIGIT;
(* Not exhaustively listed all possible characters.
Anything unicode graphic not '|', '#', '[', or ']' is
possible, also an arbitrary number and nesting of internal
brackets is allowed, but this EBNF grammar is limited in
this capacity *)
MODGRAPHICCHARACTER = ALPHANUMERIC|SIGN|"
"|"?"|"."|","|"<"|">"|"{"|"}"|"("|")"|"_"|"="|":"|";"|"/"|"
\\"|"'"|'"'|"*"|"~"|"`"|"@"|"$"|"%"|"^"|"&";
MODTEXT = {MODGRAPHICCHARACTER}, {"[",
{MODGRAPHICCHARACTER}, "]"}, {MODGRAPHICCHARACTER};
NAMEGRAPHICCHARACTER = ALPHANUMERIC|SIGN|"
"|"?"|"."|","|"<"|">"|"{"|"}"|"["|"]"|"_"|"="|":"|";"|"/"|"
\\"|"'"|'"'|"*"|"~"|"`"|"@"|"$"|"%"|"^"|"&";
STARTNAMEGRAPHICCHARACTER = ALPHANUMERIC|SIGN|"
"|"?"|"."|","|"<"|"{"|"}"|"["|"]"|"_"|"="|":"|";"|"/"|"\\"|
"'"|'"'|"*"|"~"|"`"|"@"|"$"|"%"|"^"|"&";
NAMETEXT = STARTNAMETEXT, NAMETEXTSECTION;
STARTNAMETEXT = (STARTNAMEGRAPHICCHARACTER) | ("(",
NAMETEXTSECTION, ")");
NAMETEXTSECTION = {NAMEGRAPHICCHARACTER}, {"(",
{NAMEGRAPHICCHARACTER}, ")"},{NAMEGRAPHICCHARACTER};
```

```
WS = {" "|"\t"};
```

14. Appendix Ⅱ: Data schema

The full markdown data schema as described in section 5.2.

Properties for CompoundPeptidoformIon

- **fixed_modifications** *(array)*
  - **Items** *(object)*: A fixed modification. Cannot contain additional properties.
    - **modification**: Fixed modifications cannot be cross-linked or ambiguous. Refer to *#/$defs/tags*.
    - **position_rules** *(array)*
      - **Items**: Refer to *#/$defs/position_rule*.
- **isotope_replacement** *(array)*
  - **Items** *(object)*: A global isotope replacement, 'D' is also allowed to be written as a shortcut, but this is encoded as 2H. Cannot contain additional properties.
    - **element**: Refer to *#/$defs/element*.
    - **isotope**: Nucleon count. Refer to *#/$defs/positive_integer*.
- **name** *(string)*: The compound peptidoform ion name, expressed with (>>>name) in ProForma 2.1.
- **peptidoform_ions** *(array, required)*
  - **Items** *(object)*: A peptidoform ion. Cannot contain additional properties.
    - **charge**: Refer to *#/$defs/global_charge*.
    - **name** *(string)*: The peptidoform ion name, expressed with (>>name) in ProForma 2.1.
    - **peptidoforms** *(array, required)*
      - **Items** *(object)*: A Peptidoform. Cannot contain additional properties.
        - **c_term_modifications** *(array)*
          - **Items**: Refer to *#/$defs/modification*.
        - **labile_modifications** *(array)*: Labile modifications cannot be cross-linked or ambiguous.
          - **Items**: Refer to *#/$defs/tags*.
        - **n_term_modifications** *(array)*
          - **Items**: Refer to *#/$defs/modification*.
        - **name** *(string)*: The peptidoform name, expressed with (>name) in ProForma 2.1.
        - **sequence** *(array, required)*
          - **Items**: Refer to *#/$defs/sequence*.
        - **unlocalised_modifications** *(array)*
          - **Items**: Refer to *#/$defs/modification_ambiguous*.

Definitions

- **amino_acid**: An amino acid including unconventional ones (U/O) and ambiguous ones (X/J/B/Z). Must be one of: ["A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M", "N", "O", "P", "Q", "R", "S", "T", "U", "V", "W", "X", "Y", "Z"].
- **charged_formula** *(object)*: A formula that can be charged, expressed in ProForma as Formula:C2H6:z+2. Can contain additional properties.
  - **charge** *(integer)*: The charge state for this formula.
  - **formula**: Refer to *#/$defs/formula*.
- **CV**: One of the five supported controlled vocabularies. Must be one of: ["Unimod", "PSI-MOD", "RESID", "GNOme", "XL-MOD"].
- **element**: All elements. Must be one of: ["He", "Li", "Be", "Ne", "Na", "Mg", "Al", "Si", "Cl", "Ar", "Ca", "Sc", "Ti", "Cr", "Mn", "Fe", "Co", "Ni", "Cu", "Zn", "Ga", "Ge", "As", "Se", "Br", "Kr", "Rb", "Sr", "Zr", "Nb", "Mo", "Tc", "Ru", "Rh", "Pd", "Ag", "Cd", "In", "Sn", "Sb", "Te", "Xe", "Cs", "Ba", "La", "Ce", "Pr", "Nd", "Pm", "Sm", "Eu", "Gd", "Tb", "Dy", "Ho", "Er", "Tm", "Yb", "Lu", "Hf", "Ta", "Re", "Os", "Ir", "Pt", "Au", "Hg", "Tl", "Pb", "Bi", "Po", "At", "Rn", "Fr", "Ra", "Ac", "Th", "Pa", "Np", "Pu", "Am", "Cm", "Bk", "Cf", "Es", "Fm", "Md", "No", "Lr", "Rf", "Db", "Sg", "Bh", "Hs", "Mt", "Ds", "Rg", "Cn", "Nh", "Fl", "Mc", "Lv", "Ts", "Og", "U", "W", "I", "Y", "V", "K", "S", "B", "C", "N", "O", "F", "H", "P"].
- **formula** *(array)*: A molecular formula.
  - **Items** *(object)*: Cannot contain additional properties.
    - **element**: Refer to *#/$defs/element*.
    - **isotope**: Nucleon count. Refer to *#/$defs/positive_integer*.
    - **occurance** *(integer, required)*: The number of times this part occurs.
- **global_charge**
  - **Any of**
    - *integer*
    - : Refer to *#/$defs/global_charge_carriers*.
- **global_charge_carriers** *(array)*
  - **Items** *(object)*: Cannot contain additional properties.
    - **charged_formula**: Refer to *#/$defs/charged_formula*.
    - **occurance** *(number, required)*
- **modification**: A modification, could be ambiguous, cross-linked, or just a single modification (which might be defined multiple times).
  - **Any of**
    - : Refer to *#/$defs/modification_ambiguous*.
    - : Refer to *#/$defs/modification_cross_linker*.
    - : Refer to *#/$defs/tags*.
- **modification_ambiguous**
  - **Any of**
    - : Refer to *#/$defs/modification_ambiguous_primary*.
    - : Refer to *#/$defs/modification_ambiguous_secondary*.

- **modification_ambiguous_primary** *(object)*: The primary definition of an ambiguous modification. Cannot contain additional properties.
    - **comkp** *(bool)*: ColocaliseModificationsOfKnownPosition: this controls if this modification of unknown position can be placed on a location where a placed modification is located. By default this is not allowed.
    - **comup** *(bool)*: ColocaliseModificationsOfUnknownPosition: this control if this modification of unknown position can coexist at the same location as any other modifications of unknown position. By default this is not allowed.
    - **label** *(string, required)*: The part after the #. Length must be at least 1.
    - **limit**: The maximal number of this modification of unknown position on one position. If this value is not specified a default value of 1 is used. This setting can only be applied on modifications of unknown position with a specified occurrence. Refer to *#/$defs/positive_integer*.
    - **position** *(array)*: A list of positions where this modification is allowed. The positions are written the same as global modifications.
        - **Items**: Refer to *#/$defs/position_rule*.
    - **score** *(number)*: The score for this location of this ambiguous modification. Minimum: 0. Maximum: 1.
    - **tags**: Refer to *#/$defs/tags*.
- **modification_ambiguous_secondary** *(object)*: A reference to an ambiguous modification. Cannot contain additional properties.
    - **label** *(string, required)*: The part after the #. Length must be at least 1.
    - **score** *(number)*: The score for this location of this ambiguous modification. Minimum: 0. Maximum: 1.
- **modification_cross_linker** *(object)*: A cross-linked modification, if the tags are missing this is a reuse of a modification that should be defined in another place, if the label is missing this indicates that this is a #BRANCH, and not a #XL. Cannot contain additional properties.
    - **label** *(string)*: The part after the #XL, or if this is a #BRANCH this entire property is missing. Length must be at least 1.
    - **tags**: Refer to *#/$defs/tags*.
- **monosaccharide**
    - **Any of**
        - : Refer to *#/$defs/monosaccharide_name*.
        - : Refer to *#/$defs/charged_formula*.
- **monosaccharide_name**: A named monosaccharide. Must be one of: ["aHex", "Dec", "dHex", "en,aHex", "Fuc", "Hep", "Hex", "HexN", "HexNAc", "HexNAcS", "HexNS", "HexP", "HexS", "Neu", "NeuAc", "NeuGc", "NeuAc", "NeuGc", "Non", "Oct", "Pen", "Phosphate", "S", "Sug", "Sulfate", "Tet", "Tri"].
- **position_rule** *(object)*: Cannot contain additional properties.
    - **amino_acid**: Refer to *#/$defs/amino_acid*.
    - **terminal**: Must be one of: ["Anywhere", "NTerm", "CTerm"].
- **positive_integer** *(integer)*
- **sequence**: A region in a peptide sequence.
    - **Any of**

        - : Refer to *#/$defs/sequence_ambiguous*.
        - : Refer to *#/$defs/sequence_element*.
        - : Refer to *#/$defs/sequence_region*.
- **sequence_ambiguous** *(array)*
    - **Items**: Refer to *#/$defs/sequence_element*.
- **sequence_element** *(object)*: Cannot contain additional properties.
    - **amino_acid**: Refer to *#/$defs/amino_acid*.
    - **modifications** *(array)*
        - **Items**: Refer to *#/$defs/modification*.
- **sequence_region** *(object)*: Cannot contain additional properties.
    - **modifications** *(array, required)*
        - **Items**: Refer to *#/$defs/modification*.
    - **sequence** *(array, required)*
        - **Items**: Refer to *#/$defs/sequence_element*.
- **tag_accession** *(object)*: The accession for a modification. Cannot contain additional properties.
    - **accession** *(string, required)*: The accession number, or for GNOme and RESID the string. Length must be at least 1.
    - **cv**: Refer to *#/$defs/CV*.
- **tag_formula**: A Formula:xxx modification. Refer to *#/$defs/charged_formula*.
- **tag_glycan** *(array)*: A glycan composition modification.
    - **Items** *(object)*: Cannot contain additional properties.
        - **monosaccharide**: Refer to *#/$defs/monosaccharide*.
        - **occurance** *(integer, required)*: The number of times this part occurs.
- **tag_info** *(string)*: A INFO tag modification.
- **tag_mass** *(object)*: A mass modification. Cannot contain additional properties.
    - **cv**: Refer to *#/$defs/CV*.
    - **mass** *(number, required)*
- **tag_name** *(object)*: A named modification. Cannot contain additional properties.
    - **cv**: Refer to *#/$defs/CV*.
    - **name** *(string, required)*: Length must be at least 1.
- **tags** *(array)*: A single modification which could consist of multiple definitions.
    - **Items**
        - **Any of**
            - : Refer to *#/$defs/tag_accession*.
            - : Refer to *#/$defs/tag_formula*.
            - : Refer to *#/$defs/tag_glycan*.
            - : Refer to *#/$defs/tag_info*.
            - : Refer to *#/$defs/tag_mass*.
            - : Refer to *#/$defs/tag_name*.

15. Authors Information

Douwe Schulte
Utrecht University
d.schulte@uu.nl

Juan Antonio Vizcaíno
European Bioinformatics Institute (EMBL-EBI), Hinxton, Cambridge, United Kingdom
juan@ebi.ac.uk

Eric W. Deutsch
Institute for Systems Biology, Seattle WA, USA
edeutsch@systemsbiology.org

Joshua A. Klein
Boston University, Boston MA, USA
joshua.adam.klein@gmail.com

Ralf Gabriels
Ghent University, Ghent, Belgium; VIB-UGent Center for Medical Biotechnology, VIB, Ghent, Belgium
ralf.gabriels@ugent.be

Petr Novák
Institute of Microbiology, The Czech Academy of Sciences, 14220 Prague, Czech Republic
pnovak@biomed.cas.cz

Dimitry Loginov
Institute of Microbiology, The Czech Academy of Sciences, 14220 Prague, Czech Republic
dmitry.loginov@biomed.cas.cz

The authors of ProForma 2.0 were as follows:

Juan Antonio Vizcaíno, European Bioinformatics Institute (EMBL-EBI), Hinxton, Cambridge, United Kingdom
Eric W. Deutsch, Institute for Systems Biology, Seattle WA, USA
Pierre-Alain Binz, CHUV Lausanne University Hospital, Lausanne, Switzerland
Ryan Fellers, Northwestern University, Evanston IL, USA
Anthony J. Cesnik, Stanford University, Stanford CA, USA
Joshua A. Klein, Boston University, Boston MA, USA
Tim Van Den Bossche, Ghent University, Ghent, Belgium; VIB-UGent Center for Medical Biotechnology, VIB, Ghent, Belgium

Ralf Gabriels, Ghent University, Ghent, Belgium; VIB-UGent Center for Medical Biotechnology, VIB, Ghent, Belgium
Yasset Perez-Riverol, European Bioinformatics Institute (EMBL-EBI), Hinxton, Cambridge, United Kingdom
Jeremy Carver, University of California San Diego, San Diego CA, USA
Shin Kawano, Toyama University of International Studies, Toyama, Japan
Benjamin Pullman, University of California San Diego, San Diego CA, USA
Nuno Bandeira, University of California San Diego, San Diego CA, USA
Paul M. Thomas, Northwestern University, Evanston IL, USA
Richard Leduc, Northwestern University, Evanston IL, USA

16. Contributors

In addition to the authors, a number of additional contributions have been made during the specification document. The contributors who actively participated to the documentation of either ProForma 2.0 or ProForma 2.1 (not included already as authors) are:

- Brian L. Frey, University of Wisconsin-Madison, Madison, WI, USA
- Alexander Leitner, ETH Zurich, Zurich, Switzerland
- Luis Mendoza, Institute for Systems Biology, Seattle, WA, USA
- Gerben Menschaert, Ghent University, Ghent, Belgium
- Jim Shofstahl, Thermo Fisher Scientific, San Jose, CA, USA
- Zhi Sun, Institute for Systems Biology, Seattle, WA, USA
- Leah V. Schaffer, University of Wisconsin-Madison, Madison, WI, USA
- Michael R. Shortreed, University of Wisconsin-Madison, Madison, WI, USA
- Veit Schwämmle, University of Southern Denmark, Odense, Denmark
- Wout Bittremieux, University of California San Diego, San Diego CA, USA

We would also like to acknowledge the contributions of the peer reviewers that participated in the PSI review process of ProForma 2.0:
- Gloria Sheynkman and Erin Jeffery (University of Virginia, VA, USA).
- Xiaowen Liu (Tulane University School of Medicine, LA, USA).

Finally, we are also appreciative of the contributions made by the Executive Board of the CTDP, led by Paul Danis.

17. Intellectual Property Statement

The PSI takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain

a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the PSI Chair.

The PSI invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this recommendation. Please address the information to the PSI Chair (see contact information at PSI website).

18. Copyright Notice

19. Glossary

All non-standard terms are already defined in detail in section 3.

20. References

1. LeDuc, R.D., Schwammle, V., Shortreed, M.R., Cesnik, A.J., Solntsev, S.K., Shaw, J.B., Martin, M.J., Vizcaino, J.A., Alpi, E., Danis, P. *et al.* (2018) ProForma: A Standard Proteoform Notation. *Journal of proteome research*, 17, 1321-1325.
2. LeDuc, R.D. Deutsch, E.W., Binz, P.A., Fehlers, R.T., Cesnik, A.J., Klein, J.A., Van Den Bossche, T., Gabriels, R., Perez-Riverol, Y., Carver, J., Bittremieux, W., Kawano, S., Pullman, B., Bandeira, N., Thomas, P.M., Kelleher, N., Vizcaíno, J.A. (2022). Proteomics Standards Initiative's ProForma 2.0: unifying the encoding of Proteoforms and Peptidoforms. *Journal of proteome research* 21(4):1189-1195.
3. https://topdownproteomics.github.io/ProteoformNomenclatureStandard/.
4. Smith, L., Kelleher, N. The Consortium for Top Down Proteomics (2013) Proteoform: a single term describing protein complexity. *Nature Methods,* 10, 186–187
5. Binz, P.A., Shofstahl, J., Vizcaíno, J.A., Barsnes, H., Chalkley, R.J., Menschaert, G., Alpi, E., Clauser, K., Eng, J.K., Lane, L. *et al.* (2019) Proteomics Standards Initiative Extended FASTA Format. *Journal of proteome research*, 18, 2686-2692.
6. Rosenberger, G., Liu, Y., Rost, H.L., Ludwig, C., Buil, A. Bensimon, A., Soste, M., Spector, T.D., Dermitzakis, E.T., Collins, B.C. *et al.* (2017) Inference and quantification of peptidoforms in large sample cohorts by SWATH-MS. *Nature Biotechnology*, 35(8):781-788.
7. Lermyte, F., Tsybin, Y.O., O'Connor, P.B. Loo, J.A. (2019) Top or Middle? Up or Down? Toward a Standard Lexicon for Protein Top-Down and Allied Mass Spectrometry Approaches. *Journal of the American Society of Mass Spectrometry*, 30(7):1149-1157.

8. Deutsch, E.W., Perez-Riverol, Y., Carver, J., Kawano, S., Mendoza, L., Van Den Bossche, T., Gabriels, R., Binz, P.A., Pullman, B., Sun, Z., Shofstahl, J., Bittremieux, W., Mak, T.D., Klein, J., Zhu, Y., Lam, H., Vizcaíno, J.A., Bandeira, N. (2021). Universal Spectrum Identifier for mass spectra. *Nature Methods,* 18(7):768-770.
9. Cornwell, O., Radford, S. E., Ashcroft, A. E., & Ault, J. R. (2018). Comparing Hydrogen Deuterium Exchange and Fast Photochemical Oxidation of Proteins: a Structural Characterisation of Wild-Type and ΔN6 β2-Microglobulin. In *Journal of the American Society for Mass Spectrometry*(Vol. 29, Issue 12, pp. 2413–2426). American Chemical Society (ACS). https://doi.org/10.1007/s13361-018-2067-y
10. Ramsbottom, K. A., Prakash, A., Riverol, Y. P., Camacho, O. M., Martin, M.-J., Vizcaíno, J. A., Deutsch, E. W., & Jones, A. R. (2022). Method for Independent Estimation of the False Localization Rate for Phosphoproteomics. *Journal of proteome research*, 21(7), 1603–1615. https://doi.org/10.1021/acs.jproteome.1c00827