Notes from the PEFF discussion at 2015 PSI spring workshop

The following comments were provided by Pierre-Alain prior to the session (he was not present at the session) :

-        The PEFF document requires a reread/proofread

- The terms that need a specific format to populate the associated information need to clarified some (for instance DB :0001011   Variant), one possibility is to use regexp, in order to define the allowed format (for instance     (223|223|[KR])      ) ; one proposal is to define a DB term that represents the regexp to use, and associate it with a relationship ; this allows to get rid of an external mapping file that would only define these formats

-        Is there any requirement to add many terms such as those presented at the end of the obo file

-        Confirm the decisions labelled in the commentaries of the word specification file

-        Confirm that the obo file can be included into the PSI-MS CV (and handled by the PSI editor, i.e. Gerhard). Where should we put it? Under a specific PEFF branch?

The next steps from Pierre-Alain were:

-        Prepare an updated version of the PEFF reader (see with Harald)

-        Ask neXtProt to update the writer

-        Re-do the small example

-        Compile all of these docs (specs, obo file, simple example, writer, reader) and prepare the submission to the PSI doc process

Comments from the discussion at the workshop session:

* There was a lot of concern from some software implementers about making the \Variant feature too complicated. Concerns were voiced that if the allowed variation is too complex, search engines would likely not implement PEFF or at least these features. The following scenarios were discussed and the apparent consensus of the group present is recorded.
* Is (223|223|K) legal ? YES, of course, this is the simplest case
* Is (223|225|KPA) legal (i.e. no indel, consecutive substitution)? YES
* Is (223|223|K)(223|223|R) legal as a way of encoding multiple variants ? YES
* Is (223|223|[KR]) legal as a way of encoding multiple variants? UNDECIDED
* Is (223|223|KPMELISA) legal (i.e. insertion) ? NO. This was thought to be too hard for search engine software to handle. This must be a new entry
* Is (223|235|A) legal (i.e. deletion) ? NO. This was thought to be too hard for reader software. This must be a new entry
* The most vocal members of the group seemed to feel that encoding indels in this way will be too complex for search engine software developers to be persuaded to implement
* We should make this assumption explicit: if a residue is changed, then all PTMs at the position are obsolete (explicitly disallow PTMs on variants). This clearly can happen. As an example: default residue 74 is F; residue 74 can be S; if residue 74 is S, then it can be phosphorylated. Yet another example: default residue 76 is T; residue 76 can also be S; either way it can be phosphorylated. Consensus seemed to be that this is getting too complicated. How would this be encoded?
* Should we make examples for these scenarios on how they would be encoded?
* Would we consider an “advanced variation” flag where any writer that dares to write “advanced” variations like indels or residue-conditional PTMs would set the advanced flag at the top, and some search engines would be free to decline to parse such a file, or ignore advanced features. Search engines could ignore advanced features anyway, but should it be explicitly considered in the specification?

Other questions/comments:

* How are FDR calculations affected by target sequences with variants and ptms and decoys that are not?
* Who else is interested in working on this: Patrick Pedrioli, …?
* There is a sleeper feature which might cause some consternation: whereas one entry is usually composed of:

>accession description

PEPTIDESEQUENCE

it is explicitly permitted by the current spec to have:

>accession1 description1(CTRL+A)>accession2 description2

PEPTIDESEQUENCE

where “(CTRL+A)” indicates ASCII character 001 SOH (Start of heading)

I (Eric Deutsch) don’t necessarily object to this, but there are potentially significant ramifications for PEFF reader software that everyone should be aware of. This would appear to be a space-saving feature that introduces conceptual complexity that everyone must implement.