UniRule

Format of rules for automated protein annotation in UniProtKB/Swiss-Prot

Brigitte Boeckmann, Alexandre Gattiker, Edouard de Castro

This is a draft version. Do not distribute.

Table of Contents

1 Introduction to UniRule format . . . 1

2 Header section . . . 2

2.1 AC line . . . 2
2.2 DC line . . . 2
2.3 TR line . . . 4
2.3.1 *TR motif* line . . . 4
2.3.2 *TR Metamotif* line . . . 5
2.4 XX line . . . 5
2.5 Names line . . . 6
2.6 Function line . . . 6
2.7 Internal comments . . . 6

3 Annotation section . . . 8

3.1 ID line . . . 8
3.2 DE line . . . 8
3.3 GN line . . . 9
3.4 CC line . . . 9
3.5 DR line . . . 10
3.6 KW line . . . 12
3.7 GO line . . . 13
3.8 FT line . . . 13
3.8.1 *FT From* line . . . 14
3.8.2 *FT feature* line . . . 15
3.8.3 *FT constraints* line . . . 18

4 Computing section . . . 20

4.1 Warn line . . . 20
4.2 Chop line . . . 20
4.3 Size line . . . 21
4.4 Related line . . . 21
4.5 Repeats line . . . 22
4.6 Topology line . . . 22
4.7 Template line . . . 23
4.8 Example line . . . 23
4.9 Scope block . . . 24
4.10 Fusion block . . . 24
4.11 Duplicate line . . . 25
4.12 Plasmid line . . . 25
4.13 Comments block . . . 26

5 History section . . . 27

6 Control statements . . . 28

6.1 Case statement . . . 28
6.1.1 Ternary logic . . . 29
6.1.2 Condition line *c!* or *c?* . . . 30

7 Hidden information 32
7.1 The keyword *Repeat* 32
7.2 The *FT constraints* line for the feature key *DISULFID* 32

Appendix A UniAln 33

Appendix B Sample UniRules entries 35

1 Introduction to UniRule format

UniRule is a format describing conditional annotation templates (rules) used by UniProtKB/Swiss-Prot automated annotation projects. It defines annotation which will be generated for (selected) predicted features.

Structure

Each UniRule entry consists of the following parts.

- The *Header section* contains the UniRule accession number(s), the data class, identifier(s) of the feature(s) that trigger(s) the rule, as well as some basic information on the rule.
- The *Annotation section.* All UniProtKB/Swiss-Prot annotation relevant for a UniRule is indicated. Not all annotation lines apply to all true hits. Therefore, *case* statements restrict all or part of the annotation to further conditions like the taxonomic group or the size of a protein.
- The *Computing section* covers information on a protein, domain or site relevant to programs. It is supposed to be checked automatically and it is still incomplete.
- A // (terminator) line which contains no data or comments and designates the end of an entry.

Application

'`Protein rules`'
Used to annotate protein family : HAMAP rules.

'`Domain/Site rules`'
Used to annotate protein domain or site : ProRules.

Explanation of terms

'`motif`' Any sequence feature predictor or discriminator, such as a pattern, profile or profile-HMM.

'`taxcode`' A mnemonic code of at most 5 alphanumeric characters specific for a taxonomic species used in UniProtKB/Swiss-Prot.

2 Header section

The *Header section* contains technical information about the UniRules.

2.1 AC line

Each UniRule starts with one or more *AC* (ACcession number) lines. The format of the accession number is for intance '`PRU`' followed by 5 digits for ProRules and '`MF_`' followed by 5 digits for HAMAP rules.

Application:

```
Protein rule: mandatory
Domain/Site rule: mandatory
```

Format:

```
AC   primary accession number;[ secondary accession numbers;]
```

Examples:

```
AC   PRU00001;
AC   PRU00002; PRU99998; PRU99999;
AC   MF_00022;
```

Note: The UniRule file name in the CVS directory corresponds to the primary accession number of a UniRule.

2.2 DC line

The *DC* (Data Class) meta data line specifies the rule annotation category.

Application:

```
mandatory
```

Format:

```
DC   data_class;[ auto]
```

Examples:

```
DC   Protein; auto
DC   Domain;
DC   Site;
```

The *data_class* is either of the following values:

- '`Protein`'
- '`Topo_domain`'
- '`Domain`'
- '`Repeat`'
- '`Segment`'
- '`Comp_bias`'

$Id: unirules.txi,v 1.113 2016/08/08 08:05:00 ecastro Exp $

- ‘`Site`’
- ‘`Undef`’

‘`Protein`’ Refers to a rule that generates complete annotation of a UniProtKB/Swiss-Prot entry (and also prevents other rules from being triggered independently = totipotent rule, unless listed as cross-reference/child c.f. "DR" line)

‘`Topo_domain / Domain / Repeat / Segment / Comp_bias / Site / Undef`’
Non protein ("totipotent") rules: could be triggered by selected associated prediction (when there are no totipotent rule matching the same target entry). Distinct types represent just meta-data that could be useful for rule checking and maintenance (e.g. association to controlled vocabularies). This also define rule application order / precedence (on target entry): same priority as in list (Protein > Topo_domain > ...). Topo_domain: for rules about topological domains (e.g. Extracellular domain). Domain: about "signature" domains (e.g. prosite PS50240) Repeat: about repeats (e.g. LRR Repeat). Segment: about ~short non signature domains (e.g. Transmembrane). Comp_bias: compositional bias (e.g. Coiled coils, Poly-Lys) Site: site (e.g. active site) Undef: anything else.

The optional trailing term ‘`auto`’ historically meant that a UniRule can be applied to automated annotation. If absent, it meant that the UniRule should be used only for the support of manual annotation. Now this flag is just used to define if a rule could be "shown" to the public, but only for "protein" rules! The other (non protein) rules are public by default = Now only the absence of "auto" in protein rules mean they are private, all other (committed) rules are public.

2.3 TR line

The *TR* (Trigger) line describes which (selected) sequence analysis features trigger the application of the curent UniRule. Each UniRule may contain one or more *TR* lines. Each feature name should appear only once in the TR line in the whole UniRule database, as one type of feature is expected to trigger only one rule.

Application:

```
Protein rule: mandatory
Domain/Site rule: mandatory
```

There are 2 types of TR lines:

2.3.1 *TR motif* line

Format:

```
TR   dbname; identifier1; identifier2; nbhits; level=level
```

Examples:

```
TR   PROSITE; PS50075; ACP_DOMAIN; 0-1; level=0
TR   HAMAP; MF_00401; -; 1; level=0
TR   General; Transmembrane; -; 1; level=0
TR   IPRO; IPR009003; -; 1; level=0
```

- The *dbname* is generally the name of a motif database, for example '`PROSITE`' or '`HAMAP`'. '`General`' is used for rules which refer to a specific feature type, independently of the prediction method used.
- The *identifier1* is the **feature name** *e.g.* the unique identifier of the motif in the given database, which is generally an accession number. For '`General`' methods a term indicates the feature identifier which triggers the rule, for example Transmembrane, Signal. If *dbname* is '`IPRO`', the *identifier1* is an InterPro number, and all motifs under that InterPro number will be triggers.
- The *identifier2* is a secondary identifier for the motif, which is generally an entry name, and which is empty ('`-`') in trigger lines of type '`HAMAP`' and '`General`'.
- The *nbhits* field indicates the expected number of hits to the motif. Obviously, the rule is only triggered if at least one hit is found. However, since motifs referred to in the TR line should not be repeated in the DR line, this field controls whether a FALSE_NEG cross-reference should be created if another TR line triggers the rule but no hit is found to the motif in the given TR line (See Section 3.5 [DR line], page 10 for a discussion). Legal values are '`1`' and '`0-1`', and the latter value is only valid if there are multiple TR lines.
- The *level* indicates the minimum level that a hit must have to trigger the rule. Note: obsolete, not used anymore!

2.3.2 *TR Metamotif* line

Format:

```
TR   Metamotif; mmsearch-options; metamotif
```

Examples:

```
TR   Metamotif; -; Signal>=GPI_anchor
```

Technical note: Now metamotifs are detected (through mmsearch) by anabelle psat; they are just 'classical' gff psat features. Therefore metamotifs detection is not tied anymore to the rule evaluation step. The metamotif TR should not be included in a condition to test if its required features are present, this is now useless/obsolete (as the metamotif was detected by psat, it implies that its features were present).

The *metamotif* field should correspond to the name of a detectable metamotif feature. The metamotif feature name is at the same time the metamotif description. It describes the arrangements of sub-features (by their name) in a sequence. The 'available' metamotifs are listed in *$ANABELLE/data/metamotifs.dat* This file is used by mmsearch, and is created at release (anabelle_update.sh) by extracting TR metamotif lines from CVSed UniRules.

The field *mmsearch-options* used to control the behavior of the metamotif search automaton (during 'picohamap' times?). It is now obsolete, just use '-'.

2.4 XX line

This separator line conveys no meaning. It is used to separate blocks of lines.

2.5 Names line

The *Names* line indicates the name(s) of the motif described by the rule. The first name is the one used in UniProtKB/Swiss-Prot, synonyms are listed below, one name per line. If there is no specific name for a motif, the term '`Undefined`' should be inserted as a placeholder.

Application:

```
Protein rule: 1 name is mandatory
Domain/Site rule: mandatory
```

Format:

```
Names: Undefined
Names: Synonym1
 Synonym2
 ...
```

Example:

```
Names: Nacht domain
 NACHT-NTPase domain
 Nucleoside triphosphatase domain
```

2.6 Function line

The *Function* line indicates the 'generic' function of a domain. Usually it should not be longer than one line, but in case it is, the lines following the first one will be indented by a space.

Application:

```
Protein rule: forbidden
Domain/Site rule: mandatory
```

Format:

```
Function: Undefined/Unknown
Function: text
```

Examples:

```
Function: Protein-binding
Function: DNA-binding
Function: Inhibits fibrinogen interaction with platelet receptors in
 snake venoms
```

2.7 Internal comments

Comments concerning the rule, which are for internal use, and do not appear in published versions. It should appear on lines prefixed with two asterisks and a space. They may appear at the very end of the Header section, and throughout the Annotation section and the Computing section. However, it is good practice to put such comments at the end of the Header section where they are immediately visible, and only put them exceptionally in the Annotation and Computing sections.

Example:

```
** SALRD unusually long in the C-terminus, flagged as atypical.
```

3 Annotation section

The *Annotation section* includes all UniProtKB/Swiss-Prot annotation lines that can be applied to a rule match. Additionally there can be lines which indicate condition statements (‘`case`’, ‘`else case`’, ‘`else`’, ‘`c?`’, ‘`c!`’) and a line which indicates the motif or alignments, according to which the feature positions are calculated (See Section 3.8.1 [FT From line], page 14). The line order is the same as in UniProtKB/Swiss-Prot.

3.1 ID line

The *ID* line contains the mnemonic code for a protein name used in the entry name of a UniProtKB/Swiss-Prot entry.

Application:

```
Protein rule: mandatory
Domain/Site rule: forbidden
```

Format:

```
ID   protein_name_code
```

Example:

```
ID   ACKA
```

Note: If specified code is ’Ynnn’ it will be replaced by ’Y’ followed by the OLN (OrderedLocusName) number of the annotated protein.

3.2 DE line

The *DE* line corresponds to the description line of a UniProtKB/Swiss-Prot entry. To define DE annotation that should be added to existing data (from original entry and/or from previous rule element applications) instead of replacing it: prepend DE block with ‘+’.

Application:

```
Protein: mandatory
Domain/Site: optional
```

Formats:

```
DE   Description.

DE   + partial_description
```

Example:

```
DE   RecName: Full=Putative 3-methyladenine DNA glycosylase;
DE            EC=3.2.2.-;

DE   + RecName: EC=2.7.3.-;
```

The DE line may contain the placeholder ‘`<locus_tag>`’, which is to be replaced (in annotation output) by the Ordered Locus Name (OLN) of the entry (or by its ORFNames if no OLN was

found; if nothing can be found: will be left as it is but a warning will be generated).
It may also contain the placeholder '`<gene_name>`' which is to be replaced by the entry gene name (GN Name).

Example:

```
DE   RecName: Full=Hemerythrin-like protein <locus_tag>.
```

The DE line may contain the `@`gn(ANYGENENAME) placeholder/command, which is to be replaced by the correct casing and gene count numbering (according to the entry taxonomy) of the specified gene name.

3.3 GN line

The *GN* line contains the common gene name (and optionally synonyms) of a protein family, when one exists.

Application:

```
Protein: optional
Domain/Site: forbidden
```

Format:

```
GN   Name=name;[ Synonyms=synonym[, synonym]...;]
```

Examples:

```
GN   Name=acpD;
GN   Name=groS; Synonyms=groES;
```

3.4 CC line

The *CC* line(s) contains all applicable comment lines of a UniProtKB/Swiss-Prot entry.

Application:

```
Protein: optional
Domain/Site: optional
```

Format:

```
CC   -!- topic: text.
```

Example:

```
CC   -!- SIMILARITY: Belongs to the ABC transporter family.
```

The CC line topic '`SIMILARITY: Contains`' is often not complete. The hash sign '`#`' in the text stands for the number of hits found for the given motif and is replaced during the automated annotation procedure; the term '`repeat`', '`domain`' or '`zinc finger`' will be set to plural if more than 1 true positive hit was found of them in a protein.

The CC line may contain :

- The placeholder '`<gene_name>`', which is to be replaced (in annotation output) by the entry gene name (GN Name).
- The #(ftdesc,text) placeholder *e.g.* #(manganese,ion) which is to be replaced by the number of distinct instances of the specified ft description element (from FT lines takes the highest number corresponding to the specified element; works only for numered ft desc (with '`#`'); (See Section 3.8 [FT line], page 13) plus if specified, the '`text`' pluralized if more than 1 instances were found. *e.g.* 2 manganese ions. If the text (and the ,) is ommited, the ftdesc itself will be pluralized if needed.
- The `@`gn(ANYGENENAME) placeholder/command, which is to be prelaced by the correct casing and gene count numbering (according to the entry taxonomy) of the specified gene name.
- The #{[Aaa-]POS[:template_name]} placeholder will be replaced by the position mapped - from the position *POS* in the match region or, in protein rule mode, the defined entry template - to the correct numbering in the target sequence, prefixed with the matched amino acid (3 letters code). If *Aaa* amino-acid is specified (3 letters code), a warning will be generated if the mapped residue does not correspond to the specified one. If a *template_name* is defined, the corresponding sequence template will be used for the mapping (needed for disambiguation when there are several entry templates! *e.g.* for a HAMAP alignment with several defined templates)

```
CC   -!- PTM: The phosphorylation of #{120} activates ...
CC   -!- PTM: Phosphorylation at #{Ser-187:FEN1_HUMAN} by CDK2 ...
CC   -!- PTM: The reversible ADP-ribosylation of #{Arg-101} inactivates ...
```

The latter becomes for example:

```
CC   -!- PTM: The reversible ADP-ribosylation of Arg-112 inactivates ...
...
```

3.5 DR line

The *DR* line main usage is to trigger 'child' rules in order to avoid duplication of rule content. It has not much to do with 'real' DR UniProtKB/Swiss-Prot annotation lines (in Anabelle, DR annotation is only performed for PROSITE motifs). The format of the DR line is similar to that of a TR line (See Section 2.3 [TR line], page 4).

Application:

`Protein rule`*: optional*

`Domain/Site rule`*: optional*

Format:

DR   *type/dbname*; *feature name*; *identifier*; *nbhits*; trigger=[any|yes|strict|no]

Example:

```
DR   PROSITE; PS00419; PHOTOSYSTEM_I_PSAAB; 1; trigger=no
DR   PROSITE; PS00010; ASX_HYDROXYL; 0-1; trigger=any
DR   General; Signal; -; 0-1; trigger=strict
DR   General; Transmembrane; -; 10-11; trigger=yes
```

- The *type/dbname* field indicates the database name - for example 'PROSITE' or 'HAMAP' - or 'General' (for non database features *e.g.* Signal) or ADD_TOPO_DOMAIN (see below).

- The **feature name** field indicates the name of the feature that should be used (*e.g.* Transmembrane, motif accession number, etc...) to propagate annotation through the triggering of its associated rule.
- The *identifier* is a secondary identifier (ID) for the motif which is empty (‘`-`’) in trigger lines of type ‘`HAMAP`’ and ‘`General`’ (for information purposes, not checked/used).
- The *nbhits* field indicates whether the "child feature" is mandatory, and the number of (’hits’) features which are expected. The value of this field may be a number (‘`1`’), or a range (‘`0-1`’). A range in the form ‘`*number*-unlimited`’ indicates that the number of matches is unlimited.

  If the number of expected features does not match the number of detected ones a warning (**HW SAM line) will appear in the generated annotation.

  If a PROSITE cross-reference is found in a mandatory (*i.e.* where *nbhits* does not contain zero) TR or DR line but no match is found, a DR line will be created which includes the status qualifier ‘`FALSE_NEG`’. In rules of the data class ‘`Protein`’ it is further possible to include cross-reference lines with the status qualifier ‘`FALSE_POS`’ for matches to which a rule does not refer.

  If the *nbhits* field of the DR line is ‘`0-1`’, the feature applies to the domain only in some proteins (it won’t ’complain’ if none is found). In that case for PROSITE motifs, DR FALSE_NEG will not be created if no match is found.
- The *trigger*, must be set to either of ‘`yes`’, ‘`any`’, ‘`strict`’ or ‘`no`’. The values ‘`yes`’, ‘`any`’ or ‘`strict`’ indicates that the rule associated with the specified features should be triggered to generate annotation. With ’strict’, only selected features (by the analysis result automatic selection module) will be used. With ’any’, all specified features will be used. With ’yes’, selected features will be used preferentially, but if none is found, all (non-selected) features (of the specified type) will be used!

  The value ‘`no`’ indicates that features (overlapping, if parent rule is not a protein type rule, all, if parent rule is of protein type) with the specified feature name will not be used to generate annotation.

Notes:

- To specify TOPO_DOMs (by triggering topological domain associated rules) use DR ADD_TOPO_DOMAIN

  Examples:

  ```
  DR   ADD_TOPO_DOMAIN; Cytoplasmic; -; 2; trigger=yes
  DR   ADD_TOPO_DOMAIN; Periplasmic; -; 1; trigger=yes
  ```

  The first line will set the Nter TOPO_DOM to ’Cytoplasmic’, the second line specifies the alternative compartment ’Periplasmic’ to propagate alternate Cytoplasmic / Peroplasmic TOPO_DOMs between (selected) Transmembranes. N.B. the ’fake’ ADD_TOPO_DOMAIN type/dbname is a flag for the sequence analysis module (that peaks inside the rule) to tell it to create corresponding topological domain sequence analysis "results" between selected transmembrane.
- DR lines can be used to trigger rules associated with any kind of feature, not just motif/profile matches (see examples). For example it can be used to trigger Signal, Transmembrane , topological domains (see above) etc...
- DR lines can be inside cases...
- the used features will not generate annotation by themself (hence a second time) after they are ’used’ through those rule DR lines.
- A protein type rule will prevent all lower (*e.g.* domain) rules from independently being triggered by their associated feature. Therefore "under" a protein rule: the only non

protein rules that will be used are the ones associated to features specified in those DR (with trigger=yes|strict) lines.

- The DR line could also be used, with trigger=no, to prevent the generation of specific child annotation. It will generate warnings if the number of expected features does not match the number of detected ones. If the feature name is a PROSITE motif: It will also set the status of DR PROSITE lines (see above "The nbhits field").
- Through a protein type rule, the DR directive will consider could rules associated with any feature detected within the same protein. Through a non protein type rule, only the features overlapping with the 'current' rule associated features will be considered.
- **Considered features (the ones used to trigger annotation) will prevent the subsequent triggering / usage of any other overlapping features** specified in DR lines, except for the ADD_TOPO_DOMAIN. **Features are examined following the order in the DR lines**. In the first example, the Signal (if present) will be used first and so prevent the usage of any overlapping transmembrane. Transmembranes that do not overlap with the used signal will then be used after.
  For topological domains (via ADD_TOPO_DOMAIN): place them first in the DRs (so that they are not blocked by a previously triggered overlapping feature), they won't block the triggering of other features.

3.6 KW line

The *KW* line contains all applicable keywords for a UniProtKB/Swiss-Prot entry, one per line.

Application:

```
Protein rule: optional
Domain/Site rule: optional
```

Format:

```
KW   keyword
...
```

Example:

```
KW   Transferase
KW   Kinase
```

3.7 GO line

The *GO* line contains all applicable Gene Ontology terms, one per line.

Application:

```
Protein rule: optional
Domain/Site rule: optional
```

Format:

```
GO   accession-number; aspect:term
...
```

Example:

```
GO   GO:0019104; F:DNA N-glycosylase activity
GO   GO:0006281; P:DNA repair
```

3.8 FT line

The *FT* (Feature table) line contains applicable features for a UniProtKB/Swiss-Prot entry. The feature positions are calculated by the automated annotation program based on the rule and the motif match positions.

Application:

```
Protein rule: optional
Domain/Site rule: optional
```

Format:

```
FT   From: template-id (template-accnumber) OR unique identifier for a motif
FT   key      from     to       desc.
[FT   [Optional;] [Group: n;] [Condition: pattern]]
```

Examples:

```
FT   From: CARB_ECOLI (P00968)
FT   DOMAIN     Nter    403      Carboxyphosphate synthetic domain.

FT   From: PS00385
FT   SITE          6      6      By similarity.
```

3.8.1 *FT From* line

The *FT From* line is mandatory above the *FT feature* line. It defines the "template" / "frame of reference" to allow the rule specified coordinates to be mapped into the target sequence coordinates, for all subsequent *FT feature* lines. The template must be one of the following: 'any', the unique identifier for a motif, a protein identifier, or a metamotif.

Format:

```
FT   From: unique identifier for a motif [:regexp]
FT   From: entry_name (accession_number) [:regexp]
FT   From: metamotif [:regexp]
FT   From: any [:regexp]
```

Examples:

```
FT   From: PS50234
FT   From: ACP_ECOLI (P02901)
FT   From: PS50021=7,91=PS50021
FT   From: any
FT   From: PS50217 :([KR])[^LI].*([KR])[^LI]([LI]).*([LI])
```

If (accession_number) is given = the template/frame of reference is a UniProt protein = the numbering is based on this protein, the subsequent positions will be mapped to the annotated target sequence.
If (accession_number) is not given = the frame of reference is the trigger (motif) match region.
If a specific motif identifier is given the subsequent FTs will only be propagated if the current trigger is of the specified kind (this is useful when one rule as multiple triggers...). If "any" is used, no filtering by trigger name will be performed.
The optional regexp, can be used in conjonction with $1-9 position "keys" in subsequent FTs to annotate floating postions.

N.B. If the template is a metamotif, it must be present verbatim in the TR line.

3.8.2 *FT feature* line

The *FT feature* line defines the actual FT lines that are to be propagated in member entries.

Format:

```
FT   key       from      to        desc.
```

key A UniProtKB/Swiss-Prot feature key. (N.B. if set to '`HIDDEN`', the feature will not be propagated. Just to add constraint to a FT Group or raise a Tag...)

from
to The feature positions in one of the following formats:

- '`1`' The start of the *template*.
- '`12`' A position relative to the beginning of the *template*.
- '`8+1`' A position relative to the beginning of the *template*, shifted by a number of residues relative to the target sequence (not always the same as position 9 on the template sequence, as there might be inserts).
- '`<1`'
  '`?251`' A position relative to the beginning of the *template*, with a modifier used in UniProtKB/Swiss-Prot features.
- '`Nter`'
  '`Cter`' The first, respectively last residue of the target sequence.
- '`from`'
  '`to`' The start and the end of the *template*, extended by as many residues as necessary to produce a complete match in case of partial profile matches. The number of residues to add is indicated in the '`FeatureFrom`' and '`FeatureTo`' GFF attributes.
- '`entry`'
  '`exit`' The start and the end of the *template*, not extended in case of partial matches.
- '`$1`' The start of the matching FT From regexp capture group (1 to 9)

desc The feature description.

Examples:

```
FT   CHAIN      to+1   Cter       <name>.
FT   LIPID         1      1       GPI-anchor amidated <residue_name>.
FT   DOMAIN     from     to       Laminin G-like #.
FT   TOPO_DOM   Nter      6       Periplasmic (Potential).
FT   DOMAIN        ?    8+1       EGF-like #.
FT   From: PS50217 :([KR])[^LI].*([KR])[^LI]([LI]).*([LI])
FT   REGION       $1      $2       Basic motif #.
FT   REGION       $3      $4       Leucine-zipper #.
```

Placeholders (in the description)

- The '`<name>`' placeholder will be replaced (in the annotation output) by the protein Rec-

Name (Full) (DE line 1st elem).

- The '<gene_name>' placeholder will be replaced by the entry gene name (GN Name).
- The '@gn(ANYGENENAME)' placeholder/command, will be replaced by the correct casing and gene count numbering (according to the entry taxonomy) of the specified gene name.

```
FT   REGION      101    124        Necessary for interaction with @gn(ABC-1).
```

- The '<residue_name>' placeholder will be replaced by the chemical name of the sequence residue on which the feature resides. Thus the following line in a UniRule:

```
FT   LIPID         1      1       GPI-anchor amidated <residue_name>.
```

may be transformed by samann.pl into the following UniProtKB/Swiss-Prot line:

```
FT   LIPID       300    300       GPI-anchor amidated aspartate.
```

- The #{[Aaa-]POS} placeholder will be replaced by the position mapped - from the position *POS* in the match region or, in protein rule mode, the defined current entry template (c.f. "FT From") - to the correct numbering in the annotated sequence, prefixed with the matched amino acid (one letter code). If an Aaa amino-acid is specified (3 letters code), a warning will be generated if the mapped residue does not correspond to the specified one.

```
FT   CROSSLNK    238    264      Tryptophyl-tyrosyl-methioninium (Tyr-Met)
FT                               (with #{Trp-90}) (By similarity).
```

The latter will generate for example:

```
FT   CROSSLNK    240    266      Tryptophyl-tyrosyl-methioninium (Tyr-Met)
FT                               (with W-92) (By similarity).
```

- for FT ACT_SITE description: text between square brackets [ ] will be shown only for entry that have more than one enzymatic activity (based on the number of '(EC num' fields found in the entry DE line). If the entry has 1 or no defined enzymatic activity, the whole text (and the brackets) won't be annotated.

```
FT   ACT_SITE      6      6       [For protease activity] By similarity.
```

may be transformed by samann.pl into the following UniProtKB/Swiss-Prot line: If the entry has several EC:

```
FT   ACT_SITE    506    506       For protease activity (By similarity).
```

Note that if the text is added, the qualifier (By similarity), is put between brackets (as it is not alone).
If the entry has only 1 enzymatic activity:

```
FT   ACT_SITE    506    506       By similarity.
```

- The '#' placeholder may be used to assign automatic numbers in the *desc* field. If the feature occurs over once with the same text before the occurrence of the placeholder, the placeholder is replaced with subsequent numbers.
- The '#*n*' placeholder, where *n* is a number starting from 1, should be used when it is needed to reference distinct ligands with the same name. For example, the following made-up rule:

```
FT   DOMAIN     from     to       Foobar #.
FT   METAL        87     87       Iron #1 (By similarity).
FT   METAL       118    118       Iron #1 (By similarity).
FT   METAL       118    118       Iron #2 (By similarity).
FT   METAL       180    180       Iron #2 (By similarity).
```

Would yield the following annotation in a protein with a single match:

```
FT   DOMAIN        1    200       Foobar.
FT   METAL        87     87       Iron 1 (By similarity).
FT   METAL       118    118       Iron 1 (By similarity).
FT   METAL       118    118       Iron 2 (By similarity).
FT   METAL       180    180       Iron 2 (By similarity).
```

And the following annotation in a protein with two matches:

```
FT   DOMAIN        1    200       Foobar 1.
FT   DOMAIN      201    400       Foobar 2.
FT   METAL        87     87       Iron 1 (By similarity).
FT   METAL       118    118       Iron 1 (By similarity).
FT   METAL       118    118       Iron 2 (By similarity).
FT   METAL       180    180       Iron 2 (By similarity).
FT   METAL       287    287       Iron 3 (By similarity).
FT   METAL       318    318       Iron 3 (By similarity).
FT   METAL       318    318       Iron 4 (By similarity).
FT   METAL       380    380       Iron 4 (By similarity).
```

3.8.3 *FT constraints* line

The *FT constraints* line (also known as the *FT Condition* line) gives constraints on the FT line immediately above it.

Format:

```
FT   [Tag: tagname[, tagname]...;] [Optional;] [Group: n;] [Condition: pattern]
```

The Feature line is most often constrained by a pattern on the sequence (in cases where more complex rules are needed, the case statement (See Section 6.1 [Case statement], page 28) should be used.

Example:

```
FT   BINDING      37     37       Phosphopantetheine (By similarity).
FT   Tag: phospho; Condition: S
```

The ‘`pattern`’ is specified in the PROSITE pattern format with the addition

that the character ‘`*`’ may be used to specify an unconstrained range, *e.g.* ‘`C-x*-C`’. The region of the sequence corresponding to the feature must exactly match this pattern.

For the consistency of annotation, multiple FT lines that should be applied either all together or not at all should be grouped within a ‘`Group`’, to constrain the common presence of all sites. This group can the be referenced by *case* statements, for instance in the relevant KW and CC lines that depend on the presence of the feature.

Example:

```
case <FTGroup:1>
KW   GTP-binding
end case
XX
case <OC:Bacteria>
FT   From: IF2_ECOLI (P02995)
FT   DOMAIN      392    540       G-domain.
FT   Group: 1
FT   NP_BIND     398    405       GTP (By similarity).
FT   Group: 1; Condition: G-H-V-D-H-G-K-T
FT   NP_BIND     444    448       GTP (By similarity).
FT   Group: 1; Condition: D-T-P-G-H
FT   NP_BIND     498    501       GTP (By similarity).
FT   Group: 1; Condition: N-K-[LIVCM]-D
end case
```

Note: One FT line can be part of several FTGroups. If at least one of those groups is complete, the FT line passes its FTGroup constraint (implicit OR).

Example:

```
FT   DISULFID     25     31       By similarity.
FT   Group: 1; Group: 2; Condition: C-x*-C
```

The ‘`Optional`’ label can be used to indicate that the absence of a feature should not be considered a trigger for warnings in annotation programs. It can only be present if a ‘`Condition`’ pattern is supplied.

Example:

```
FT   BINDING      37     37       Phosphopantetheine (By similarity).
FT   Optional; Condition: S
```

A ‘`Tag`’ name (or several) can be given to the feature (N.B. distinct features can have the same tag name) for easier use of case against specific features (with case <FTTag:tagname>) See Section 6.1 [Case statement], page 28.

Example:

```
FT   DISULFID     48     51       Redox-active (By similarity).
FT   Tag: disulf, redox; Condition: C-x*-C
```

4 Computing section

The *Computing section* contrasts with the *Annotation section* in that the line identifiers are no longer restricted to 2 letters. The information starts in the line of the line identifier, the following lines are indented by 1 space.

General format:

```
Line identifier1: info in line1
 following lines are indented
Line identifier2: info in line1
 following lines are indented
 ...
```

Not all line types are relevant to any data class (See Section 2.2 [DC line], page 2).

4.1 Warn line

Specify a warning that should be generated when the rule is used for automatic annotation. Most often used within a case statement to indicate the occurrence of an inconsistency that cannot be solved by the rule, or that some annotation should be completed manually by a curator. The SAM module transfers the text of Warn lines into the '`**HW`' section of a UniProtKB/Swiss-Prot entry.

Application:

```
Protein rule: optional
Domain/Site rule: optional
```

Format:

```
Warn: text
```

Example:

```
case <OC:Proteobacteria>
Warn: Check manually domain bounds
end case
```

4.2 Chop line

Range by which the bounds of a domain may be chopped in order to annotate successive domains in an exactly consecutive manner. This line can only be used by programs if the complete size of the domain can be annotated; generally it will not be possible to use it with a pattern that covers only part of the domain.

Application:

```
Protein rule: forbidden
Domain/Site rule: mandatory
```

Format:

```
Chop: Nter=max; Cter=max;[ Xter(motif)=max;]*
```

Examples:

```
Chop: Nter=0; Cter=3;
Chop: Nter=1; Cter=unlimited;
Chop: Nter=0; Cter=0; Nter(Signal)=50;
```

- *max* indicates the maximum number of positions, by which the N- terminal or C-terminal may be trimmed to be able to annotate adjacent identical (by name) domains which overlap slightly. *max* can be either of 0 (by default), a positive value, or the word '`unlimited`'.
- '`Nter`' and '`Cter`' indicate the number of positions that may be trimmed when the triggering motif is found adjacent to any other identical (by name) motif. Additionally, specific adjacent motifs for which trimming is also allowed can be indicated by giving the motif name in braces, for example '`Nter(Signal)`'.

4.3 Size line

The *Size* line indicates the size relevant to a protein family or motif. For entries of the data class '`Protein`', the minimal and maximal size of proteins matching the rule are listed. For entries of the data class '`Domain`', this line contains the size range of the complete domains annotated in UniProtKB/Swiss-Prot. Members that are strongly divergent in size may be excluded from the range. A size may be specified as '`unlimited`'.

Application:

```
Protein rule: mandatory
Domain/Site rule: mandatory
```

Format:

```
Size: minimal_size-maximal_size;
Size: fixed_size;
```

Examples:

```
Size: 176-239;
Size: 13-unlimited;
Size: unlimited;
```

4.4 Related line

Lists UniRules that are known to be similar in sequence, and risk to produce cross-matches. If the string '`!`' or '`!!`' is appended to the name of a rule, it means that the rule listed in the Related line supersedes the current rule, *i.e.* matches to the current rule should be disregarded if a match with the listed rule is found: '`!`' in an overlapping region; '`!!`' anywhere on the protein.

The marker '`!`' is particularly useful when two different rules exist for a 'short' and a 'long' version of the same protein (as occurs sometimes in HAMAP families). 'Long' proteins will match both profiles; under these circumstances the 'longer' UniRule should contain the '`!`' marker to supersede the shorter UniRule.

Application:

```
Protein rule: mandatory
Domain/Site rule: mandatory
```

Format:

```
Related: None;
Related: Protein[!][!];[ Protein[!][!];]...
```

Example:

```
Related: MF_00492; MF_00493; MF_00494;
Related: MF_00344!;
Related: ANA00003!!;
```

4.5 Repeats line

The observed number of repetitions of a domain or site in a UniProtKB/Swiss-Prot entry. The number may be specified as '`unlimited`'.

Application:

```
Protein rule: forbidden
Domain/Site rule: mandatory
```

Format:

```
Repeats: value;[ no keyword;]
Repeats: min-max;
```

The optional attribute '`no keyword`' indicates that the presence of several copies of a rule of the type '`Domain`' should not trigger the addition of the keyword '`Repeat`' (See Section 7.1 [The keyword Repeat], page 32).

Examples:

```
Repeats: 1;
Repeats: 2-4;
Repeats: unlimited; no keyword;
```

4.6 Topology line

Specifies the subcellular location(s) in which a domain or site may occur.

Application:

```
Protein rule: optional
Domain/Site rule: mandatory
```

Formats:

```
Topology: Undefined;
Topology: location;
```

Values for this topic are restricted to '`Undefined`', '`Cytoplasmic`' or '`Not cytoplasmic`'.

Example:

```
Topology: Not cytoplasmic;
```

4.7 Template line

Lists accession number(s) of characterized proteins which were used to build the UniRule (Note: indicative only). Uncharacterized protein families do not necessarily have a template, this is noted as ‘`Template: None;`’. Note that in many cases the propagated annotation is a subset of that found in the characterized entries.

Application:

`Protein rule`*: mandatory*
`Domain/Site rule`*: forbidden*

Format:

`Template:` *accession_number*`;[` *accession number*`;]...`
`Template: None;`
`Template: Undefined;`

Examples:

```
Template: P12345;
Template: None;
Template: Undefined;
```

4.8 Example line

One or more example entry targeted by the rule.

Application:

`Protein rule`*: forbidden*
`Domain/Site rule`*: mandatory*

Format:

`Example:` *accession_number*`;[` *accession number*`;]...`
`Example: Undefined;`

Examples:

```
Example: P12345;
Example: Undefined;
```

4.9 Scope block

Lists the taxonomic classes in which a rule match may be found.

Application:

```
Protein rule: mandatory
Domain/Site rule: mandatory
```

Format:

```
Scope:
 kingdom[; sub-taxon]
  [except sub-taxon
  ...]
  [not in taxcode[, taxcode]...]
 ...
```

The *kingdom* line is indented by one space, while the subsequent lines are indented by two spaces.

Example:

```
Scope:
 Bacteria; Proteobacteria
  except Enterobacteriales
  except Pasteurellales
 Bacteria; Actinobacteria
 Archaea
  not in ARCFU, HALN1, METTH, METJA, PYRAB, PYRHO, SULSO, SULTO,
  THEAC, THEVO
 Plastid
```

The taxonomic classification is composed of the kingdom, optionally followed by the name of a sub-taxon, to further limit the application of the UniRule to any taxonomic level. Valid values for *kingdom* are: '`Eukaryota`', '`Bacteria`', '`Archaea`', '`Viruses`', '`Bacteriophage`', '`Plastid`' and '`Mitochondrion`'. The two latter values designate proteins encoded in the organelle genome but not proteins encoded in the nucleus and targeted to the organelle.

If it has been assessed with certainty that a UniRule is not represented in:

- A taxonomic group, its name may be indicated in the '`except`' field.
- A complete proteome, its taxcode may be indicated in the '`not in`' field.

Note: Plasmid is not defined as *kingdom*; there is a separate line type (See Section 4.12 [Plasmid line], page 25).

4.10 Fusion block

Lists UniRules to which a given UniRule may be fused in some instances.

Application:

```
Protein rule: mandatory
Domain/Site rule: forbidden
```

Format:

```
Fusion:
 NT: None
 CT: None
Fusion:
 NT: Protein[; Protein]...
 CT: Protein[; Protein]...
```

Protein may be either a UniRule accession followed by an identifiers between round brackets (*e.g.* '`MF_00222 (aroE)`'), or a designation between angle brackets (*e.g.* '`<Thioredoxin domain>`') if no UniRule is available.

Example:

```
Fusion:
 NT: None
 CT: MF_00222 (aroE); <Unknown>
```

4.11 Duplicate line

Lists the organisms for which the motif which triggers the rule is found in multiple copies.

Application:

```
Protein rule: mandatory
Domain/Site rule: forbidden
```

Format:

```
Duplicate: None
Duplicate: in taxcode[, taxcode]...
```

Example:

```
Duplicate: in ANASP, CAUCR, LACLA, RHILO, RHIME, STAAU, SYNY3
```

4.12 Plasmid line

Lists the organisms for which the motif which triggers the rule is found encoded on a plasmid.

Application:

```
Protein rule: mandatory
Domain/Site rule: forbidden
```

Format:

```
Plasmid: None
Plasmid: in taxcode[, taxcode]...
```

Example:

```
Plasmid: in RHIME
```

4.13 Comments block

Comments concerning the rule, which should be made visible to the public.

Application:

```
Protein rule: mandatory
Domain/Site rule: mandatory
```

Format:

```
Comments: None
Comments: comment_text
```

Example:

```
Comments: NUDIX-like C-terminal domain in SYNY3
```

5 History section

Whenever curators commit a UniRule file, they are asked for a short description of the modifications which have been made. This description as well as the revision number, date and author of the modification are added automatically by CVS into the file, whenever the special keyword '`$Log$`' is encountered.

Example:

```
# $Log: PRU00001.dat,v $
# Revision 1.2  2002/09/09 16:41:42  boeckman
# format changes
#
# Revision 1.1  2002/08/20 12:16:52  gattiker
# Created
#
```

The last (uppermost) Revision line indicates the current revision number and revision date of the UniRule. The revision number is composed of two integers connected by a period. The first integer is the version of the UniRule format specification used (Currently always 1). The second integer is the version number of the rule, which normally starts from 1 and is increased each time a modification to the rule is committed. The value 0 for the version number is used for rules created before UniRule version control was implemented (*i.e.* old HAMAP rules).

If a major modification which breaks format compatibility is ever made to the UniRule specification (*i.e.* this document), the first integer of the revision number will be increased. Thus a rule which used to be '`Revision 1.4`' will become '`Revision 2.5`'.

The entire content of the History section consists of lines prefixed by a hash sign ('`#`'). It is managed entirely by CVS, and it is strictly forbidden to alter its contents manually or by programs, even to correct typos.

Only the last Revision line appears in published version.

6 Control statements

6.1 Case statement

Format:

```
case <condition>[ and|or [not] [defined] <condition>]...
else case <condition>[ and|or [not] [defined] <condition>]...
else
end case
```

The ‘case’ and ‘else case’ lines include conditions that must be met for the lines below it to be applied, until the next ‘else case’, ‘else’ or ‘end case’ statement. Condition lines (c! and c?, see below) do not break the latest case statement.

Note: It is not possible to use a ‘case’ statement within a ‘case’ statement, but it is possible to use the condition lines c! or c?.

Types of cases:

- **OS/OC/OG**: on taxonomy and organelle (OS, OC and OG lines):

  ```
  case <OG:Chloroplast> or <OC:Cyanobacteria>
  case not <OG:Chloroplast> and not <OG:Cyanelle>
  case <OC:Archaea>
  case <OC:Bacteria>
  case <OS:Staphylococcus aureus>
  ```

  Note for conditions on organism names (‘case <OS:*taxon*>’): the organism name matches also subspecies, *i.e.* organisms with the same name followed by a space and then any text. For example, ‘Staphylococcus aureus’ matches ‘Staphylococcus aureus RF122’, but ‘Salmonella typhi’ does not match ‘Salmonella typhimurium’.

- **FT**: on a feature (FT block) propagated from the rule (a pattern that must be matched by the feature can also be specified):

  ```
  case <FT:1>
  case <FT:1=A-x-x>
  ```

  Note: In the FT lines, this condition can only be used if the targeted FT is itself not in a FT[Group] case!

  Note: The targeted feature number corresponds to its position/order in the FT block. Caution: FT lines in a FT[Group] case are also numbered according to their relative position in the FT lines, but the numbering starts at 1 + the number of FT element - not in a FT[Group] case (it is therefore simpler to put those FT elements at the bottom of FT lines)...

- **FTGroup**: on the fact that a feature group was propagated: (<*FTGroup:n*> See Section 3.8.3 [FT constraints line], page 18)

  ```
  case <FTGroup:1>
       ↦ true if all features in Group 1 were be propagated
  ```

  Note: In the FT lines, this condition can only be used if the targeted FTGroup is itself not in a FT[Group] case!

- **FTTag**: on the fact that a feature with a certain tag was propagated: (*Tag: tagname*. See Section 3.8.3 [FT constraints line], page 18)

  ```
  case <FTTag:phospho>
       ↦ true if at least one feature with the ‘phosho’ tag was propagated
  ```

- **(Any)Feature**: on the presence of a feature in the GFF file:

  ```
  case <Feature:PS50084>
  case <AnyFeature:PS50084>
  case <Feature:Transmembrane>2>
       ↦ feature must be present over two times
  ```

  The operators '>', '<', '==', '>=', '<=' are supported.

  The difference between '`Feature`' and '`AnyFeature`' is as follows. '`Feature`' only refers to the (rule) triggering feature + features that overlap (by at least 50%) with it. '`AnyFeature`' refers to all features matching the sequence.

  Those conditions refer to both selected and unselected features, except if an operator (>,<,==,>=,<=) is present: here only selected features will be examined. For example to test that there is no selected Signal_anchor feature (somewhere in the sequence), use:

  ```
  case <AnyFeature:Signal_anchor<1>
  ↦ selected feature must be absent
  ```

  Whereas without the operator, something like:

  ```
  case not <AnyFeature:Signal_anchor>
  ↦ feature must be absent (whether selected or not)
  ```

- on a pattern being matched at a certain position of a GFF feature:

  ```
  case <Feature:PS50084:5=E>
  case <Feature:PS99999:10-13=N-{P}-[ST]-{P}>
  ```

- on the rule having been triggered by another rule:

  ```
  case <Triggered>
  ```

- on a metabolic pathway or property (only '=' operator can be used):

  ```
  case <Property:Membrane=2>
  case <Property:NITROGEN_FIXATION>
  case <Property:NODULATION>
  case not <Property:METHANOG>
  ```

  Note (HAMAP 'hack'): If Property:Membrane is unknown (and if organism corresponding proteome is complete) it will be considered equal to 1 (so case <Property:Membrane=1> and case <Property:Membrane> will be true).

- on the presence of a (motif) feature with a certain InterPro id

  ```
  case <IPRO:IPR000859>
  ```

- on protein size

  ```
  case <Length<256>
  case <Length>420>
  ```

  Note: ProRule hack! to 'play' with domain coverage...

6.1.1 Ternary logic

UniRule conditions should be evaluated using ternary logic, where conditions evaluate to one of three values: *true*, *false*, or *undef*. Operators are defined as follows, consistently with their implementation in the Perl programming language. Note that certain rules are counterintuitive.

Binary operators: '`and`' and '`or`'

i	**j**	**i and j**	**i or j**

true	*true*	true	true
true	*false*	false	true
true	*undef*	undef	true
false	*true*	false	true
false	*false*	false	false
false	*undef*	false	undef
undef	*true*	undef	true
undef	*false*	undef	false
undef	*undef*	undef	undef

Unary operators: '`not`' and '`defined`'

i	**not i**	**defined i**
true	false	true
false	true	true
undef	undef	false

Operator associativity and precedence

The precedence order from highest to lowest and associativity are as follows.

associativity	**operator**
right	defined
right	not
left	and
left	or

Example of application: if the number of membranes is known and equal to 2, then apply a given annotation item. Otherwise, apply a less specific annotation item.

```
case defined <Property:Membrane> and <Property:Membrane=2>
CC   -!- SUBCELLULAR LOCATION: Inner membrane-associated (By similarity).
else
CC   -!- SUBCELLULAR LOCATION: Membrane-associated (By similarity).
end case
```

6.1.2 Condition line *c!* or *c?*

The condition line *c!* or *c?* contains additional constraints for the propagation of the line immediately below it. The format of this line is either:

```
c!   condition
c?   condition
```

where *condition* has the same syntax as in a *case* line, or, before a FT line, it can also include a PROSITE pattern expression.

Note: In FT lines, FT[Group] condition cannot be used (use case instead!).

The condition line differs from the *case* line in that

- the constraint affects only the next line (only 1 line!);
- it does not break a previous *case* condition.

The condition of the *c!* line must be true, otherwise an error is expected. Tools using UniRules are recommended to produce an error message.

Example:

```
c!   <Feature:PS00013>
KW   ATP-binding
```

The condition of the *c?* line may be true or not, as the feature does not appear in all matches of the UniRule.

Example:

```
c?   <Feature:PS99999:10-13=N-{P}-[ST]-{P}> and <OC:Eukaryota>
FT   CARBOHYD      10    13       N-linked (Potential).
```

Exceptions: See Chapter 7 [Hidden information], page 32.

Transition: Condition lines should get automatically replaced by *c!* lines, and some of them later by *c?* lines. Mandatory conditions for disulfides should be suppressed, optional ones replaced by *c?* lines.

7 Hidden information

UniRules strive to include all information relevant to a motif. However, to avoid repetition, we did not include the information below, which is implicitly ‘known’ by the automatic annotation pipeline tools.

7.1 The keyword *Repeat*

The keyword *Repeat* is relevant to all rules of the data class *Domain*. This keyword applies when a domain or repeat is found at least twice in a protein. The corresponding part of the rule would be:

```
case <Feature:current_rule_accession_number>1>
KW   Repeat
end case
```

This behavior can be prevented by using the attribute ‘`no keyword`’ in the Repeats line (See Section 4.5 [Repeats line], page 22).

7.2 The *FT constraints* line for the feature key *DISULFID*

For features with the key *DISULFID*, the constraint that the *From* and *To* positions both need to be a cysteine is implicit. The corresponding line would be the second line of the following example:

```
FT   DISULFID      4    23      By similarity.
FT   Condition: C-x*-C
```

Appendix A UniAln

Introduction

UniAln is a format for protein sequence alignments that complement the UniRules collection. Some UniRules are developed based on predictors from specialized databases such as PROSITE. However, other UniRules are based on curated alignments that constitute the UniAln collection. That is the method used in the HAMAP annotation project.

Format

UniAln alignments are in a format similar to that produced by the CLUSTAL suite of programs. Each alignment is composed of:

- A *Header line* starting with the string '`CLUSTAL`' or '`MUSCLE`' or '`T_COFFEE`'. The rest of the line is free-text, but special strings are recognized by programs (see below).
- Two *Blank lines*.
- Alignment blocks, each one followed by a consensus line, separated by blank lines.

The alignment is subject to the following constraints:

- The file may not contain any trailing spaces at the end of lines. This allows safe editing with text editors that may suppress such spaces.
- The sequences in the alignment may only contain uppercase letters (with the exception of the letters '`O`', '`U`', '`J`', '`B`', '`Z`'), the gap character '`-`', and the special characters '`<`' and '`>`' to indicate that a sequence portion has been excised to the NH2-side of the following amino acid, or to the COOH-side of the preceding amino acid, respectively. *Note: although the letters '`B`' and '`Z`' are allowed by CLUSTAL, they are reserved internally to escape the special characters '`<`' and '`>`' when the alignment is converted through the CLUSTAL program.*
- The line wrapping and content of the consensus lines must be such that the alignment may pass through the command '`clustalw -convert`' without modification (after escaping the characters '`<`' and '`>`', and not taking into account trailing spaces at the end of consensus lines created by '`clustalw -convert`').
- Sequence identifiers must be valid identifiers (ID line) from Swiss-Prot. The sequences must correspond to the sequences in Swiss-Prot. However, portions of sequences that are not aligned may be clipped by curators, and replaced with the special character '`<`' (for N-terminal clipping) or '`>`' (for C-terminal clipping), or by the succession '`><`' (for internal clipping).

The alignment header line

The first line of the alignment must start with the string '`CLUSTAL`' or '`MUSCLE`' or '`T_COFFEE`'. The rest of the line is free-text, but special tags are recognized by programs. The tags may be repeated. Tags are:

'`template=`*`identifier`*'
: indicate that a sequence in the alignment is a template for feature propagation in the UniRule that uses the alignment. It is mandatory to indicate template sequences in alignments to allow alignment-based feature propagation in UniRules.

'`profile_method=`*`method`*'
: indicate the method that should be used to generate a profile from the alignment. Allowed values for *method* are:

'`profile_method=pfmake`'
: (default) A profile should be generated using '`pfw`' and '`pfmake`' from the PFTOOLS package. There is no need to indicate this method, as it is the default.

'`profile_method=hmmbuild`'
: A Hidden Markov Model should be generated using '`hmmbuild`' from the HMMER package and converted to a profile using '`htop`' from the PFTOOLS package. Profiles generated with '`pfmake`' are usually more sensitive than those generated with '`hmmbuild`'. In some cases this means that they are less discriminating. If it is observed that the default method causes false positives, it can be attempted to use the '`hmmbuild`' method to see if it solves the problem. See *HAMAP 2003 paper* for a discussion.

  In a few HAMAP families, '`hmmbuild`' was able to avoid false positives and negatives, while '`pfmake`' was not:

  - for closely related protein families;
  - for certain very short proteins ('`pfmake`' gave very low scores).

Example header lines:

```
CLUSTAL
CLUSTAL W (1.83) multiple sequence alignment template=XYLA_ECOLI template=XYLA_ACTMI
CLUSTAL W (1.83) multiple sequence alignment template=XYLA_ECOLI profile_method=hmmbuild
MUSCLE (3.52) multiple sequence alignment
```

Appendix B Sample UniRules entries

Example of a ’Domain’ UniRule

```
AC   PRU00241;
DC   Domain;
TR   PROSITE; PS50903; RUBREDOXIN_LIKE; 1; level=0
XX
Names: Rubredoxin-like domain
Function: It is involved in electron transfer processes.
XX
CC   -!- SIMILARITY: Contains # rubredoxin-like domain.
DR   PROSITE; PS00202; RUBREDOXIN; 0-1; trigger=no
case <FTGroup:1>
GO   GO:0009490; F:mononuclear iron electron carrier
GO   GO:0006810; P:transport
GO   GO:0006118; P:electron transport
KW   Transport
KW   Electron transport
KW   Metal-binding
KW   Iron
end case
XX
FT   From: PS50903
FT   DOMAIN     from     to       Rubredoxin-like #.
FT   METAL         6      6       Iron #1 (By similarity).
FT   Group: 1; Condition: C
FT   METAL         9      9       Iron #1 (By similarity).
FT   Group: 1; Condition: C
FT   METAL        38     38       Iron #1 (By similarity).
FT   Group: 1; Condition: C
FT   METAL        41     41       Iron #1 (By similarity).
FT   Group: 1; Condition: C
XX
Chop: Nter=0; Cter=0;
Size: 34-54;
Related: None;
Repeats: 2;
Topology: Cytoplasmic;
Example: Q9V099;
Scope:
 Bacteria
 Archaea
# $Log: ex_domain.txt,v $
# Revision 1.5  2007/07/25 09:51:28  lesaux
# Doc updated version 1 LSV/CR
#
//
```

Example of a ’Protein’ UniRule

```
AC   MF_00198;
DC   Protein; auto
TR   HAMAP; MF_00198; -; 1; level=0
XX
ID   SPEE
case <OC:Bacteria>
DE   Spermidine synthase (EC 2.5.1.16) (Putrescine aminopropyltransferase)
DE   (PAPT) (SPDSY).
end case
case <OC:Archaea>
```

```
DE   Probable spermidine synthase (EC 2.5.1.16) (Putrescine
DE   aminopropyltransferase) (PAPT) (SPDSY).
end case
GN   Name=speE;
XX
CC   -!- FUNCTION: Catalyzes the production of spermidine from putrescine
CC       and decarboxylated S-adenosylmethionine (dcSAM), which serves as
CC       an aminopropyl donor (By similarity).
CC   -!- CATALYTIC ACTIVITY: S-adenosylmethioninamine + putrescine = 5'-S-
CC       methyl-5'-thioadenosine + spermidine.
CC   -!- PATHWAY: Amine and polyamine biosynthesis; spermidine
CC       biosynthesis; spermidine from putrescine: step 1/1.
case <OC:Proteobacteria>
CC   -!- SUBUNIT: Homodimer (By similarity).
else case <OC:Thermotogales>
CC   -!- SUBUNIT: Homotetramer (By similarity).
else
CC   -!- SUBUNIT: Homodimer or homotetramer (By similarity).
end case
CC   -!- SIMILARITY: Belongs to the spermidine/spermine synthase family.
XX
DR   Pfam; PF01564; Spermine_synth; 1; trigger=no
DR   TIGRFAMs; TIGR00417; speE; 1; trigger=no
DR   PROSITE; PS01330; SPERMIDINE_SYNTHASE_1; 1; trigger=no
DR   PROSITE; PS51006; SPERMIDINE_SYNTHASE_2; 1; trigger=no
XX
KW   Polyamine biosynthesis
KW   Spermidine biosynthesis
KW   Transferase
XX
GO   GO:0004766; F:spermidine synthase activity
GO   GO:0008295; P:spermidine biosynthetic process
XX
FT   From: SPEE_THEMA (Q9WZC2)
FT   REGION      152    153       S-adenosylmethioninamine binding (By
FT                                similarity).
FT   Condition: [DN]-[AGV]
FT   BINDING      46     46       S-adenosylmethioninamine (By similarity).
FT   Condition: [QHNR]
FT   BINDING     101    101       S-adenosylmethioninamine (By similarity).
FT   Condition: [DE]
FT   BINDING     121    121       S-adenosylmethioninamine (By similarity).
FT   Condition: [ED]
FT   BINDING     170    170       S-adenosylmethioninamine (By similarity).
FT   Condition: D
FT   BINDING     173    173       Putrescine (By similarity).
FT   Condition: [DE]
XX
Size: 261-366;
Related: None;
Template: P09158; P70998; Q9WZC2; Q8U4G1; O25503;
Scope:
 Bacteria
  not in AGRT5, ANASP, BACTN, BORBR, BORBU, BORPA, BORPE, BRAJA, BRUME,
  BRUSU, BUCBP, CAMJE, BLOFL, CAUCR, CHLCV, CHLMU, CHLPN, CHLTE, CHLTR,
  CORGL, COXBU, DEIRA, ENTFA, FUSNN, GLOVI, HAEDU, HAEIN, HELHP, LACLA,
  LACPL, LISIN, LISMO, MYCGA, MYCGE, MYCLE, MYCPE, MYCPN, MYCPU, PASMU,
  PORGI, PSEPK, RHILO, RHIME, RICCN, RICPR, STAAM, STAAN, STAAW, STAES,
  STRA3, STRA5, STRMU, STRP3, STRP8, STRP1, SYNEL, SYNY3, TREPA, TROW8,
  TROWT, UREPA, VIBCH, VIBPA, WIGBR
 Archaea
  not in HALSA, METAC, METKA, METMA, METTH
Fusion:
 NT: <Unknown>
 CT: <Unknown>
Duplicate: in AQUAE, BACAN, BACCR, LEPIN, PSEAE, RALSO, STRCO, THETN
Plasmid: in RALSO
Comments: None
**In Buchnera sp. only speE and speD are present, neither the pathway from ornithine
**nor the pathway from arginine are complete.
XX
# $Log: ex_protein.txt,v $
# Revision 1.1  2007/07/25 09:51:28  lesaux
# Doc updated version 1 LSV/CR
#
//
```