Patterns and regular expressions for analysis of biological sequences

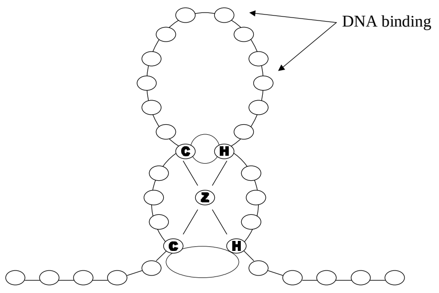
Björn Olsson
Högskolan i Skövde
Department of Computer Science
Bioinformatics Group

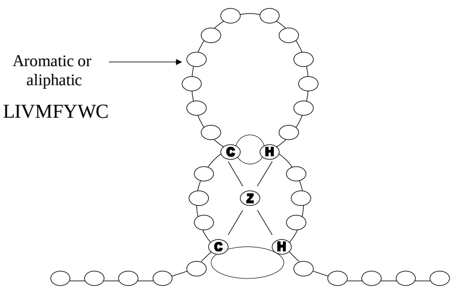
Why patterns?

- Patterns often correspond to functional or structural features.
- When constructing patterns, PROSITE developers look for motifs corresponding to:
  - Cysteines forming disulphide bonds
  - Regions involved in binding to other molecules (including other proteins)
  - Catalytically active regions
  - etc

Zink finger domain:

- DNA binding (usually by 3 or more zink fingers)
- The C_2H_2 zink finger has a loop of 12 amino acids, anchored by two cysteines and two histidines co-ordinating a zink atom.





C-x(2,4)-C-x(3)-[LIVMFYWC]-x(8)-H-x(3,5)-H

EGF-like domain:

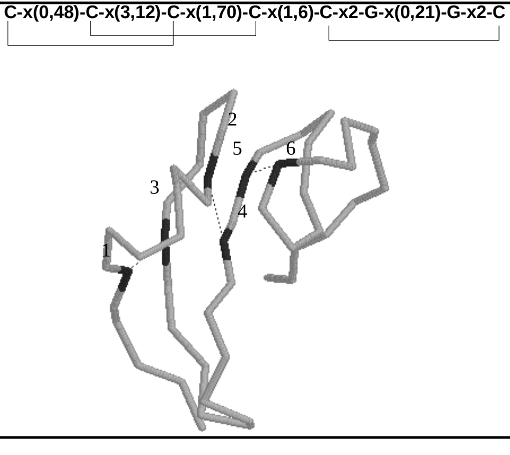
- Identified in epidermial growth factor, but present in many other proteins
- Function unclear, but found in extracellular part of membrane proteins
- Includes six cysteins involved in disulfide bonds

```
          2                    4        5                    6
          +----------------+        +-----------------+
          |                |        |                 |
 C-x(0,48)-C-x(3,12)-C-x(1,70)-C-x(1,6)-C-x2-G-x(0,21)-G-x2-C
 |                |
 +----------------+
 1                3
```

`'C': conserved cysteine involved in a disulfide bond.`

`'G': often conserved glycine`

C-x(0,48)-C-x(3,12)-C-x(1,70)-C-x(1,6)-C-x2-G-x(0,21)-G-x2-C



C-x(0,48)-C-x(3,12)-C-x(1,70)-C-x(1,6)-C-x2-G-x(0,21)-G-x2-C

```
CMLPIF..ECKQF..SECNSYT..........GRCECIEG.....FAGDDC
CDQN....PCLNG..GACLPYLINE...VTHLYTCTCENG.....FQGDKC
CFQS....DCKND..GFCQSPSD........EYACTCQPG.....FEGDDC
CFDN....NCEY....QCQPVGR.......SEHKCICAEGFAPVPGAPHKC
CTIKN...QCMNF..GTCEPVLQG....NAIDFICHCPVG.....FTDKVC
CGILN...GCEN...GRCVRVQEG........YTCDCLDG.YHLDTAKMTC
CKTKEAGFVCKH....GCRSTGK........AYECTCPSG.STVAEDGITC
CRKN....ACLHN..AECRNTW........NDYTCKCPNG.....YKGKKC
CPRNVS..ECS....HDCVLTSE........GPLCFCPEG.SVLERDGKTC
CTDFVDV.PCSH....FCNNFIG........GYFCSCPPE.YFLHDDMKNC
CLDNN..GGCSH....VCNDLKI........GYECLCPDG..FQLVAQRRC
CEEDNG..GCSH....LCLLSPR......EPFYSCACPTG.VQLQDNGKTC
CADGNG..GCSH....LCFFTPR........ATKCGCPIG.LELLSDMKTC
CARDNG..GCSH....ICIAKGD.......GTPRCSCPVH.LVLLQNLLTC
CKLRKG..NCSS...TVCGQDLQ........SHLCMCAEG.YALSRDRKYC
CPKA....NCSH....FCIDRRD.......VGHQCFCAPG.YILSENQKDC
CDSH....PCLHG..GTCEDDG........REFTCRCPAG.....KGGAVC
CVLEP...NCIH...GTCNKP...........WTCICNEG.....WGGLYC
CNPL....PCNEDGFMTCKDGQ........ATFTCICKSG.....WQGEKC
CNAEFQN.FCIH...GECKYIEH......LEAVTCKCQQE.....YFGERC
CPLSHDG.YCLHD..GVCMYIEALD......KYACNCVVG.....YIGERC
```

Thiolases. Enzymes of two types:

– Thiolase I involved in degradative pathways such as fatty acid beta-oxidation

– Thiolase II involved in biosynthetic pathways such as poly beta-hydroxybutyrate synthesis or steroid biogenesis

“There are two conserved cysteine residues important for thiolase activity. The first located in the N-terminal section of the enzymes is involved in the formation of an acyl-enzyme intermediate; the second located at the C-terminal extremity is the active site base involved in deprotonation in the condensation reaction.” (Prosite 17.26, PDOC_00092)

[LIVM]-[NST]-x(2)-C-[SAGLI]-[ST]-[SAG]-[LIVMFYNS]-x- [STAG]-[LIVM]-x(6)-[LIVM]

[AG]-[LIVMA]-[STAGCLIVM]-[STAG]-[LIVMA]-C-x-[AG]-x-[AG]-x- [AG]-x-[SAG]

```
TSNIAREAALLAGVPDKIPAHTVTLACISSNVAMTTGMGMLATGNANAIIAGGVELL
TSNVAREAALGAGFSDKTPAHTVTMACISSNQAMTTAVGLIASGQCDVVVAGGVELM
GQNPARQAAIKAGLPAMVPAMTINKVCGSGLKAVMLAANAIMAGDAEIVVAGGQENM
GQNPARQTTLMAGLPHTVPAMTINKVCGSGLKAVHLAMQAVACGDAEIVIAGGQESM
GQNPARQAHIKVGLPRESAAWVINQVCGSGLRTVALAAQQVLLGDARIVVAGGQESM
GQNPARQAAMKAGVPQEATAWGMNQLCGSGLRAVALGMQQIATGDASIIVAGGMESM
GQNPARQSAIKGGLPNSVSAITINDVCGSGLKALHLATQAIQCGEADIVIAGGQENM
GQNPARQALLKSGLAETVCGFTVNKVCGSGLKSVALAAQAIQAGQAQSIVAGGMENM
GQNPARQAALKAGIEKEIPSLTINKVCGSGLKSVALGAQSIISGDADIVVVGGMENM
GQNPARQASFKAGLPVEIPAMTINKVCGSGLRTVSLAAQIIKAGDADVIIAGGMENM
GQMPARQAAVAAGIGWDVPALTINKMCLSGLDAIALADQLIRAGEFDVVVAGGQESM
GQIPSRLAARLAGMPWSVPSETLNKVCASGLRAVTLCDQMIRAQDADILVAGGMESM
GQAPARQVALKAGLPDSIIASTINKVCASGMKAVIIGAQNIICGTSDIVVVGGAESM
```

[LIVM]-[NST]-x(2)-C-[SAGLI]-[ST]-[SAG]-[LIVMFYNS]-x- [STAG]-[LIVM]-x(6)-[LIVM]

Basic syntax for PROSITE patterns

`[AS]`	A and S allowed.
`D`	D allowed.
`x4`	Four arbitrary residues.
`{PG}`	Any residue except P and G.
`[FY]2`	Two positions where F and Y allowed.
`x(3,7)`	Minimum 3 and maximum 7 residues.

`D-[IVL]-x(4)-{PG}-C-x(4,10)-[DE]-R-[FY]2-Q`

Additional syntax (rarely used)

- Occasional patterns are anchored to the beginning or end of the sequence:

[KRHQSA]-[DENQ]-E-L>

must match at the end of the sequence.

VTLACISSNVAMTTGMGMLATGNANAIIAGQNEL

Wildcards

`G-x2-[LIV]-x(4,7)-R-x-[GA]`

- **`x`** represents an arbitrary symbol.
- **`xN`** represents a sub-sequence of **`N`** arbitrary symbols.
- **`x(i,j)`** represents a sub-sequence of arbitrary symbols with minimum length **`i`** and maximum length **`j`**.

Wildcards

G-x2-[LIV]-x(4,7)-R-x(5,6)-[GA]

- The wildcard **x(i,j)** has flexibility **j-i+1**.
- Total flexibility of a pattern can be defined as the product of the flexibilities of its wildcards, i.e. in this example pattern (7-4+1)+(6-5+1) = 4x2 = 8.
- High flexibility increases search time, since the number of potential matches increases exponentially.

A-x(4,7)-C-x(5,6)-D

```
VTLACISSNVAMTTGMGMLATGNANAIIAGQNEL
A....C.....D
A....C......D
A.....C.....D
A.....C......D
A......C.....D
A......C......D
A.......C.....D
A.......C......D
```

Flexibility *N* means *N* potential matches per position.

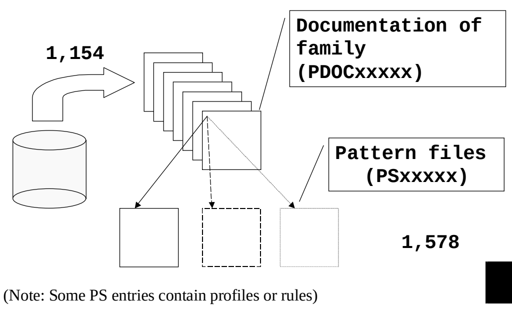
Why patterns?

- Fast database search.
- Interpretable models.
- Simplicity.
- Highlight the conserved features.

PROSITE

- First and most ”famous” pattern database.
- Patterns for over 1,000 protein families.
- ”.. with appropriate computational tools it can rapidly and reliably identify to which known family of protein (if any) the new sequence belongs”.

PROSITE’s structure



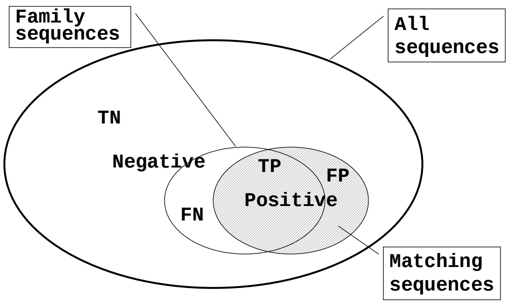
(Note: Some PS entries contain profiles or rules)

Classification accuracy

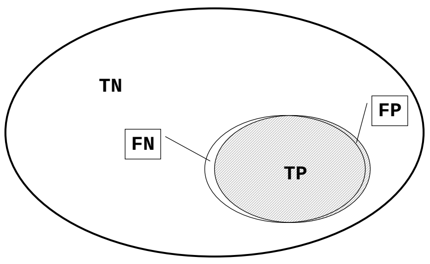
- Also called ”discriminatory power” or ”diagnostic power”.
- Ideally PROSITE should be a database of diagnostic patterns:

  ”.. with appropriate computational tools it can rapidly and reliably identify to which known family of protein (if any) the new sequence belongs”.

- ***Positive***:
  Sequence matching a given pattern.
- ***Negative***:
  Sequence not matching a given pattern.
- ***True***:
  A correct result.
- ***False***:
  An incorrect result.
- ***True positive***:
  A matching family sequence.
- ***True negative***:
  A non-matching non-family sequence.
- ***False positive***:
  A matching non-family sequence.
- ***False negative***:
  A non-matching family sequence.



We want to minimize FN + FP.



A ***diagnostic*** pattern matches all family sequences and no other sequence, i.e. FN = FP = 0.

Classification accuracy

- PROSITE's PS files give the following measures for each pattern:

Precision: $\frac{TP}{(TP+FP)}$

Recall: $\frac{TP}{(TP+FN)}$

(More often called **specificity** and **sensitivity**)

Example

- The flavodoxin family (PS00201):

True positives:	32
False positives:	7
False negatives:	2

Precision (specificity): $\frac{32}{(32+7)} \approx 0.82$

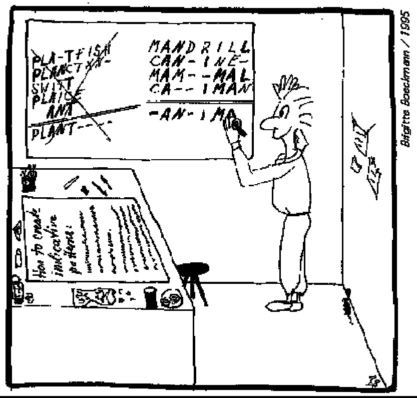
Recall (sensitivity): $\frac{32}{(32+2)} \approx 0.94$

Classification accuracy varies...

Family	TP	FP	FN	Spec	Sens	%id
14_3_3	81	0	0	1	1	69
2_HacidDH	40	0	2	1	.95	32
7_tm1	869	50	70	.95	.93	19
7_tm2	55	0	4	1	.93	34
7_tm3	22	0	0	1	1	38
A2M	23	0	3	1	.88	27
AAA	123	0	5	1	.96	28
ABC_tran	462	42	16	.92	.95	26
Acyl_CoA_dh	30	0	0	1	1	23
Cytochrome_c	772	454	7	.48	.98	28

.. and this is classification accuracy on the training set!

How we develop Prosite patterns!



PROSITE's method

- 1) Study literature on the protein family.
- 2) Study one or more alignments of sequences belonging to the family. Look for motifs corresponding to, for example:
  - Catalytically active region
  - Cysteines forming disulphide bonds
  - Regions involved in binding to other molecules (including other proteins)

PROSITE's method

- 3) Identify a conserved sequence of 4-5 residues within the chosen motif.
- 4) Search SWISS-PROT for sequences matching this initial pattern.
- 5) Extend the pattern until its classification accuracy is "adequate".

Automated pattern construction

- Patterns can be built bottom-up or top-down:
  - BU: Enumerate possible patterns, and rank them according to the number of occurrences in the sequences.
  - TD: Compare (align) the sequences and find similarities from which patterns can be built.
- Advantages of BU:
  - Guarantees finding the best pattern for a given pattern size
  - Not affected by multiple alignment quality
- Advantages of TD:
  - Faster
  - Conserved positions in multiple alignment are likely to be biologically meaningful

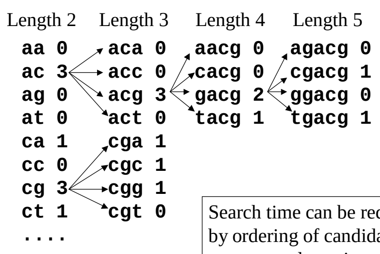
PRATT

- Uses a heuristic bottom-up approach.

Sequences

```
tcgacgga
ttgacgca
gctacgaa
```

Enumeration of patterns



```
Length 2  Length 3  Length 4  Length 5
aa 0      aca 0     aacg 0    agacg 0
ac 3      acc 0     cacg 0    cgacg 1
ag 0      acg 3     gacg 2    ggacg 0
at 0      act 0     tacg 1    tgacg 1
ca 1      cga 1
cc 0      cgc 1
cg 3      cgg 1
ct 1      cgt 0
....
```

Search time can be reduced by ordering of candidate patterns and pruning of the search space.

```
tcgacgga            tcgacgga
ttgacgca            ttgacgca
gctacgaa            gctacgaa

g-a-c-g (2)         g-a-c-g (2)
     ↓                   ↓
[gt]-a-c-g (3)      g-x(0,2)-a-c-g (3)
```

- A candidate pattern can be generalized so that it matches more sequences.
- How to choose between alternative generalizations?
- PRATT uses pattern information content as a guide.

Score of pattern P is defined as:

$$\boxed{I(P) = \sum_{i=1}^{p} I(A_i) - c \cdot \sum_{k=1}^{p-1} (j_k - i_k)}$$

where:	p = number of elements
	c = a constant (0.5)
	i_k = lower length for wildcard k
	j_k = upper length for wildcard k

The second term measures information content of wildcards. For the pattern C-x(3,5)-[DE]-x(2)-A we get:

$$c \cdot \sum_{k=1}^{p-1} (j_k - i_k) = 0.5 \cdot \left((5-3) + (2-2)\right) = 1.0$$

$$\boxed{I(P) = \sum_{i=1}^{p} I(A_i) - c \cdot \sum_{k=1}^{p-1} (j_k - i_k)}$$

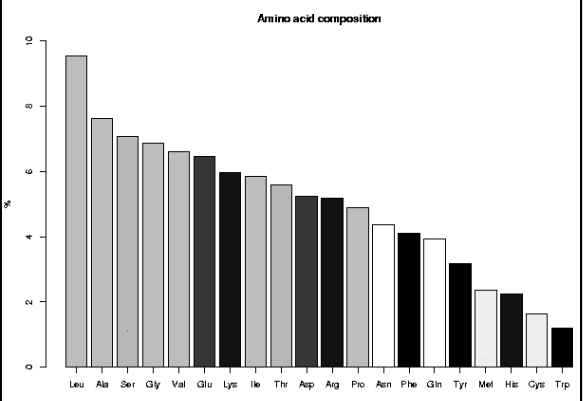
The first term measures information content of symbol elements, which is defined as:

$$I(A_i) = -\sum_{a \in A} (p_a \cdot \log(p_a)) + \sum_{a \in A_i} \left(\frac{p_a}{p_{A_i}} \cdot \log\left(\frac{p_a}{p_{A_i}} \right) \right)$$

where:	p_a = probability of amino acid a
	$p_{A_i} = \sum_{a \in A_i} p_a$

The pattern C-x(3,5)-[DE]-x(2)-A contains symbols CDEA:

$$p_C = (0.02 \cdot \log 0.02) = -0.034$$
$$p_A = (0.08 \cdot \log 0.08) = -0.088$$



PRATT parameters

C%	Minimum percentage of matching seqs
PP	Pattern position (anywhere, complete seq, start)
PL	Max pattern length (C-x(3,5)-[DE] has length 7)
PN	Max pattern symbols (elements)
PX	Max wildcard length (wildcard x(3,5) has length 5)
FN	Max nr of flexible wildcards
FL	Max flexibility of wildcards (wildcard x(3,5) has 2)
FP	Max product of flexibilities

PRATT parameters

BI	Use file of symbols for initial pattern search and during refinement (in pattern C-x(3,5)-[DE] symbols are C and DE)
BN	Specify that the first BN lines of the file contain symbols for initial pattern search
S	Scoring method (information content, and various other)

Fuzzy Regular Expressions

- Method of making patterns more general, so that remote homologues can be found.
- Amino acids grouped, e.g.:
  - Aromatic: FYW
  - Acidic: HKR
  - Hydrophobic: ILVM
  - etc.
- Used in the system eMOTIF maker.

Fuzzy Regular Expressions

```
ADLGAVFALCDRYFQ
SDVGPRSCFCERFYQ
ADLGRTQNRCDRYYQ
ADIGQPHSLCERYFQ
```

Create pattern

`[AS]-D-[IVL]-G-x5-C-[DE]-R-[FY]2-Q`

Generalize

`[ASGPT]-D-[IVLM]-G-x5-C-[DENQ]-R-[FYW]2-Q`

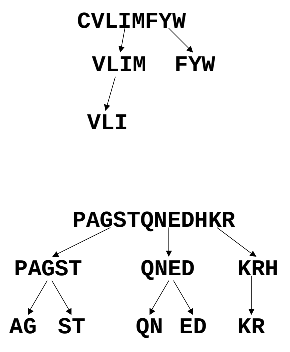
eMOTIF maker

http://motif.stanford.edu/emotif/emotif-maker.html

- Makes fuzzy pattern from multiple alignment using overlapping groups:

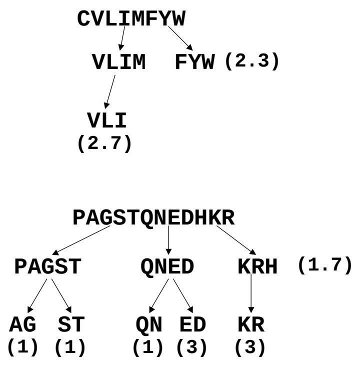
1. AG	7. KR
2. ST	8. KRH
3. PAGST	9. VLI
4. QN	10. VLIM
5. ED	11. FYW
6. QNED	12. PAGSTQNEDHKR
	13. CVLIMFYW

- For each alignment column, chooses smallest group containing all symbols from the column.



MD-matrix for 250 PAM

	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W
C	12																			
S	0	2																		
T	2	1	3																	
P	3	1	0	6																
A	2	1	1	1	2															
G	3	1	0	1	1	5														
N	4	1	0	1	0	0	2													
D	5	0	0	1	0	1	2	4												
E	5	0	0	1	0	0	1	3	4											
Q	5	1	1	0	0	1	1	2	2	4										
H	3	1	1	0	1	2	2	1	1	3	6									
R	4	0	1	0	2	3	0	1	1	1	2	6								
K	5	0	0	1	1	2	1	0	0	1	0	3	5							
M	5	2	1	2	1	3	2	3	2	1	2	0	0	6						
I	2	1	0	2	1	3	2	2	2	2	2	2	2	2	5					
L	6	3	2	3	2	4	3	4	3	2	2	3	3	4	2	6				
V	2	1	0	1	0	1	2	2	2	2	2	2	2	2	4	2	4			
F	4	3	3	5	4	5	4	6	5	5	2	4	5	0	1	2	1	9		
Y	0	3	3	5	3	5	2	4	4	4	0	4	4	2	1	1	2	7	10	
W	8	2	5	6	6	7	4	7	7	5	3	2	3	4	5	2	6	0	0	17



eMotif, example

```
      LASPTHEDLSL
      LASPTNEELSM
      IASPIEVQVSL
      IAGPIVQQISL
       A P     S
  3     X             3. PAGST
  6          X        6. QNED
  9   X        X      9. VLI
 10              X   10. VLIM
```

`[VLI]-A-[PAGST]-P-x3-[QNED]-[VLI]-S-[VLIM]`

The EMOTIF database

http://fold.stanford.edu/emotif/

- Huang & Brutlag, *NAR* **29**(1): 202-204, 2001.
- > 170,000 motifs
- Representing biochemical properties and biological functions.
- All 8,244 alignments in PRINTS.
- All 7,697 non-PRINTS alignments in BLOCKS+.

Pros and cons of patterns

+ Fast search.
+ Simple and interpretable models.
- Limited power of expression.
- Discrete results (”yes/no”).
- Lack of reliable methodology for automated design of patterns.
- Only use information from a single motif.

Regular expressions

- Patterns are represented by regular expressions.
- Regular expressions (with varying syntax) are used in:
  - operating system tools (*awk*, and *grep* in UNIX)
  - programming languages (e.g. PERL)
  - text editors
  - search engines
- In all cases to search text for occurrence of a pattern.

Regular expressions in UNIX/Linux

```
ID   ASN_GLYCOSYLATION; PATTERN.
AC   PS00001;
DT   APR-1990 (CREATED); APR-1990 (DATA UPDATE).
DE   N-glycosylation site.
PA   N-{P}-[ST]-{P}.
CC   /TAXO-RANGE=??E?V;
CC   /SITE=1,carbohydrate;
CC   /SKIP-FLAG=TRUE;
DO   PDOC00001;
```

```
> grep ’CC’ file1
> CC   /TAXO-RANGE=??E?V;
> CC   /SITE=1,carbohydrate;
> CC   /SKIP-FLAG=TRUE;
>
```

```
ID   ASN_GLYCOSYLATION; PATTERN.
AC   PS00001;
DT   APR-1990 (CREATED); APR-1990 (DATA UPDATE).
DE   N-glycosylation site.
PA   N-{P}-[ST]-{P}.
CC   /TAXO-RANGE=??E?V;
CC   /SITE=1,carbohydrate;
CC   /SKIP-FLAG=TRUE;
DO   PDOC00001;
```

```
> grep ’PA’ file1
> ID   ASN_GLYCOSYLATION; PATTERN.
> PA   N-{P}-[ST]-{P}.
>
```

```
> grep ’^PA’ file1
> PA   N-{P}-[ST]-{P}.
>
```

```
> grep ’^PA’ prosite.dat > prosite_patterns.dat
>
```

```
ID   ASN_GLYCOSYLATION; PATTERN.
AC   PS00001;
DT   APR-1990 (CREATED); APR-1990 (DATA UPDATE).
DE   N-glycosylation site.
PA   N-{P}-[ST]-{P}.
CC   /TAXO-RANGE=??E?V;
CC   /SITE=1,carbohydrate;
CC   /SKIP-FLAG=TRUE;
DO   PDOC00001;
```

```
> grep ’^D[EOT]’ file1
> DT   APR-1990 (CREATED); APR-1990 (DATA UPDATE);
> DE   N-glycosylation site.
> DO   PDOC00001;
>
```

```
> grep ’^D.’ file1
> DT   APR-1990 (CREATED); APR-1990 (DATA UPDATE);
> DE   N-glycosylation site.
> DO   PDOC00001;
>
```

```
ID   ASN_GLYCOSYLATION; PATTERN.
AC   PS00001;
DT   APR-1990 (CREATED); APR-1990 (DATA UPDATE).
DE   N-glycosylation site.
PA   N-{P}-[ST]-{P}.
CC   /TAXO-RANGE=??E?V;
CC   /SITE=1,carbohydrate;
CC   /SKIP-FLAG=TRUE;
DO   PDOC00001;
```

```
> grep ';$' file1
> AC   PS00001;
> CC   /TAXO-RANGE=??E?V;
> CC   /SITE=1,carbohydrate;
> CC   /SKIP-FLAG=TRUE;
> DO   PDOC00001;
>
```

```
ID   ASN_GLYCOSYLATION; PATTERN.
AC   PS00001;
DT   APR-1990 (CREATED); APR-1990 (DATA UPDATE).
DE   N-glycosylation site.
PA   N-{P}-[ST]-{P}.
CC   /TAXO-RANGE=??E?V;
CC   /SITE=1,carbohydrate;
CC   /SKIP-FLAG=TRUE;
DO   PDOC00001;
```

```
> grep '\.$' file1
> ID   ASN_GLYCOSYLATION; PATTERN.
> DT   APR-1990 (CREATED); APR-1990 (DATA UPDATE).
> DE   N-glycosylation site.
> PA   N-{P}-[ST]-{P}.
>
```

- The *grep* program uses:

Symbol	Action
.	Match any character (except newline)
*	Match zero or more of the preceding character
^	Match the beginning of the line
$	Match the end of the line
\	Turn off special meaning of following character
[ ]	Match one of the enclosed characters
\{ \}	Match a range of instances

\{*n*\}	Matches exactly *n* occurrences.
\{*n*,\}	Matches at least *n* occurrences.
\{*n*,*m*\}	Matches at least *n* and at most *m* occurrences.

```
SKKIGLFYGT
AKIGLFFGSN
KIGIFFSTST
MKISILYSSK
MKIVYWSGTG
```

Which sequences contain two aliphatic hydrophobes?

```
> grep '[LIV].*[LIV]' file2
> SKKIGLFYGT
> AKIGLFFGSN
> KIGIFFSTST
> MKISILYSSK
> MKIVYWSGTG
```

Which sequences contain two aliphatic hydrophobes in succession?

```
> grep '[LIV]\{2\}' file2
> MKISILYSSK
> MKIVYWSGTG
```

\{*n*\}	Matches exactly *n* occurrences.
\{*n*,\}	Matches at least *n* occurrences.
\{*n*,*m*\}	Matches at least *n* and at most *m* occurrences.

```
SKKIGLFYGT
AKIGLFFGSN
AKIGLFFGSN
KIGIFFSTST
MKISILYSSK
MKIVYWSGTG
```

Two tiny residues that are exactly one residue apart?

```
> grep '[ACGS].[ACGS]' file2
> KIGIFFSTST
> MKIVYWSGTG
```

Two tiny residues that are one or two residues apart?

```
> grep '[AGCS].\{1,2\}[AGCS]' file2
> AKIGLFFGSN
> KIGIFFSTST
> MKIVYWSGTG
```

Two tiny residues that are at least five residues apart?

```
> grep '[AGCS].\{5,\}[AGCS]' file2
> SKKIGLFYGT
> AKIGLFFGSN
> KIGIFFSTST
```

PROSITE patterns in *grep*

- Simply using *grep*, we could find all lines in a sequence file that match a PROSITE pattern:

G-x2-[LIV]-R-x-[GA]

```
> grep 'G.{\2\}[LIV]R.[GA]' seq_file
```

G-x2-[LIV]-x(4,7)-R-x(5,6)-[GA]

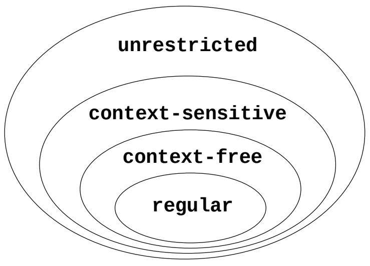
```
> grep 'G..[LIV].\{4,7\}R.\{5,6\}[GA]' seq_file
```

Regular expressions in PERL

- The programming language PERL uses a regular expression syntax that is similar to *grep*'s.

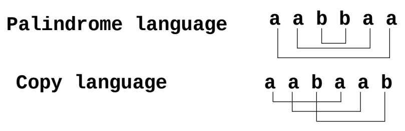
```
$pattern = ”G.{\2\}[LIV]R.[GA]”;
while ($name=<SPROT>) {
    $seq = <SPROT>;
    if ($seq =~ /$pattern/) {
     $nr_matching_seqs++;
     printf STDOUT "$name";
     printf STDOUT "$seq";
    }
}
```

The Chomsky hierarchy of transformational grammars



What patterns can't do

- Regular grammars cannot describe palindrome languages or copy languages, i.e. cannot handle long-range interactions.



Palindrome languages

'Doc, note. I dissent. A fast never prevents a fatness. I diet on cod.'

'Doc no teId i .ssentAf a stneverp reven tsaf a .tnessId i .eton cod.'

- Do palindrome languages occur in biological sequences?
- Consider RNA secondary structure (stem loops, ”hairpins”), or anti-parallel beta strands, and the pairwise dependencies between positions that these result in (in the sequence).

```
   1            2            3

  A A          C A          C A
 G   A        G   A        G   A
  G•C          U•A          UxC
  A•U          C•G          CxU
  C•G          G•C          GxG
```

```
1 CAGGAAACUG
2 GCUGCAAAGC
3 GCUGCAACUG
```

Identical positions

	2	3
1	3	5
2	-	7

Sequences 1 and 2 are the most dissimilar, but they can fold to the same structure, because they share the same pattern of A-U and G-C pairs (purine-pyrimidine).

The structure imposes a set of nested pairwise constraints, just like a palindrome language.

```
                        1      2      3
1 CAGGAAACUG           A A    C A    C A
2 GCUGCAAAGC          G   A  G   A  G   A
3 GCUGCAACUG           G•C    U•A    UxC
                       A•U    C•G    CxU
                       C•G    G•C    GxG
```

How to formulate a pattern, when allowed symbols at position 10 depends on which symbol we see at position 1?

A context-free grammar is required:

$$\begin{array}{lll} S & \rightarrow & aW_1u \mid cW_1g \mid gW_1c \mid uW_1a \\ W_1 & \rightarrow & aW_2u \mid cW_2g \mid gW_2c \mid uW_2a \\ W_2 & \rightarrow & aW_3u \mid cW_3g \mid gW_3c \mid uW_3a \\ W_3 & \rightarrow & gaaa \;\mid gcaa \end{array}$$

Exercise idea

1) Write a program that finds patterns in a multiple alignment by looking for columns:
   - containing no gap-characters
   - containing a set of amino acids in which all pairs have positive scores in the PAM250 substitution matrix
   - being at most 2 columns apart

   A valid pattern must model at least four columns.
2) Test your program on alignments downloaded from Pfam and check your patterns through the PROSITE scanProsite interface.
3) Compare the patterns your program finds with those in PROSITE for the same family.