TEC | Tecnológico de Costa Rica

Escuela de Ingeniería en Computación

Programa de Maestría en Computación

# Application of Nearest Neighbour Search techniques to Peptide identification from Mass Spectrometry

A thesis submitted in partial fulfilment to opt for the degree of

*Magister Scientiæ* in Computer Science

Author
José de Jesús Ulate Cárdenas

Supervisor
Francisco Torres Rojas, PhD.

June 5, 2018

www.tec.ac.cr

Programa de Maestría
en Computación
TEC | Tecnológico de Costa Rica
Teléfono: 2550-2402

## APROBACIÓN DE LA TESIS

"APPLICATION OF NEAREST NEIGHBOUR SEARCH TECHNIQUES TO PEPTIDE IDENTIFICATION FROM MASS SPECTROMETRY"

## TRIBUNAL EXAMINADOR

Dr. Francisco Torres Rojas
Profesor Asesor

Máster Armando Arce Orozco
Profesor Lector

Máster Karol Jiménez Quesada
Profesor Externo

Dr. Roberto Cortés Morales
Coordinador del Programa
de Maestría en Computación

TEC | Tecnológico de Costa Rica
Maestría en Computación

Junio, 2018

To my beloved Julia and Antonio, my grandparents. For all the years of joy that I will never forget.

---

A mis queridos abuelos Julia y Antonio. Por todos los años de alegría que nunca olvidaré.

# Acknowledgements

I want to thank Alejandro Fernández Woodbridge, PhD, researcher of the Science for Life Laboratory in Stockholm, Sweeden; who provided the initial inspiration for this research. Luis Alejandro Acuña Prado, MSc, mathematics professor of the Costa Rica Institute of Technology, whose course of Functional Analysis and Topology gives us the necessary formality and support. Bruno Lomonte, PhD, researcher of the Clodomiro Picado Institute, who guided us and provided profound insights and revisions of the research results.

Nancy Peréz, MSc, Statistics analyst of the Costa Rica Central Bank, and Ivania Montero, Oficial Translator of the Ministry of Foreign Affairs and Worship of Costa Rica, for the English corrections of this document.

National Collaborative of Advanced Computing (CNCA) of the Costa Rica National High Technology Center (CENAT) for providing the infrastructure for execution of many tests.

Finally, to our family and friends, in particular to Aaron Blanco, for the support during this document development.

**Abstract**

We explored if the application of Nearest Neighbour Search thecniques to the *de novo* sequence of amino acids from Tandem Mass Spectrometry will provide better performance and accuracy. We present an indexation strategy that allows a posterior interpretation of the spectral data using concepts of Topology. We also offer an analysis of the execution time and space needed for both strategies.

**Resumen**

Exploramos si la aplicación de las técnicas de búsqueda del vecino más cercano a la identificación *de novo* desde Tandem Mass Spectrometry proveería un mejor desempeño. Presentamos una estrategia de indexación y una posterior estrategia de búsqueda/interpretación inspirada en conceptos de Topología. También presentamos el análisis del tiempo de ejecución y espacio necesario para ambas estrategias.

# Contents

# List of Figures



[1] High-resolution Isoelectric Focusing

# List of Tables

# Listings

Chapter 1

# Introduction

"But when Jesus heard that, he said unto them, They that be whole need not a physician, but they that are sick."

Matthew 9:12

Once Jesuscrist said: *"They that be whole need not a physician, but they that are sick."* Matthew9:12. When trying to find the cause of disease, we can often find aspects that are the same. However; these aspects are not the goal of the physician. Nowadays, this problem is widespread, more than expected.

In dealing with Cancer research, one of the primary sources of cancer is gene mutations. Many factors such as tobacco, ultraviolet radiation, viruses, and age can cause these mutations. Though the body can fix those mutations [84; 158], sometimes those mutations can remain in the body.

Although, there are *tumour* suppressor genes in the body, (in some way protective genes), some other genes like HER2[1] are related to some kinds of Cancer [16]. Normally, *tumour* suppressor genes limit cell growth by monitoring how quickly cells divide into new cells, repairing mismatched DNA[2], and controlling when a cell dies. When a *tumour* suppressor gene is mutated, cells grow uncontrollably and may eventually form a mass called *tumour*. Tumours are either malignant (harmful) or benign (safe) [72; 106; 149].

MS[3] is one analytic technique used to identify proteins [136]. Though current MS tools do a good work identifying *canonical* proteins [4], they fail when the protein is mutated. More details about this situation are addressed in Section 1.1, Problem [32; 137].

[1]Human Epidermal Growth Factor Receptor 2
[2]Deoxyribonucleic Acid
[3]Mass Spectrometry
[4]The reader can interpret *canonical* protein as a protein present without any mutation according to a reference genome.

## 1.1. Problem

This research addresses the Peptide Sequence Problem with a *de novo* approach. But, first we want to present some background before entering into technical details. These details are thoroughly explained thoroughly in Section 2.1.8.

There are two big problems:

- The performance of the current MS tools is deficient. They take hours, or even days, to detect peptides **[89]**.
- Each genome carries in average **0.7 Mbp** of sequence changes. Current tools will not interpret these changes **[146]**.

According to the project of sequencing 10,545 human genomes at 30-40x coverage with an emphasis on quality metrics, novel variant and sequence discovery; done by Telenti et al.. Each genome carries in an average of **0.7Mbp** of sequence that is not found in the main build of the HG38[5] [146].

Normal variations to a *canonical* reference genome can cause that a particular peptide will not be detected. These variations can be related to race, and some other factors, or even to the unique genotype of a person. Also, in cancer related samples, these variations can be common and the tool cannot detect them.

As Kim and Pevzner mentioned, MS instruments and experimental protocols are rapidly advancing, but software tools to analyse MS/MS are lagging behind [82]. These tools are designed to find an exact match for a peptide. But, in cancer related mutations cases, finding the mutated peptide is the most important part of the solution to a disease. Immunotherapy can train the Immune System to treat cancer [106]. But, the first step to start with Immunotherapy is to recognize what the mutations that need to be fixed are.

In addition to this, the performance of current MS tools is deficient. They take

[5]Human Genome Version 38

hours, or even days, to detect peptides in a given sample. Malm et al. mentioned that it can even take weeks [89] depending on the number of replicates. These tools do not take advantage of the hardware capabilities of the machine where they ran; because they were designed to work on personal computers.

## 1.2. Document Structure

Going into detail, this document presents the findings of the research to evaluate the feasibility of applying NNS techniques to solve the Peptide Sequence Problem. This chapter will present a small prologue of the topics and the state-of-the-art researches in this field. In the following paragraphs there is a short description of the contents of this document.

First, in Theoretical Framework, Chapter 2, you will find an introduction and study of the current field status. We present a small introduction about Molecular Biology, in Section 2.1 where we approach Amino Acids, Proteins, and Proteogenomics. These topics serve as an introduction to Mass Spectrometry. As an example, in particular, we mention High-resolution Isoelectric Focusing Liquid Chromatography Mass Spectrometry. In Section 2.1.8, Peptide Sequence Problem we dig into the subject of this research.

The Section 2.2, Mathematical Fundamentals presents a summary of the topics needed to approach this problem. The Section 2.2.1, Set Theory will provide some basics concepts that we will extend in Section 2.2.2, Algebraic Structures and Section 2.2.3, Topology is the fundamental basis of the Nearest Neighbour Search, especially Definition 2.2.3.1, *Topological space* and Definition 2.2.3.2, *metric space*. Just after that, Section 2.3, Computer Science comes up with some examples of the application these techniques: Space Partitioning and Min Hash in sections 2.3.1.1 and 2.3.2.2.

Theoretical Framework concludes with a review of the related work in the field. We cite: MS-GF+, Mascot, HiXCorr, Sailfish. SEQUEST, Tempest, OpenMS, PepNovo. Section 2.4.9, Other Investigations ends the chapter.

Details of how this research was done are detailed in Chapter 3, Research Methodology. Section 3.2, Proposed Approach presents the initial plan. But after a detailed analysis we found that a *De novo* Approach, Section 4.1, will fitted better with the nature of the NNS techniques. Details are presented in Section 4.1.1, Notation and Definitions. The strategy for the indexation, first part of the solution, is exhibited in Section 4.1.2, Indexation, and the strategy for the search/interpretation of a spectra in Section 4.1.3. This chapter will close with the Metrics review in Section 4.4.1.

We review the formalities of this investigation in Chapter 3, Research Methodology starting with revision of Main Objective and Specific Objectives in Section 3.1. The Section 4.2.2.1, Analysis of Metrics will point to the deliverables of the research that are presented in other parts of this document. The analysis of the results is reviewed in Chapter 4, Results and discussion.

Our conclusions are exhibit in Section 5.1, Conclusions and Recommendations and we provided some recommendations. We finalise Chapter 5 with many possibilities that we wanted to explore in Section 5.2, Future Work.

As part of this research, we include some prototypes. They are in Chapter 6, Prototypes There is an acronyms list in Chapter 7, Acronyms and Chapter 8, License includes Prototype License.

Chapter 2

# Theoretical Framework

"Es lícito aprender hasta del enemigo".

San Mateo

This chapter consists of three main topics: Molecular Biology, Mathematical Fundamentals, and Computer Science. We conclude this chapter with the mention of the work done by other researchers on Related Work. We presented first the Section 2.1, Molecular Biology because it contains the necessary fundamentals to understand the Peptide Sequence Problem, that at some point overlaps with some mathematical definitions. Later, in Section 2.2, Mathematical Fundamentals, we cite definitions and theorems needed to develop this research methodology. In Section 2.3, Computer Science we present some examples of the applications of the NNS techniques to solve other problems.

Important note:

To understand the basics of this research we suggest that the reader, keep in mind this. For commodity to the reader we will use the following color code.

***We tried to solve a Molecular Biology problem with Mathematical techniques applied to Computer Science.***

## 2.1. Molecular Biology

This is a small introduction to the mandatory topics related to Molecular Biology required to understand this research. For further understanding, we suggest the reader refers to text books such as [5; 25; 77].

### 2.1.1. Amino Acids

Amino acids are biologically important organic compounds containing amine (-NH2) and carboxylic acid (-COOH) functional groups, along with a side-chain (R group) specific to each amino acid [19].

In Table 2.1 there is a list of the 20 standard amino acids with codes, mw[1], and

[1]Molecular Weight

**Table 2.1:** Aminoacids Codes, Molecular Mass, and Composition

| Amino Acid | SLC | AA | mw (da) | Composition |
|---|---|---|---|---|
| Isoleucine | I | Ile | 113.2 | $C_6H_{11}NO$ |
| Leucine | L | Leu | 113.2 | $C_6H_{11}NO$ |
| Valine | V | Val | 99.07 | $C_5H_9NO$ |
| Phenylalanine | F | Phe | 147.2 | $C_9H_9NO$ |
| Methionine | M | Met | 131.2 | $C_5H_9NOS$ |
| Cysteine | C | Cys | 103.1 | $C_3H_5NOS$ |
| Alanine | A | Ala | 71.08 | $C_3H_5NO$ |
| Glycine | G | Gly | 57.05 | $C_2H_3NO$ |
| Proline | P | Pro | 97.12 | $C_5H_7NO$ |
| Threonine | T | Thr | 101.1 | $C_4H_7NO_2$ |
| Serine | S | Ser | 87.08 | $C_3H_5NO_2$ |
| Tyrosine | Y | Tyr | 163.2 | $C_9H_9NO_2$ |
| Tryptophan | W | Trp | 186.2 | $C_{11}H_{10}N_{2O}$ |
| Glutamine | Q | Gln | 128.1 | $C_5H_8N_2O_2$ |
| Asparagine | N | Asn | 114.1 | $C_4H_6N_2O_2$ |
| Histidine | H | His | 137.1 | $C_6H_7N_3O$ |
| Glutamic acid | E | Glu | 129.1 | $C_5H_7NO_3$ |
| Aspartic acid | D | Asp | 115.1 | $C_4H_5NO_3$ |
| Lysine | K | Lys | 128.2 | $C_6H_{12}N_2O$ |
| Arginine | R | Arg | 156.2 | $C_6H_{12}N_4O$ |

composition . In Table 2.2 there is an **amino acids/codon** translation. In the subsequent list, we enumerate the 20 standard aminoacids.

1. **Alanine** is an $\alpha$-amino acid [2]. that is used in the biosynthesis of proteins. It contains an $\alpha$-amino group [3], an $\alpha$-carboxylic acid group [4], and a side chain methyl group [5], making it a nonpolar [6], aliphatic [7] amino acid. The human body can synthesise it, so it is considered non-essential in humans. It is encoded by all codons starting with `GC`, namely `GCU`, `GCC`, `GCA`, and `GCG` [27; 111].

2. **Arginine** is an $\alpha$-amino acid; it is encoded by the codons `CGU`, `CGC`, `CGA`, `CGG`,

[2] An amino acid of the general formula $R-CHNH_3 \equiv -COO^-$, that is, the amino in the $\alpha$ position [96].
[3] Which is in the protonated form, $-NH_3^+$ [96].
[4] Which is in the deprotonated form, $-COO-$ [97].
[5] An alkyl group $CH_3$ that is derived from methane by removal of one Hydrogen atom [98].
[6] Not polar; consisting of molecules not having a dipole a nonpolar solvent [99].
[7] Being an organic compound having an open-chain structure [95].

**Table 2.2:** Aminiacid/codon Translations

| Amino Acid | DNA codons |
|---|---|
| Isoleucine | ATT, ATC, ATA |
| Leucine | CTT, CTC, CTA, CTG, TTA, TTG |
| Valine | GTT, GTC, GTA, GTG |
| Phenylalanine | TTT, TTC |
| Methionine | ATG |
| Cysteine | TGT, TGC |
| Alanine | GCT, GCC, GCA, GCG |
| Glycine | GGT, GGC, GGA, GGG |
| Proline | CCT, CCC, CCA, CCG |
| Threonine | ACT, ACC, ACA, ACG |
| Serine | TCT, TCC, TCA, TCG, AGT, AGC |
| Tyrosine | TAT, TAC |
| Tryptophan | TGG |
| Glutamine | CAA, CAG |
| Asparagine | AAT, AAC |
| Histidine | CAT, CAC |
| Glutamic acid | GAA, GAG |
| Aspartic acid | GAT, GAC |
| Lysine | AAA, AAG |
| Arginine | CGT, CGC, CGA, CGG, AGA, AGG |
| Stop codons | TAA, TAG, TGA |

AGA, and AGG. It contains an $\alpha$-amino group, an $\alpha$-carboxylic acid group, and a side chain consisting of a 3-carbon aliphatic straight chain ending in a guanidino group [8] [51]. Arginine is classified as a semi-essential or conditionally essential amino acid [9] [143; 161].

3. **Asparagine**, encoded by the codons AAU and AAC, is an $\alpha$-amino acid. It contains an $\alpha$-amino group, an $\alpha$-carboxylic acid group, and a side chain carboxamide, classifying it as a polar [10], aliphatic amino acid. It is non-essential in humans [121].

4. **Aspartic acid** is an $\alpha$-amino group in the protonated $-NH^+$ 3 form under

[8] relating to or containing the group $H_2NC(=NH)NH$ [100]

[9] Depending on the developmental stage and health status of the individual it can be synthesised [161].

[10] Polar: at physiological pH.

physiological conditions, while its $\alpha$-carboxylic acid group is deprotonated $-COO-$ under physiological conditions. Aspartic acid, it is encoded by the codons `GAU` and `GAC`, also known as aspartate, is an $\alpha$-amino acid. Similar to all other amino acids it contains an amino group and $\alpha$-carboxylic acid. Aspartic acid has an acidic side chain $CH_2COOH$ which reacts with other amino acids, enzymes and proteins in the body [56; 151].

5. **Cysteine** is a non-essential sulfur-containing amino acid in humans. Its formula is $C_3H_7NO_2S$ found in beta-keratin, the main protein in nails, skin, and hair. Cysteine is important in collagen production, as well as skin elasticity and texture. Also required in the manufacture of amino acid taurine, Cysteine is a component of the antioxidant glutathione, and plays a role in the metabolism of essential biochemicals such as coenzyme A [11], heparin [12], and biotin [13] [4; 151].

6. **Glutamic acid** . The salts and carboxylate anions associated with glutamic acid are referred to as glutamates. Glutamic acid contributes to the health of the immune and digestive systems, as well as energy production. It is is an $\alpha$-amino acid with formula $C_5H_9O_4N$. Its molecular structure has two carboxyl groups $-COOH$ and one amino group $-NH_2$ [110; 151].

7. **Glutamine** is encoded by the codons `CAA` and `CAG`, is an $\alpha$-amino acid. Its side chain is like that of glutamic acid, except the carboxylic acid group is replaced by an amide. It is classified as a charge-neutral, polar amino acid. Its formula is $C_5H_{10}N_2O_3$ [165].

8. **Glycine** is the amino acid that has a single hydrogen atom as its side chain. It is the simplest possible amino acid. The chemical formula of glycine is

[11] Coenzyme: a nonprotein compound that is necessary for the functioning of an enzyme [11] .

[12] Heparin, also known as unfractionated heparin, is medication which is used as an anticoagulant [3].

[13] Biotin is a vitamin that is found in small amounts in numerous foods [94].

$NH_2-CH_2-COOH$. Glycine is one of the proteinogenic amino acids [14]. In the genetic code, all codons starting with `GG`, namely `GGU`, `GGC`, `GGA`, `GGG`, code are for glycine [105; 160].

9. **Histidine** is encoded by the codons `CAU` and `CAC`, and it is an $\alpha$-amino acid. It contains an $\alpha$-amino group, a carboxylic acid group, and an imidazole side chain [15], classifying it as a positively charged amino acid at physiological pH [65].

10. **Isoleucine** is encoded by the codons `ATT`, `ATC`, `ATA`, it is an $\alpha$-amino acid. It contains an $\alpha$-amino group, an $\alpha$-carboxylic acid group, and a hydrocarbon side chain, classifying it as a non-polar, uncharged, aliphatic amino acid [83].

11. **Leucine** is an $\alpha$-amino acid that is used in the biosynthesis of proteins. It contains an $\alpha$-amino group, an $\alpha$-carboxylic acid group, and a side chain isobutyl group, making it a non-polar aliphatic amino acid. In the genetic code, it is encoded by the six codons `UUA`, `UUG`, `CUU`, `CUC`, `CUA`, and `CUG` [127].

12. **Lysine**, an amino acid released in the hydrolysis [16] of many common proteins but present in small amounts or lacking in certain plant proteins. First isolated from case in 1889, lysine is one of several so-called essential amino acids for warm-blooded animals [37].

13. **Methionine**, sulfur-containing amino acid obtained by the hydrolysis of most common proteins. First isolated from in 1922, methionine accounts for about 5% of the weight of egg albumin; other proteins contain much smaller amounts of methionine. It is one of several so-called essential amino acids for mammals. In microorganisms, it is synthesised from the amino acids cysteine and aspartic acid [38].

[14] Proteinogenic amino acids are amino acids that are incorporated biosynthetically into proteins during translation [119]

[15] which is partially protonated [34].

[16] A chemical process of decomposition involving the splitting of a bond and the addition of the hydrogen cation and the hydroxide anion of water [101].

14. **Phenylalanine** is an $\alpha$-amino acid with the formula $C_9H_{11}NO_2$. It can be viewed as a benzyl [17] group substituted for the methyl group of alanine, or a phenyl group in place of a terminal hydrogen of alanine [24; 151].

15. **Proline**, an amino acid obtained by hydrolysis of proteins. Its molecule contains a secondary amino group NH rather than the primary amino group $NH_2$ characteristic of most amino acids. Unlike other amino acids, proline, first isolated in 1901, is readily soluble in alcohol. Collagen, the principal protein of connective tissue, yields about 15% proline [40; 151].

16. **Serine**, an amino acid obtainable by hydrolysis of most common proteins, sometimes constituting 5% to 10% by weight of the total product. First isolated in 1865 from sericin, a silk protein, serine is one of several so-called nonessential amino acids for mammals [105; 151].

17. **Threonine** contains an $\alpha$-amino group, a carboxyl group, and a side chain containing a hydroxyl group, making it a polar, uncharged amino acid. In the genetic code it is encoded by the codons `ACT`, `ACC`, `ACA`, and `ACG` [117].

18. **Tryptophan**, an amino acid that is nutritionally important and occurs in small amounts in proteins. It is an essential amino acid. It contains an $\alpha$-amino group, an $\alpha$-carboxylic acid group, and a side chain indole, making it a non-polar aromatic amino acid [42; 151].

19. **Tyrosine** is an amino acid comprising about 1% to 6% by weight of the mixture obtained by hydrolysis [18]. First isolated from casein in 1846 by German chemist Justus, baron von Liebig, tyrosine is particularly abundant in insulin and papain [19], which contain 13% of weight [43; 151].

[17] In organic chemistry, benzyl is the substituent or molecular fragment possessing the structure $C_6H_5CH_2$ .

[18] Chemical process of decomposition involving the splitting of a bond and the addition of the hydrogen cation and the hydroxide anion of water of most proteins [101]

[19] Papain, an enzyme found in papaya fruit

20. **Valine**, an amino acid obtained by hydrolysis of proteins and was first isolated by the German chemist Emil Fischer in 1901. It is one of several so-called essential amino acids for fowl and mammals; i.e., they cannot synthesise it and require dietary sources [44].

### 2.1.2. Proteins

A protein consists of one or more long chains of amino acid residues. Proteins can be considered building blocks of a cell. According to Sawai and Orgel [126], naturally occurring and synthetic polypeptides [20] having mw greater than about 10000$da$, but, the limit is not precise.

### 2.1.3. Peptides

In the IUPAC Compendium of Chemical Terminology, when two or more amino acids combine to form a peptide, the elements of water are removed, and what remains of each amino acid is called an amino acid residue [39; 126]. Peptides are distinguished from proteins based on size, and as an arbitrary benchmark can be understood to contain approximately 50 or fewer amino acids. Amides derived from two or more amino carboxylic acid molecules (the same or different) by formation of a covalent bond [21] from the carbonyl carbon of one to the Nitrogen atom of another with formal loss of water [126].

### 2.1.4. Post-Translational Modification

As defined in the Encyclopedia of genetics a Post-Translational Modification is a biochemical modification that occurs to one or more amino acids on a protein after the protein has been translated by a ribosome [17].

[20] A substance that contains many amino acids [39; 107].

[21] In chemistry, the interatomic linkage that results from the sharing of an electron pair between two atoms [36] .

Protein PTMs[22] increase the functional diversity of the proteome by the covalent addition of functional groups or proteins, proteolytic cleavage of regulatory subunits, or degradation of entire proteins [17; 90]. These modifications include phosphorylation, glycosylation, ubiquitination, nitrosylation, methylation, acetylation, lipidation and proteolysis and influence almost all aspects of normal cell biology and pathogenesis. Therefore, identifying and understanding PTMs is critical in the study of cell biology and disease treatment and prevention [124; 125].

As noted above, the large number of different PTMs precludes a thorough review of all possible protein modifications. Therefore, this overview only touches on a small number of the most common types of PTMs studied in protein research today. For this list, we consulted the work of Brenner et al.; Eckenhoff and Dmochowski; Khoury et al.; Lodish et al.; Saraswathy and Ramalingam; Yang and Seto.

- **Phosphorylation**: Reversible protein phosphorylation, principally on serine, threonine or tyrosine residues, is one of the most important and well-studied modifications on a small number of the most common types of PTMs. Phosphorylation plays critical roles in the regulation of many cellular processes, including cell cycle, growth, apoptosis and signal transduction pathways [124].

- **Glycosylation**: Protein glycosylation is acknowledged as one of the major PTMs, with significant effects on protein folding, conformation, distribution, stability and activity. Glycosylation surrounds a diverse selection of sugar-moiety additions to proteins that ranges from simple monosaccharide modifications of nuclear transcription factors to highly complex branched polysaccharide changes of cell surface receptors [33; 124]. Glycopeptide bonds can be categorized into specific groups based on the nature of the sugar–peptide bond and the oligosaccharide attached, including $N^-$, $O^-$ and C-linked glycosylation, glypiation and phosphoglycosylation [33; 124].

[22] Post-Translational Modifications

- **Ubiquitination**: Ubiquitin is an $8 - kda$ polypeptide consisting of 76 amino acids that is appended to the $-NH_2$ of lysine in target proteins via the **C**-terminal glycine of ubiquitin [18; 124].

- **S-nitrosylation** Nitric oxide (NO) is produced by three isoforms of nitric oxide synthase (NOS), and it is a chemical messenger that reacts with free cysteine residues to form **S**-nitrothiols (SNOs). S-nitrosylation is a critical PTM used by cells to stabilize proteins, regulate gene expression and provide NO donors. The generation, localization, activation and catabolism of SNOs are tightly regulated [2; 79].

- **Methylation** The transfer of one-carbon methyl groups to nitrogen or oxygen (N- and O-methylation, respectively) to amino acid side chains increases the hydrophobicity of the protein and can neutralize a negative amino acid charge when bound to carboxylic acids. Methylation is mediated by methyltransferases, and S-adenosyl methionine (SAM) is the primary methyl group donor. Methylation occurs so often that SAM has been suggested to be the most used substrate in enzymatic reactions after ATP [2; 79].

- **N-acetylation** or the transfer of an acetyl group to nitrogen, occurs in almost all eukaryotic proteins through both irreversible and reversible mechanisms. Eckenhoff and Dmochowski mentioned that **N**-terminal acetylation requires the cleavage of the **N**-terminal methionine by methionine aminopeptidase (MAP) before replacing the amino acid with an acetyl group from acetyl-CoA by **N**-acetyltransferase (NAT) enzymes. This type of acetylation is co-translational, in that **N**-terminus is acetylated on growing polypeptide chains that are still attached to the ribosome. While 80 to 90% of eukaryotic proteins are acetylated in this manner, the exact biological significance is still unclear [2; 33].

- **Lipidation**: is a method to target proteins to membranes in organelles [23], vesicles [24] and the plasma membrane. The four types of lipidation are:

  - **C**-terminal glycosyl phosphatidylinositol (GPI) anchor
  - **N**-terminal myristoylation
  - **S**-myristoylation
  - **S**-prenylation

  Each type of modification gives proteins distinct membrane affinities, although all types of lipidation increase the hydrophobicity of a protein and thus its affinity for membranes. The different types of lipidation are also not mutually exclusive, in that two or more lipids can be attached to a given protein [2; 87].

- **Proteolysis** Peptide bonds are indefinitely stable under physiological conditions, and therefore cells require some mechanism to break these bonds. Proteases comprise a family of enzymes that cleave the peptide bonds of proteins and are critical in antigen processing, apoptosis, surface protein shedding and cell signaling as mentioned by Brenner et al.. Proteolysis is a thermodynamically favorable and irreversible reaction. Therefore, protease activity is tightly regulated to avoid uncontrolled proteolysis through temporal and/or spatial control mechanisms including regulation by cleavage in cis or trans and compartmentalization [25] [17; 87]. The family of proteases can be classified by the site of action [26] which cleave at the amino or carboxyl terminus of a protein, respectively. Another type of classification is based on the active site groups of a given protease that are involved in proteolysis. Based on this classification strategy, greater than 90% of known proteases fall into one of four

[23] e.g., endoplasmic reticulum, Golgi apparatus, mitochondria.
[24] e.g., endosomes, lysosomes.
[25] e.g., proteasomes, lysosomes
[26] e.g, aminopeptidases and carboxypeptidase.

categories as follows [17; 87]:

- Serine proteases
- Cysteine proteases
- Aspartic acid proteases
- Zinc metalloproteases

These are some of the list of known PTMs. We suggest the reader to read [17; 78; 87; 164].

### 2.1.5. Proteogenomics

He et al. explains that *proteomics* is the comprehensive, integrative study of proteins and their biological functions. The goal of *proteomics* is often to produce a complete and quantitative map of the *proteome* [27] of a species, including defining protein cellular localisation, reconstructing their interaction networks and complexes, and delineating signalling pathways and regulatory post-translational protein modifications [60; 108].

The problem with the *proteomics* assumption is that many peptides are not present in a reference protein sequence database, or, inside any reference database. Peptides may contain mutations and may represent novel protein-coding *loci* and alternative splice forms [60; 108]. One example of this problem is the Telenti et al. project. This project found that each genome carries in average **0.7 Mb** of sequence that is not found in the main build of the HG38 [146]. Another source of Mutations is cancer. Cancer cells usually contain mutations that are hard to identify in a normal *proteomic* approach.

In Proteogenomics, **customised** protein sequence databases generated using genomic and *transcriptomic* [28] information are used to help identify novel peptides

[27] A proteome is the complete set of proteins expressed by an organism [130].

[28] Transcriptomics is the study of the transcriptome—the complete set of RNA[29] transcripts that are produced by the genome, under specific circumstances or in a specific cell—using high-throughput methods, such as microarray analysis [30; 71; 148].

(not present in reference protein sequence databases) from MS-based proteomic data [60; 115].

He et al. in recent years, owing to the emergence of next generation sequencing technologies such as RNA-Seq and dramatic improvements in the depths and throughput of MS-based *proteomics*, the pace of Proteogenomics research has greatly accelerated.

### 2.1.6. Mass Spectrometry

MS is an analytic technique applied to many fields such as pollution control, food control, forensic science, natural products or process monitoring. Other applications include Atomic Physics, Reaction Physics, Reaction Kinetics, Geo-chronology, Inorganic Chemical analysis, ion-molecule reactions, determination of thermodynamic parameters, and many others [61].

A mass spectrometer is composed of three main components: The ion source, the mass analyser, and the detector. Each one of these components corresponds to a step of the whole process (figure 2.1). Every step will be explained in detail in the following sections.

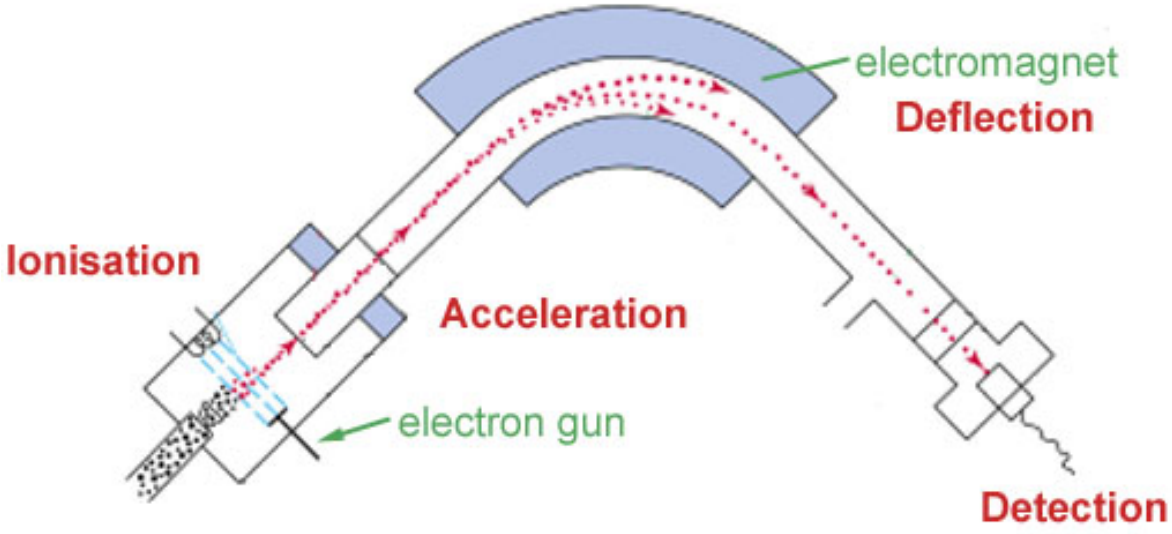


**Figure 2.1:** Mass Spectrometer Diagram

#### 2.1.6.1. Mass Spectrometry Technique

The first step in the MS analysis is the production of gas-phase ions of the sample, for example by electron ionisation [61]. This molecular ion normally undergoes fragmentation. Because it is a radical cation with an odd number of electrons, it can fragment to give either a radical and an ion with an even number of electrons, or a


**Figure 2.2:** Orbitrap Mass Spectrometer

molecule and a new radical cation. [61]

According to Hoffmann and Stroobant, these two types of ions have different chemical properties. Each primary product ion derived from the molecular ion can, in turn, undergo fragmentation, and so on. Mass analysers separate the ions according to their mass-to-charge ratio [61]. Please see Figure 2.3, Mass Spectrometry Example.

In 2.3 there is an example of the result of a MS. This research was intended to focus on the results produced by an ORBITRAP mass spectrometer [129], please see a picture in figure 2.2; but, thanks to the approach followed, others mass spectrometer can be considered.

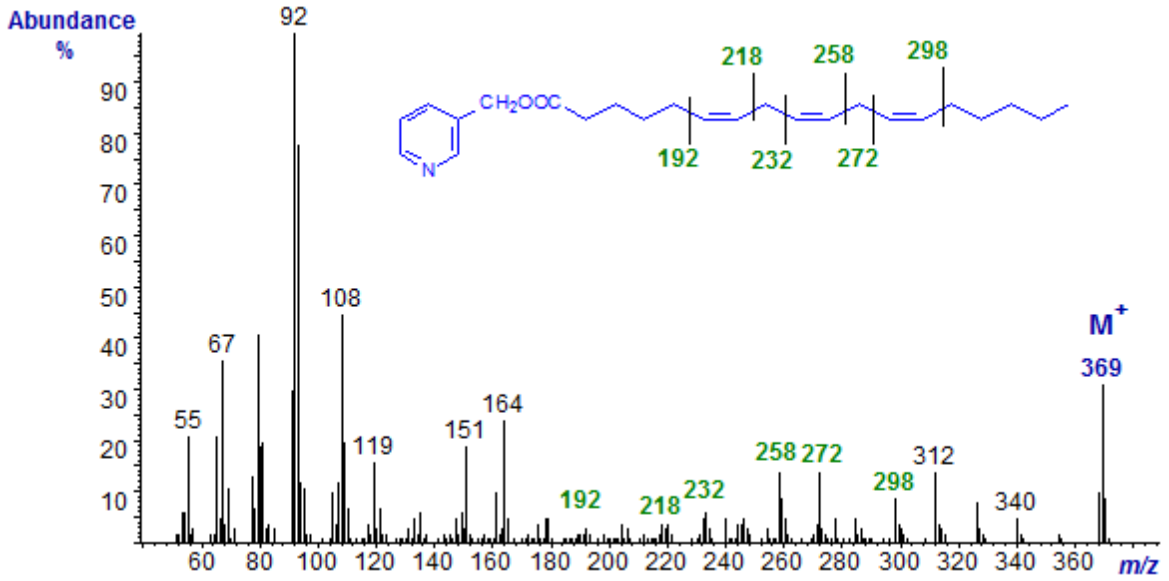


**Figure 2.3:** Mass Spectrometry Example

### 2.1.7. High-resolution Isoelectric Focusing Liquid Chromatography Mass Spectrometry

Branca et al. presented a LC-MS[30]-based method permitting unbiased genome-wide discovery of protein-coding *loci* in higher eukaryotes. Branca et al. used HiRIEF at the peptide level in the $[3.7, 5.0]$ pH[31] range and accurate peptide pI[32] prediction [15] in combination with experimental prefractionation based on peptide pI and theoretical prediction of pI to achieve search-space reduction. Lysates prepared from human A0431 cells and mouse N2A cells were digested with trypsin, and the resulting peptide samples were prefractionated using HiRIEF gel strips, each divided into 72 fractions [15].

Branca et al. evaluated the HiRIEF performance in a conventional search against the reference proteome, identifying 13,078 and 10,637 proteins in human and mouse, respectively [15].

[30]Liquid Chromatography Mass Spectrometry
[31]Hydrogenic Potential
[32]Isolectric/Isoionic point

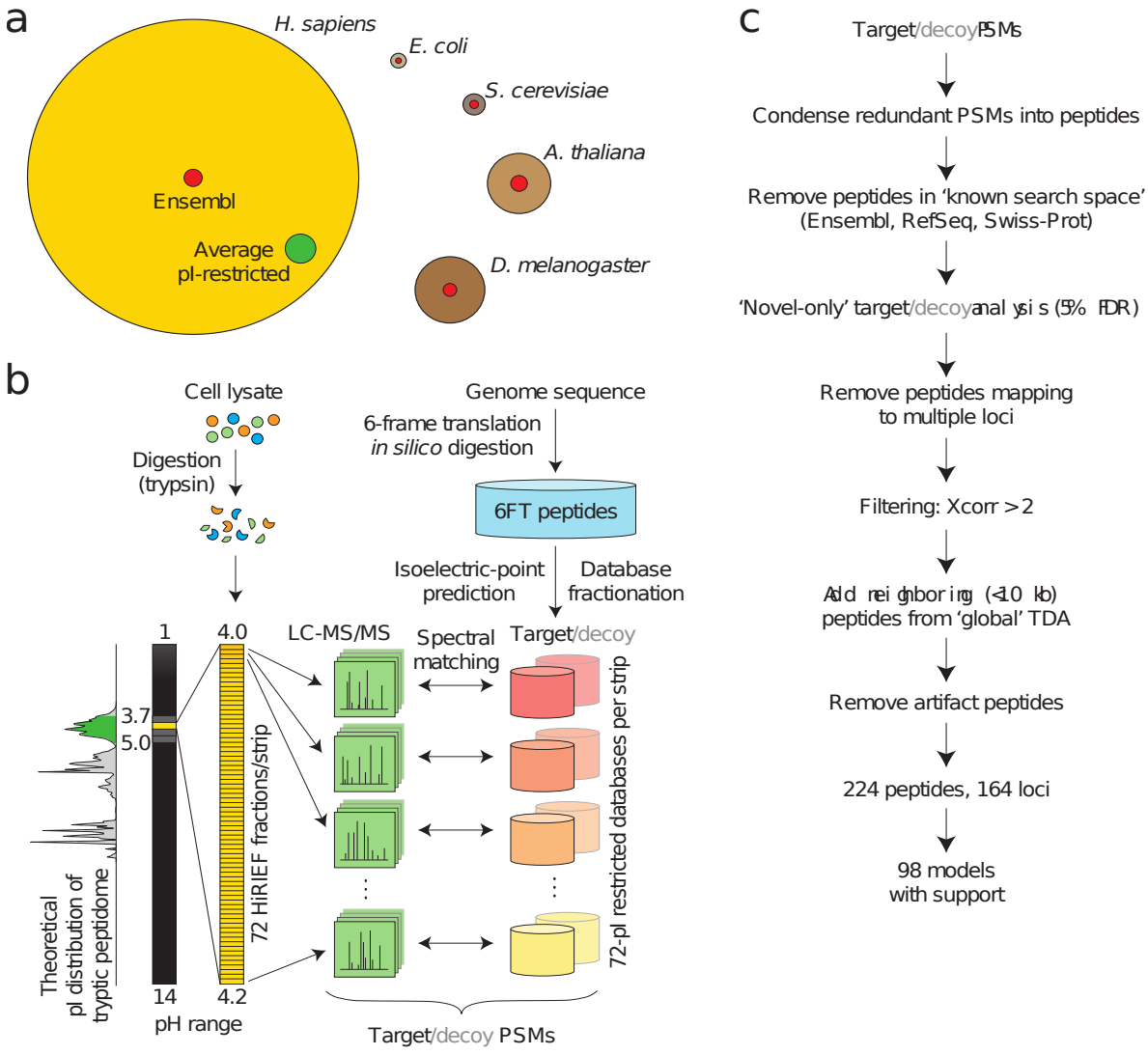


**Figure 2.4:** HiRIEF diagram (taken from [15])

### 2.1.8. Peptide Sequence Problem

The Peptide Sequence Problem is the target of this research. In Peptide Fragmentation we explain how the chemical process works, and in Section 2.1.8.2, Definition we address the problem in a formal way. We present some others *de novo* solutions in Section 2.1.8.2, Definition, and Section 2.1.8.3, Spectrum Graphs.

Important note:

The Peptide sequence problem is the target of this research.

#### 2.1.8.1. Peptide Fragmentation

An amino acid unit in a peptide is called *residue* [48; 73] . When forming the peptides bonds, an ionised amino acid molecule loses an Oxygen and two Hydrogen atoms. The result of this process is that the mass of the ionised molecule have $18da$ [33] more than the *residue* [23]. The structure of both structures is shown in Figure 2.6, Peptide Residues. In Figure 2.5, Peptide Fragmentation, we present how the peptide fragmentation takes place. The red section corresponds to a **C-terminal** ion because the last atom is a C and the blue section corresponds to a **N-terminal** because the first atom is a N.

These are the generally accepted rules for MS/MS sequencing. For further reference, please go t [66].

**Loss of Ammonia and Water**:

1. **Loss of Ammonia**: $y$ and $b$ ion fragments containing the amino acid residues `R`, `K`, `Q`, and `N` may appear to lose ammonia ($NH_3$, $-17da$) [66].

2. **Loss of Water**: $y$ and $b$ ion fragments containing the amino acid residues S, T, and E may appear to lose water ($H_2O$, $-18da$). In the case of Amino Acids, E must be at the **N-terminus** of the fragment for this observation to be made [66].

[33] Atomic mass unit (**amu**; **dalton**): A unit of mass equal to exactly $\frac{1}{12}$ the mass of one `C` atom, or $1.660538921 \times 10^{-27} kg$

| Ion type | Neutral $M_r$ | Terminal |
|---|---|---|
| a | [N]+[M]-CHO | N |
| a* | a-$NH_3$ | N |
| a° | a-$H_2O$ | N |
| **b** | **[N]+[M]-H** | **N** |
| b* | b-$NH_3$ | N |
| b° | b-$H_2O$ | N |
| c | [N]+[M]+$NH_2$ | N |
| d | a - partial side chain | N |
| v | y - complete side chain | C |
| w | z - partial side chain | C |
| x | [C]+[M]+CO-H | C |
| **y** | **[C]+[M]+H** | **C** |
| y* | y-$NH_3$ | C |
| y° | y-$H_2O$ | C |
| z | [C]+[M]-$NH_2$ | C |

**Table 2.3:** Peptide Fragmentation Ion types
**[N]** is the molecular mass of the neutral **N-terminal** group, **[C]** is the molecular mass of the neutral **C-terminal** group, **[M]** is molecular mass of the neutral amino acid residues. To obtain $\mathbf{m/z}$ values, add or subtract protons as required to obtain the required charge and divide by the number of charges. The original table was available at [92].

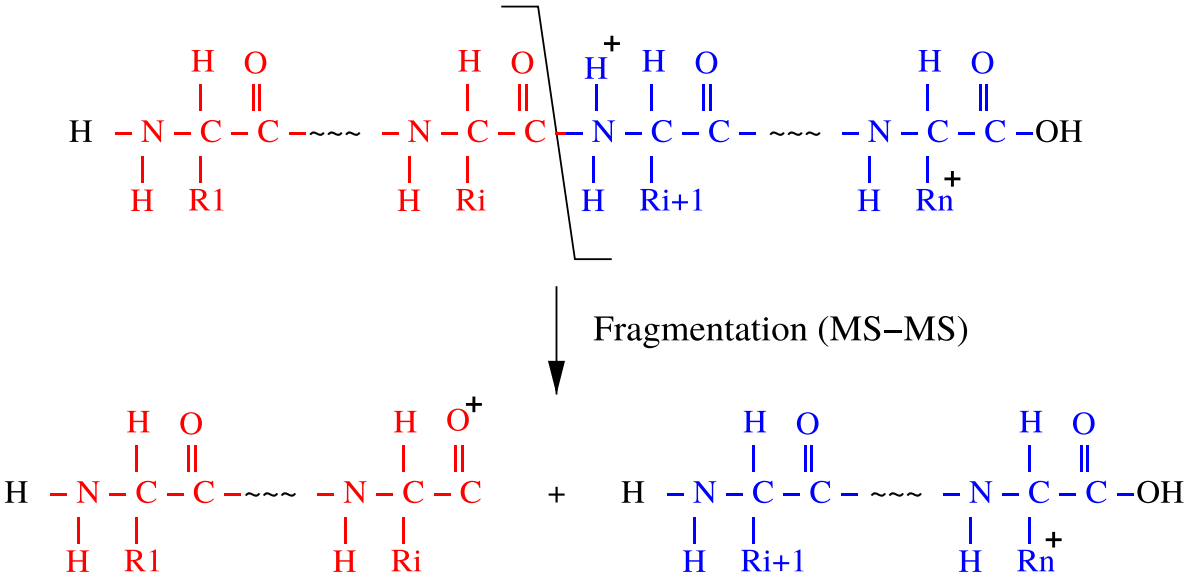


**Figure 2.5:** Peptide Fragmentation (from [23])

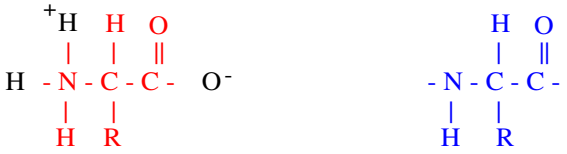


**Figure 2.6:** Peptide Residues: (a) is the ionised amino acid and (b) is the *residue*(from [23])

**Spectral Intensity Rules**:

1. *b* ion intensity will drop when the next residue is `P`, `G` or also `H`, `K`, and `R` [66].

2. Internal cleavages can occur at `P` and `H` residues. An internal cleavage fragment is a fragment that appears to be a shortened peptide with `P` and or `H` at its amino terminus; e.g. the peptide `EFGLPGLQNK` may display the *b* ions `PGLQNK`, `PGLQN`, `PGLQ`, etc. These are the result of a double cleavage event. The *y* ion intensity will often be the most prominent peak in the spectrum [66].

3. It is common for *b* and *y* ions or *y* and *b* ions to swap intensity when a P is encountered in a sequence. This can also be true when the basic residues H, K,

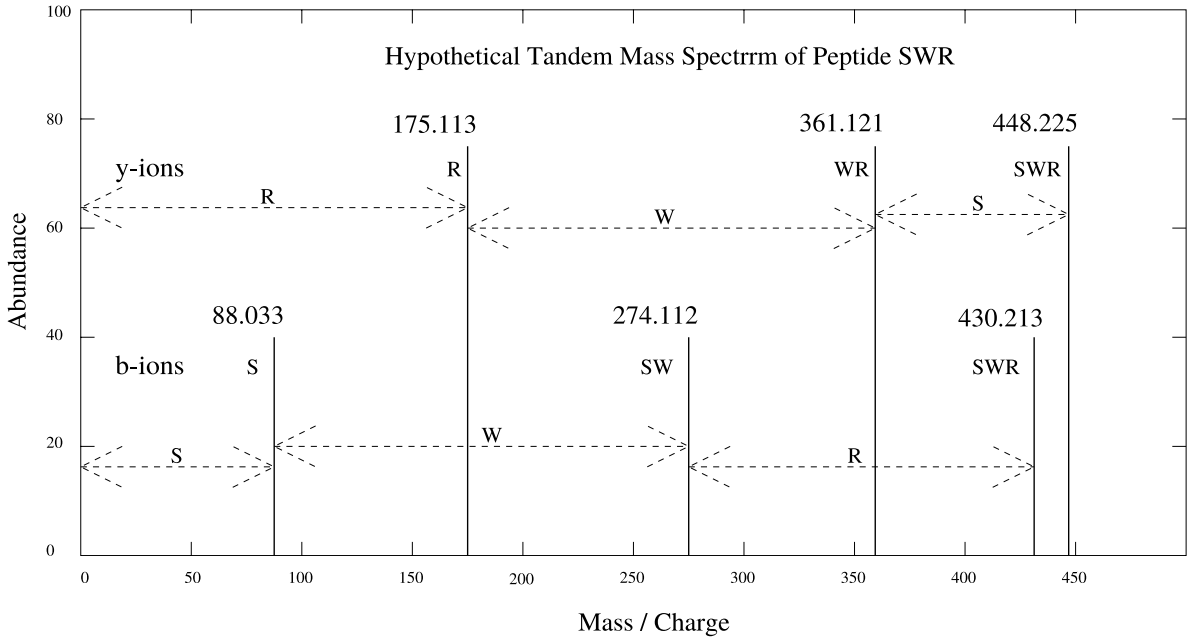


**Figure 2.7:** Peptide SWR tandem mass spectrometry (from [23])

or R are encountered in the sequence [66].

4. When a cleavage appears before or after R, the $-17da$ (loss of ammonia) peak can be more prominent than the corresponding $y$ or $b$ ion [66].

5. When encountering Amino Acids in a sequence, the ion series can die out [66].

**Amino Acid Composition**:

1. It is possible to observe immonium ions at the low end of the spectrum that can give a clue to the amino acid composition of a peptide. One caveat is that if you do not see an immonium ion for an amino acid, this does not mean that that amino acid is absent from the sequence [66].

**Isobaric Mass**:

1. Amino Acids and Amino Acids have isobaric masses [34] [35] and cannot be

[34] Two (or more) ions are isobaric if they have the same nominal mass but different exact mass. For example positive radical ions of $N_2$, CO, and $C_2H_4$ all have nominal mass of $28da$. However, their exact (monoisotopic) masses are $28.00559da$, $27.99436da$, and $28.03075da$ units respectively [163].

[35] Monoisotopic, A spectrum containing only ions made up of the principal isotopes of atoms making up the original molecule [68].

differentiated in a low energy collision. When we see this mass difference in a spectrum, we will label it X or Lxx, adopting the Hunt [62] nomenclature [66].

2. Amino Acids and Amino Acids have near isobaric masses, 128.09496$da$ and 128.05858$da$ respectively. The delta mass is 0.03638$da$; this difference can be used to differentiate K from Q on a mass spectrometer capable of higher mass accuracy and resolution, such as a `q-TOF` mass spectrometer. Usually triple quadrupole or ion trap mass spectrometers are incapable of this feat. On a lower mass accuracy mass spectrometers an acetylation can be performed to shift the mass of lysine by 42$da$. It can be assumed that a 128$da$ mass shift internally on a tryptic peptide is a Glutamine unless followed by a Proline or sometimes Aspartic Acid [66].

3. There are instances where two residues will nearly equal the mass of a single residue, or a modified residue will nearly equal the mass of another amino acid [66].

**More Rules**:

1. When starting a *de novo* sequencing project, start at the high mass end of the spectrum; the lower number of peaks at this end often makes it easier to start sequencing.

2. The region 60$u$ below the parent mass can be confounded by multiple water and ammonia losses, be careful. Realise that Amino Acids may be the first amino acid and may fall in this region [66].

3. Looking for the diagnostic $y_1$ ions at the low end of the spectrum, you may observe 147$da$ for K or 175$da$ for R will help to know if the tryptic peptide ends in a K or an R [66].

4. The $b_1$ fragment is seldom observed making it difficult to determine the order of the first two **N-terminal** amino acids in a peptide sequence. Solutions for this problem can include a one step Edman degradation or an acetylation [66].

#### 2.1.8.2. Definition

Bandeira et al. [8] presented the **Peptide sequence problem**:

**Definition 2.1.8.1.** *Bandeira et al. Spectrum:*

$$s = s_1, s_2, s_3, \ldots, s_n \text{ where } s_i \in 2 \ \forall 1 \leq i \leq n \tag{2.1}$$

**Definition 2.1.8.2.** *Bandeira et al. Peptide:*

$$\pi = \pi_1, \pi_2, \pi_3, \ldots, \pi_n \text{ where } s_i \in 2 \ \forall 1 \leq i \leq n \tag{2.2}$$

**Definition 2.1.8.3.** *Bandeira et al. Probability of* peptide $\pi$ *generating* spectrum $s$*:*

$$P(s|\pi) = \prod_{i=1}^{n} P(s_i|\pi_i) \text{ where } P(x|y) \text{ is a } 2 \times 2 \text{ matrix} \tag{2.3}$$

#### 2.1.8.3. Spectrum Graphs

Important note:

Spectrum Graphs is one technique used to solve approach the Peptide sequence problem.

**Definition 2.1.8.4.** *NC-spectrum graph:*

*Given the mass $W$ of a target peptide sequence $P$, $k$ ions $I_1, \ldots, I_k$ of $P$, and, the masses $w_1, \ldots, w_k$ of those ions. Chen et al. presented the* NC-spectrum graph $G = (V, E)$*. For each $I_j$ is not know if it is a **N-terminal** or a **C-terminal**. If $I_j$ is **C-terminal** ion, it has a complementary $I_j^C$ **N-terminal** ion with $W - (w_j - 2)$,*

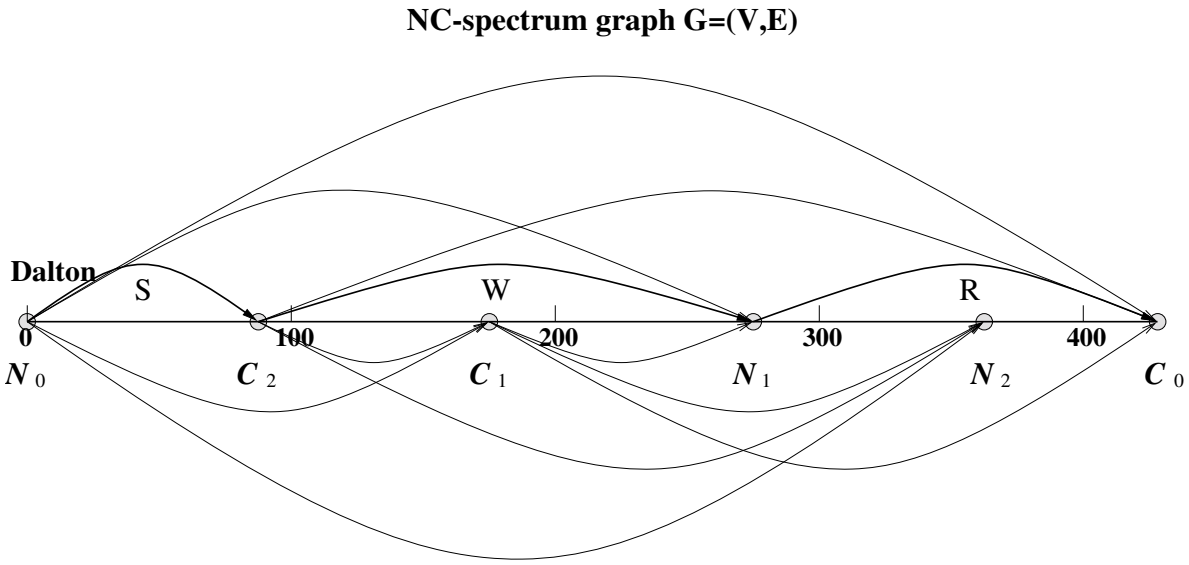


**Figure 2.8:** SWR Spectrum Graph (from [23])

*where the 2 Dalton comes from the 2 extra Hydrogen atoms of the* ***y-ion****. The nodes* $N_j$ *and* $C_j$ *are added to* $V$ *to represent the* $I_j$ *and* $I_j^C$ *where one of them is* ***N-terminal*** *and the other one is* ***C-terminal****. Also nodes* $N_0$ *and* $C_0$ *area added to* $V$ *to denote the whole peptide mass.*

$$V = \{N_0, N_1, \ldots, N_k, C_0, C_1, \ldots, C_k\} \tag{2.4}$$

*Every node of* $v \in V$ *is placed in the real line using* $cord : V \to \mathbb{R}$ *function defined as follows:*

$$cord(v) = \begin{cases} 0 & \text{if } v = N_0 \\ W - 18 & \text{if } v = C_0 \\ w_j - 1 & \text{if } v = N_j \quad 1 \leq j \leq k, j \in \mathbb{N} \\ W - w_j + 1 & \text{if } v = C_j \quad 1 \leq j \leq k, j \in \mathbb{N} \end{cases} \tag{2.5}$$

*Chen et al. defined the edges* $E$ *of* $G$ *as follows. For* $x$ *and* $y \in V$*, there is a* ***directed*** *edge from* $x$ *to* $y$ *denoted by* $(x, y)$ *and* $E(x, y) = 1$*, if the following conditions are satisfied:*

$$(x,y) \in E \text{ if } \begin{cases} x < y \\ x \text{ and } y \text{ are not been generated by the same } I_j \\ cord(y) - cord(x) \text{ is equals to total mass of an amino acid} \end{cases} \tag{2.6}$$

In Figure 2.8, SWR Peptide Spectrum Graph there is an example where the path $N_0 - C_2 - N_1 - C_0$ contains exactly one of every pair of complementary nodes derived from the same ion, the spectrum corresponds to the original peptide SWR.

#### 2.1.8.4. Peptide Sequence Problem

**Definition 2.1.8.5.** *Chen et al. Edge scoring function: The nodes of $V$ can be listed from left to right according to their coordinates as $x_0, x_1, \ldots, x_k, y_0, y_1, \ldots, y_k$; where $x_i$ and $y_i$ are complementary. But, in practice a MS/MS may contain noise as mass peaks of other types of ions from the same peptide, also mass peaks from unknown ions. Chen et al. proposed a way to deal with this problem by using a predefined edge (and node) function $s(\cdot)$ function such that node corresponding to high peaks and edges labelled with a single amino acid receive a higher scores. And the score of a path is defined as the sum of the scores of the edges (and the nodes) of the given path [23].*

**Definition 2.1.8.6.** *Chen et al. Peptide sequence problem:* For a given NC-Spectrum graph $G = (V, E)$, and a edge scoring function $s(\cdot)$, asks for the maximum score path from $x_0$ to $y_0$, such that at most one of $x_j$ and $y_i$ for every $1 \leq i \leq k$ is on the path.

**Definition 2.1.8.7.** *Chen et al. Mass array: Let $h$ the maximum mass under construction. Let $\delta$ be the measurement precision. A* mass array $\mathcal{A} : \mathbb{R} \to 2$ *is function that returns* 1 *when the given mass corresponds to an amino acid mass.*

$$\mathcal{A}(m) = \begin{cases} 1 & \exists \pi \in \Pi \text{ where } m = mw(\pi) \\ 0 & \text{otherwise} \end{cases} \tag{2.7}$$

**Theorem 2.1.8.1** (Chen et al.: Theorem 1)**.** *The construction of $\mathcal{A}$ will take $O(\frac{h}{\delta})$ time and for a given spectrum of $k$ mass peaks, $G$ can be constructed in $O(k^2)$ time. This theorem was proven by the Theorem 1 of Chen et al. paper [23].*

**Definition 2.1.8.8.** *Chen et al. Ideal peptide sequence problem:* is equivalent to the problem which, given $G = (V, E)$, asks for a directed path from $x_0$ to $y_0$ which contains on of $x_i$ and $y_i$ for each $i > 0$ *[23].*

*Listed the nodes of $G$ from left to right as $x_0, x_1, \ldots, x_k, y_k, \ldots, y_1, y_0$. Let $M(i, j)$ a $(k + 1) \times (k + 1)$ matrix with $0 \leq i, j \leq k$. Let $M(i, j) = 1$ if and only if in $G$, there is a path $L$ from $x_0$ to $x_i$ and a path $R$ frin $y_j$ to $y_0$, such that $L \cup R$ contains exactly onf of $x_p$ and $y_p$ fir every $p \in [1, i] \cup [1, j]$. Denoted the two path $L \cup R$ as $LR$ path for $M(i, j) = 1$. Let $M(i, j) = 0$ otherwise [23].*

$$M(i,j) = \begin{cases} 1 & \exists L = x_0 - \cdots - x_i, R = y_i - \cdots - x_0 \text{ where } \exists! x_p, y_p \forall p \in [1, i] \cup [1, j] \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

**Theorem 2.1.8.2** (Chen et al.: Theorem 2)**.** *Given $G = (V, E)$, the following statements hold:*

1. *Algorithm 1, Chen et al.: Compute-M(G) computes the matrix $M$ in $O(|V|^2)$ time.*
2. *Given M, a feasible solution of $G$ can be found in $O(|V|)$ time.*
3. *A feasible solution of $G$ can be found in $O(|V|^2)$ time and $O(|V|^2)$ space.*

**Algorithm 1** Chen et al.: Compute-M(G)

```
1: procedure COMPUTE-M(G)
2:     M(0,0) ← 1
3:     M(i,j) ← 0 ∀i ≠ 0 or j ≠ 0
4:     for j = 2 to k do
5:         for i = 0 to j − 2 do
6:             if M(i, j−1) = 1 and E(x_i, x_j) = 1 then
7:                 M(j, j−1) ← 1
8:             if M(i, j−1) = 1 and E(y_j, y_{j−1}) = 1 then
9:                 M(i, j) ← 1
10:            if M(j−1, i) = 1 and E(x_{j−1}, x_j) = 1 then
11:                M(j, i) ← 1
12:            if M(j−1, i) = 1 and E(y_j, y_i) = 1 then
13:                M(j−1, i) ← 1
```

*4. All feasible solutions of $G$ can be found in $O(|V|^2+n|V|)$ time and $O(|V|^2+n|V|)$ space; where $n$ is the number of solutions.*

Also Chen et al. proposed a method to reduce the time and space complexities of Algorithm 1, Chen et al.: Compute-M(G). $M$ can be encoded in two linear arrays

**Definition 2.1.8.9** (Chen et al.: *cross edges* and *inside edges*)**.** *Define an edge $(x_i, y_j)$ with $0 \leq i, j \leq k$ to be a* cross edge*, and edge $x_i, x_j$ or $(y_i, y_j)$ with $0 \leq i < j \leq k$ to be a* inside edge*. Let $lce(z)$ ($lce : V \to \mathbb{N}$) be the length of the **longest** consecutive* inside edges *starting from node $z$, i. e.,*

$$\begin{cases} lce(x_i) = j - i \quad \text{if } E(x_i, x_{i+1}) = \cdots = E(x_{j-1}, x_j) = 1 \text{ and } (j = k \text{ or } E(x_j, x_{j+1}) = 0) \\ lce(j_j) = j - i \quad \text{if } E(y_j, y_{j-1}) = \cdots = E(j_{j+1}, y_i) = 1 \text{ and } (i = 0 \text{ or } E(y_i, x_{i-1}) = 0) \end{cases} \tag{2.9}$$

*Let $dia(z)$ be two diagonals in $M$ where:*

$$dia(z) = \begin{cases} M(j, j-1) & \textbf{\textit{for}}\ 0 < j < k\ \textit{if}\ z = x_j \\ M(j-1, j) & \textbf{\textit{for}}\ 0 < j < k\ \textit{if}\ z = y_j \\ 1 & \textit{if}\ z = x_0 \\ 1 & \textit{if}\ z = y_0 \end{cases} \tag{2.10}$$

**Lemma 2.1.8.1** (Chen et al.: Lemma 3)**.** *Given $lce(\cdot)$ and $dia(\cdot)$ any entry of $M$ can be computed in $O(1)$ time.*

**Lemma 2.1.8.2** (Chen et al.: Lemma 4)**.** *Given $G = (V, E)$, $lce(\cdot)$ and $dia(\cdot)$ can be computed in $O(|V| + |E|)$ time and $O(n|V|)$ space; where $n$ is the number of solutions.*

**Theorem 2.1.8.3** (Chen et al.: Theorem 5)**.** *Given $G = (V, E)$, the following statements hold:*

1. *A feasible solution of $G$ can be found in $O(|V| + |E|)$ time and $O(|V|)$ space.*
2. *All feasible solutions of $C$ in can be found in $O(n|V|+|E|)$ time and $O(|V|)$ space; where $n$ is the number of solutions.*

Let $Q(i, j)$ a $(k + 1) \times (k + 1)$ matrix with $0 \leq i, j \leq k$. Let $Q(i, j) = 1$ if and only if in $G$, there is a path $L$ from $x_0$ to $x_i$ and a path $R$ from $y_j$ to $y_0$, such that $L \cup R$ contains exactly one of $x_p$ and $y_p$ for every $p \in [1, i] \cup [1, j]$. Denoted the two path $L \cup R$ as $LR$ path for $M(i, j) = 1$. Let $Q(i, j) = 0$ otherwise [23]. If $Q(i, j) > 0, Q(i, j) = max\{s(L) + s(R)\}$, the maximum score among all $R$ and $L$ pairs.

**Theorem 2.1.8.4** (Chen et al.: Theorem 6)**.** *Given $G = (V, E)$, the following statements hold:*

1. *Algorithm 2, Chen et al. Compute-Q(G) computes the matrix $Q$ in $O(|V||E|)$ time.*
2. *A feasible solution of $G$ can be found in $O(|V||E|)$ time and $O(|V|^2)$ space.*

**Algorithm 2** Chen et al. Compute-Q(G)

```
1: procedure COMPUTE-Q(G)
2:     Q(i, j) ← 0∀0 ≤ i, j ≤ k
3:     for j = 1 to k do
4:         if E(y_j, y_0) = 1 then
5:             Q(0, j) ← max{Q(0, j), s(y_j, y_0)}
6:         if E(x_0, x_j) = 1 then
7:             Q(j, 0) ← max{Q(j, 0), s(x_0, x_j)}
8:         for i = 1 to j − 1 do
9:             for E(y_j, y_p) = 1 and Q(i, p) > 0 do
10:                Q(i, j) ← max{Q(i, j), Q(i, p) + s(y_j, y_p)}
11:            for E(c_p, x_j) = 1 and Q(p, i) > 0 do
12:                Q(i, j) ← max{Q(j, i), Q(p, i) + s(x_p, x_j)}
```

**Definition 2.1.8.10.** *One-amino acid modification problem: is equivalent to the problem which given $G = (V, E)$, asks for two nodes $v_i$ and $v_j$, such that $E(v_i, v_j) = 0$, but adding the edge to $G$ creates a feasible solution that contains the edge.*

Let $N(i, j)$ a $(k + 1) \times (k + 1)$ matrix with $0 \leq i, j \leq k$. Let $N(i, j) = 1$ if and only if in $G$, there is a path from $x_i$ to $y_j$ which contains exactly one of $x_p$ and $y_p$ for every $p \in [1, i] \cup [1, j]$. Let $N(i, j) = 0$ otherwise [23].

**Algorithm 3** Chen et al. Compute-N(G)

```
1: procedure COMPUTE-N(G)
2:     N(i, j) ← 0∀0 ≤ i, j ≤ k
3:     Compute N(k, k − 1)
4:     Compute N(k − 1, k)
5:     for j = k − 2 to 0 do
6:         for i = k to j + 2 do
7:             if N(i, j + 1) = 1 and E(x_i, x_j) = 1 then
8:                 N(j, j + 1) ← 1
9:             if N(i, j + 1) = 1 and E(y_{j+1}, y_j) = 1 then
10:                N(i, j) ← 1
11:            if N(j + 1, i) = 1 and E(x_j, x_{j+1}) = 1 then
12:                N(j, i) ← 1
13:            if N(j + 1, i) = 1 and E(y_i, y_{j+1}) = 1 then
14:                N(j + 1, j) ← 1
```

**Theorem 2.1.8.5** (Chen et al.: Theorem 7)**.** *Given $G = (V, E)$, the following statements hold:*

1. *Algorithm 3, Chen et al. Compute-N(G) computes the matrix $N$ in $O(|V|^2)$ time.*
2. *All possible amino acid modifications can be found in $O(|V||E|)$ time and $O(|V|^2)$ space.*

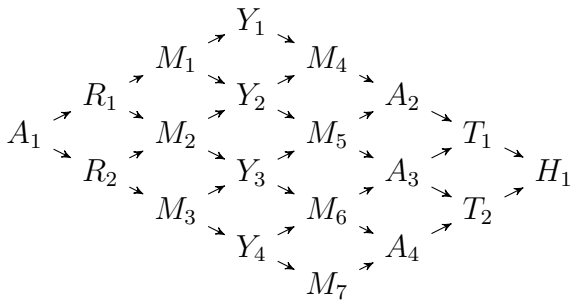


**Figure 2.9:** ARMY MATH *net/lattice*
Original figure was taken from [113]

## 2.2. Mathematical Fundamentals

This particular section is a collection of many fields of Mathematics. Most of these topics are studied as a part of Computer Science and Mathematics college courses. We present Set Theory, Algebraic Structures, and Topology. This section intends is to provide the reader quick access to the definitions all along this document.

### 2.2.1. Set Theory

These are the well known definitions of the Set theory, for further information the reader can refer to [45; 141].

**Definition 2.2.1.1** (*poset*)**.** *Partially ordered set*

*A* poset *is defined as a pair* $(X, \leq)$ *where* $X$ *is a* ***non-empty*** *set and* $\cdot \leq \cdot : X \times X \to 2$ *is a binary relation over* $X$ *that satisfies for all* $a, b, c \in X$*:*

*PO1 Reflexive:* $a \leq a$

*PO2 Anti-symmetry:* $a \leq b \wedge b \leq a \implies a = b$

*PO3 Transitivity:* $a \leq b \wedge b \leq c \implies a \leq c$

**Definition 2.2.1.2** (*net*)**.** *A* net *is a* poset *with an additional axiom for all* $a, b, c \in X$*:*

*N1 Reflexive:* $a \leq a$

*N2 Anti-symmetry:* $a \leq b \wedge b \leq a \implies a = b$

*N3 Transitivity:* $a \leq b \wedge b \leq c \implies a \leq c$

*N4 :* $\exists m \in X \quad m \leq a \wedge m \leq b$

In Figure 2.9, ARMY MATH *net/lattice* there is an example of *net*

**Definition 2.2.1.3** (*ordered set*)**.** *An* ordered set *is a* poset *with an additional axiom for all* $a, b, c \in X$*:*

*O1 Reflexive:* $a \leq a$

*O2 Anti-symmetry:* $a \leq b \wedge b \leq a \implies a = b$

*O3 Transitivity:* $a \leq b \wedge b \leq c \implies a \leq c$

*O4 Totality:* $a \leq b \vee b \leq a$

**Definition 2.2.1.4** (*Subsets*)**.** $2^K$*: The set of all the subsets of a given set* $K$ *is denoted as:*

$$2^K = \{k \text{ where } k \subset K\} \tag{2.11}$$

**Definition 2.2.1.5** (*string*)**.** $\Sigma^*$*: The set of all strings of a given known as the alphabet* $\Sigma$ *is denoted as:*

$$\Sigma^* = \bigcup_{i \in \mathbb{N}_0} \Sigma^i \tag{2.12}$$

$$\Sigma^0 = \{\lambda\} \tag{2.13}$$

$$|s| = R \iff s \in \Sigma^R, \forall R \in \mathbb{N} \tag{2.14}$$

*An element of* $\Sigma^*$ *is known as a* string *of the* $\Sigma$ *alphabet.* $\lambda$ *is a symbol that represents an empty* string*, also* $|\lambda| = 0$.

**Theorem 2.2.1.1** ($(\Sigma^*, \cdot)$ *is a monoid*)**.** *The for any given* ***non-empty*** *set* $\Sigma$*, and the concatenation function is a* monoid*.*

$$\langle ab \rangle_i = \begin{cases} \langle a \rangle_i & i \leq |a| \\ \langle b \rangle_{i-|a|} & i > |a| \end{cases} \quad \forall 1 \leq i \leq |ab| = |a| + |b| \tag{2.15}$$

**Theorem 2.2.1.2** (*lexicographical order*)**.** *for* strings

*For a given* ordered set $(\Sigma, \leq)$*,* $(\Sigma^*, \leq)$ *is a* ordered set*.*

$$a \leq b \iff \exists j \in \mathbb{N} \quad \langle a \rangle_i = \langle b \rangle_j \forall i \leq j \wedge (\langle a \rangle_j \leq \langle b \rangle_j \vee j = |a|) \tag{2.16}$$

**Theorem 2.2.1.3** (*lexicographical order*)**.** *For a given* ordered sets $(A, \leq)$ *and* $(B, \leq)$*;* $(A \times B, \leq)$ *is a* ordered sets*.*

$$(a_1, b_1) \leq (a_1, b_1) \iff \begin{cases} 1 & \neg(b_1 \leq a_1) \\ 0 & \neg(a_1 \leq b_1) \\ 1 & \neg(b_1 \leq a_1) \\ 0 & \neg(a_1 \leq b_1) \\ 0 & \textit{otherwise} \end{cases} \tag{2.17}$$

### 2.2.2. Algebraic Structures

In Algebra, arithmetical operations and formal manipulations are applied to abstract symbols rather than specific numbers [35]. These are the well known concepts of Abstract Algebra. The reader can refer to [76; 114] for further information.

**Definition 2.2.2.1** (*monoid*)**.** *A* monoid *is an algebraic structure with a single associative binary operation and an identity element. For a given set $S$ and a binary operation $\cdot : S \times S \to S$ the pair $(S, \cdot)$ is* monoid *if for all $a, b \in S$ stands:*

*M1 Associative:* $(a \cdot b) \cdot c = a \cdot (b \cdot c)$

*M2 Neutral element:* $\exists e \in S \quad e \cdot a = a = a \cdot e$

**Definition 2.2.2.2** (*commutative monoid*)**.** *A* commutative monoid *is an algebraic structure with a single associative binary operation and an identity element. For a given set $S$ and a binary operation $\cdot : S \times S \to S$ the pair $(S, \cdot)$ is* commutative monoid *if for all $a, b \in S$:*

*CM1 Associative:* $(a \cdot b) \cdot c = a \cdot (b \cdot c)$

*CM2 Commutative:* $a \cdot b = b \cdot a$

*CM3 Neutral element:* $\exists e \quad e \cdot a = a = a \cdot e$

**Definition 2.2.2.3** (*semi ring*)**.** *For a given set $R$ and two binary operation $+ : R \times R \to R$, known as addition, and, $\cdot : R \times R \to R$, known as multiplication, the tuple $(R, +, \cdot)$ is a* semi ring *if for all $a, b, cinR$:*

*SM1 :* $(R, +)$ *is a* commutative monoid *with neutral element* 0

*SM2 :* $(R, \cdot)$ *is a* monoid *with neutral element* 1

*SM3 Multiplication left and right distributes over addition:*

- $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$
- $(b + c) \cdot a = (b \cdot a) + (c \cdot a)$

*SM4 Multiplication by* 0 *annihilates* $R$*:* $0 \cdot a = a \cdot 0 = 0$

**Remark.** *:* $(\mathbb{N}_0, +, \cdot)$ *is a* semi ring*.*

**Definition 2.2.2.4** (*field*)**.** *For a given set* $F$ *and two binary operations* $+ : F \times F \to F$*, known as* addition*, and,* $\cdot : F \times F \to F$*, known as* multiplication*; the tuple* $(F, +, \dot{c})$ *is a field if the the for all* $a, b, c \in F$*:*

*F1 Associativity of addition:* $(a + b) + c = a + (b + c)$

*F2 Associativity of multiplication:* $(a \cdot b) \cdot c = a \cdot (b \cdot c)$

*F3 Commutativity of addition:* $a + b = b + a$

*F4 Commutativity of multiplication:* $a \cdot b = b \cdot a$

*F5 Addition neutral:* $\exists 0 \in F \quad a + 0 = a = 0 + a$

*F6 Multiplication neutral:* $\exists 1 \in F \quad a \cdot 1 = a = 1 \cdot a$

*F7 Addition inverse:* $\exists -a \quad a + (-a) = 0$

*F8 Multiplication inverse:* $\exists a^{-1} \quad a \cdot a^{-1} = 1$

*F9 Distributivity of multiplication over addition:* $a \cdot (b + c) = a \cdot b + a \cdot$

**Definition 2.2.2.5** (*homomorphism*)**.** *If* $f : A \to B$ *is a map between two* $(A, \cdot), (B, \cdot)$ *algebraic structures of the same type that preserves the operations of the structures:*

$$f(\cdot_A(a_1, \ldots, a_k)) = \cdot_B(f(a_1), \ldots, f(a_k)) \quad \forall a_1, \ldots, a_k \in A \text{ where } k = \text{ arity of } \cdot \tag{2.18}$$

**Definition 2.2.2.6** (*generation set*)**.** *For a given algebraic structure $G$, a* generation set *$S \subset G$ is a subset such that every element of $G$ can be expressed as the combination (under the $G$ operations), of finitely many elements of $S$. If $S$ is finite, then a group $G = \langle S \rangle$ is called finitely generated. The elements of $S$ are called* generators.

**Definition 2.2.2.7** (*Cayley Graph*)**.** *Suppose that $G$ $G$ is a group and $S$ is a* generation set. *The* Cayley graph *$\Gamma = (G, S)$ is a coloured directed graph constructed as follows:*

- *Each element $g \in G$ assigned a vertex $g \in V(\Gamma)$.*
- *Each generator $s \in S$ is assigned a colour $c_s$.*
- *For any $g \in G$, $s \in S$, there is an edge $(g, gs) \in E(\Gamma)$ with colour $c_s$.*

*The details of this structure can be found on [20; 21].*

*In Figure 2.10, Example of* Cayley Graph *there is an example of a* Cayley Graph *for a* string *with only 4 characters.*

**Definition 2.2.2.8** (*lattice*)**.** *An algebraic structure $(L, \vee, \wedge)$, consisting of a set $L$ and two binary operations $\vee : L \times L \to 2$, and $\wedge : L \times L \to 2$ with following axiomatic identities holds for all $a, b, c \in L$:*

*LT1 Commutative laws:*

- $a \vee b = b \vee a$
- $a \wedge b = b \wedge a$

*LT2 Associative laws:*

- $a \vee (b \vee c) = (a \vee b) \vee c$
- $a \wedge (b \wedge c) = (a \wedge b) \wedge c$

*LT3 Absorption laws:*

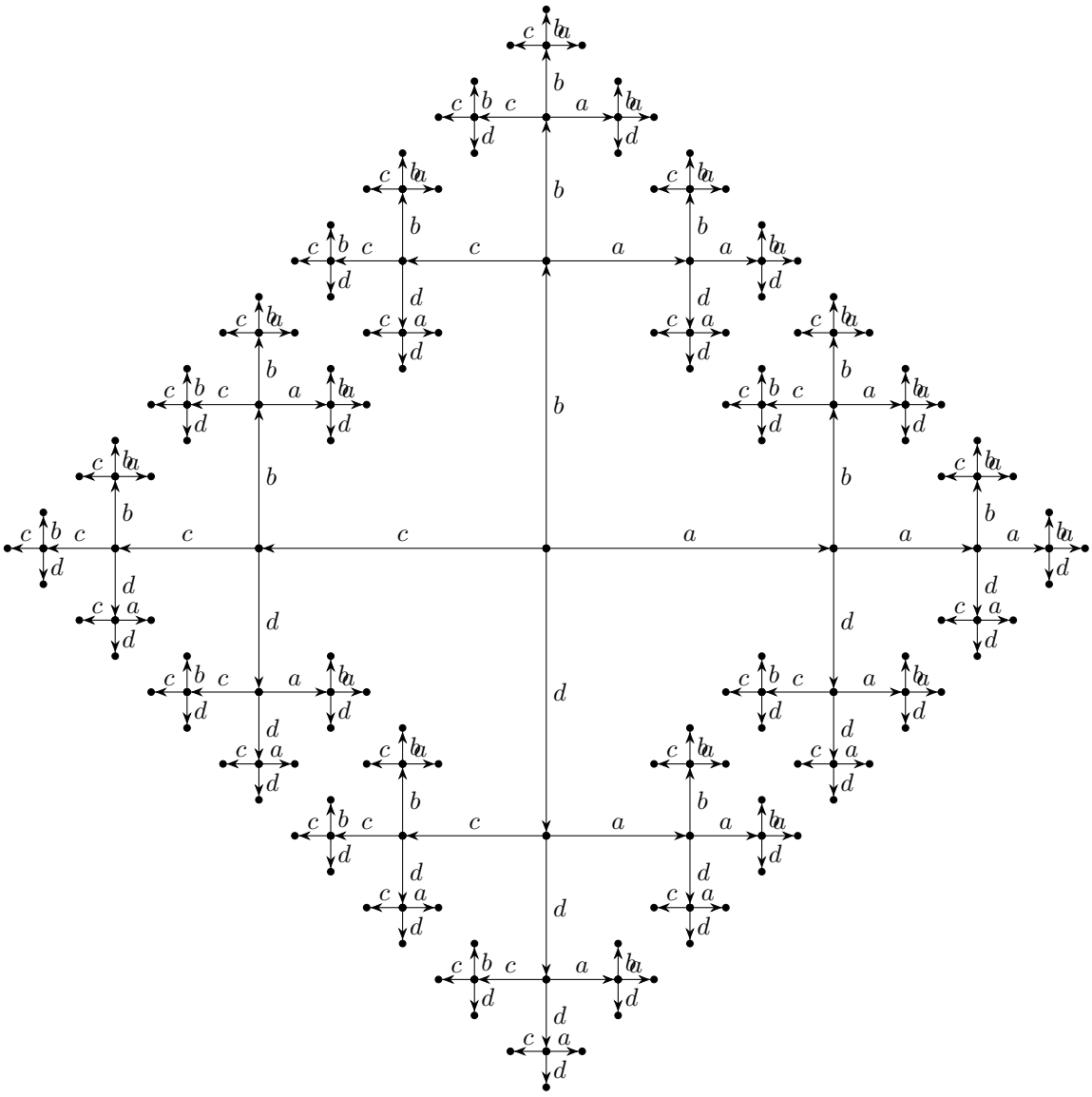

**Figure 2.10:** Example of *Cayley Graph*
For $G = (\Sigma = \{a, b, c, d\}, \rightarrow)$ *string* up to 4 characters of length. This figure was generated using a L-system [86]. In this case the *generation set* is $S = \{\langle a \rangle, \langle b \rangle, \langle c \rangle, \langle d \rangle\}$.

- $a \vee (a \wedge b) = a$
- $a \wedge (a \vee b) = a$

*LT4 Idempotent laws:*

- $a \vee a = a$
- $a \wedge a = a$

*Further material related to lattices can be found on [47; 54].*

**Definition 2.2.2.9** (*bounded lattice*)**.** *A* bounded lattice *is an algebraic structure of the form* $(L, \vee, \wedge, 0, 1)$ *such that* $(L, \vee, \wedge)$ *is a* lattice, $0$ *(the lattice's bottom) is the identity element for the join operation* $\vee$*, and* $1$ *(the lattice's top) is the identity element for the meet operation* $\wedge$*.*

*LT5 Identity laws:*

- $a \vee 0 = a$
- $a \wedge 1 = a$

**Definition 2.2.2.10** (*complemented lattice*)**.** *A* complemented lattice *is a* bounded lattice*, in which every element a has a complement unary operation* $\neg$*, i.e. an element* $\neg a$ *such that:*

*LT6 Complement laws:*

- $a \vee \neg a = 1$
- $a \wedge \neg a = 0$

**Definition 2.2.2.11** (*Boolean space*)**.** *A* Boolean space *is of a* complemented lattice $2 = (\{0, 1\}, \vee, \wedge, 0, 1, \neg)$ *with a distribution law for all* $a, b, c \in 2$*:*

*LT6 Distribution laws:*

- $a \vee (b \wedge c) = (a \wedge b) \vee (a \wedge c)$
- $a \wedge (b \vee c) = (a \vee b) \wedge (a \vee c)$

**Definition 2.2.2.12** (*Vector space*)**.** $F^R$*: For a given field* $(F, +, \cdot)$ *known as the* field *and a natural number* $R \in \mathbb{N}$ *known as the* rank*:*

$$F^R = \begin{cases} F & R = 1 \\ F \times F^{R-1} & R > 1 \end{cases} \tag{2.19}$$

**Definition 2.2.2.13** ($\vec{0}$)**.** zero *of a* Vector space*: for a given* $F^R$ Vector space *on the field* $(F, +, \cdot)$*:*

$$\langle \vec{0} \rangle_j = neutral(+) \quad \forall j \in \mathbb{N}, 1 \leq j \leq R \tag{2.20}$$

**Definition 2.2.2.14** (The $i$nth *basis* of a *Vector space*)**.** $\vec{1_i}$*: for a given* $F^R$ Vector space *on the field* $(F, +, \cdot)$*,* $i \in \mathbb{N}, i \leq R$*:*

$$\langle \vec{1_i} \rangle_j = \begin{cases} neutral(+) & i \neq j \\ neutral(\cdot) & i = j \end{cases} \quad \forall j \in \mathbb{N}, 1 \leq j \leq R \tag{2.21}$$

**Definition 2.2.2.15** (The *basis* of a *Vector space*)**.** $\tilde{1}$*: for a given* $F^R$ Vector space *on the field* $(F, +, \cdot)$*:*

$$\tilde{1} = \{\vec{1_i} \quad \forall 1 \leq i \leq R\} \tag{2.22}$$

### 2.2.3. Topology

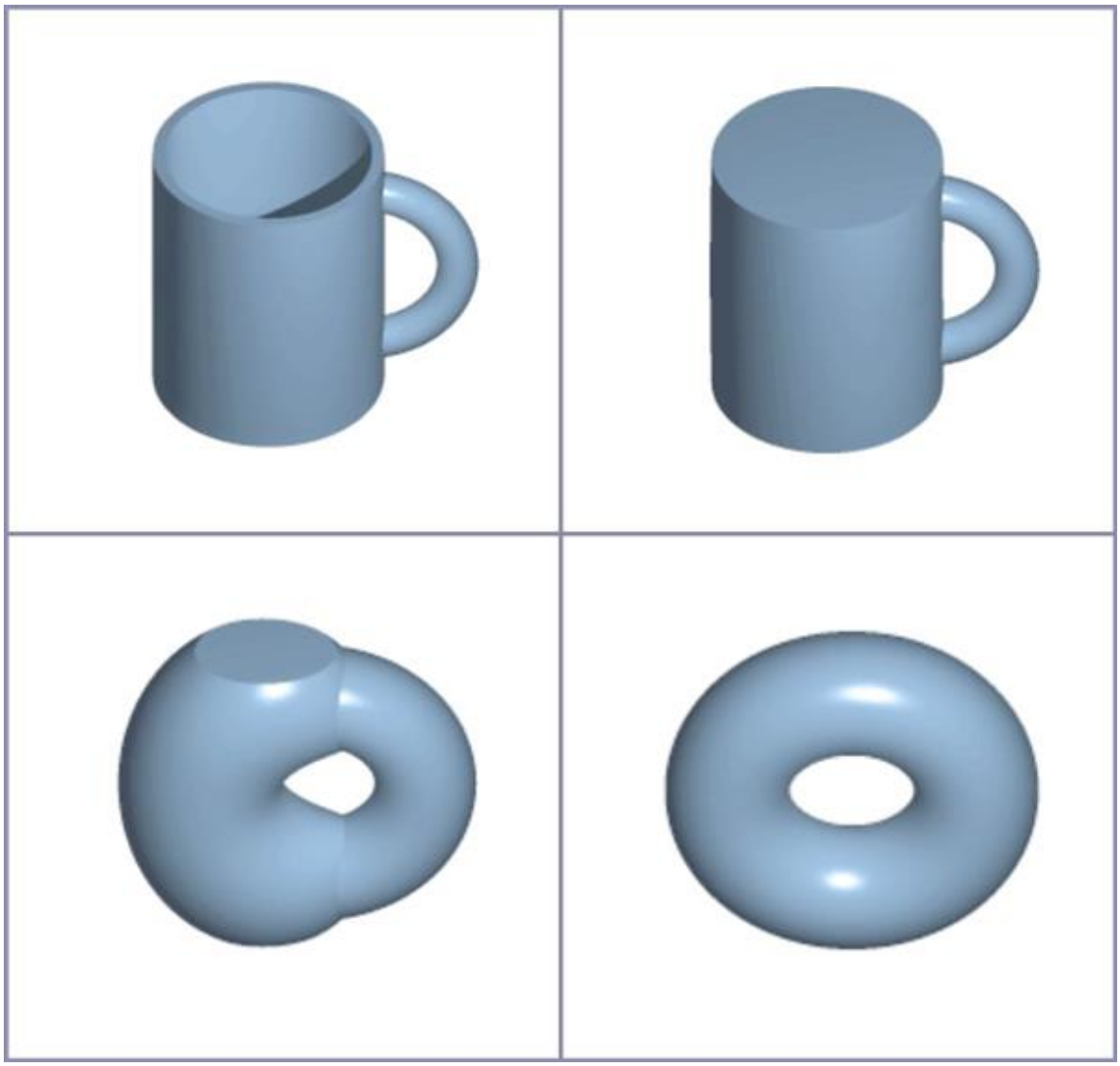

**Figure 2.11:** Example of Continuous Transformations
The classical example of Topology is the transformation of a mug in a toroid. Image taken from [162].

According to the Encyclopaedia Britannica Inc., the main topics of interest in Topology are the properties that remain unchanged by such continuous deformations. Topology studies how two objects are considered equivalent if they can be continuously deformed into one another through such motions in space as twisting, stretching, bending, and shrinking while disallowing tearing apart or gluing together

parts [41]. An example of this can be observed in Figure 2.11, Example of Continuous Transformations, where a mug can be deformed into a toroid [36].

Important note:

The main topics of interest in Topology are the properties that remain unchanged by such continuous deformations

These are well known definitions and theorems of Topology Theory; most of these definitions and theorems are referenced in [29]. For further information the reader can refer to [70; 152; 153; 154; 155; 156].

**Definition 2.2.3.1** (*Topological space*)**.** *A topological space is an ordered pair* $(X, \tau)$*, where* $X$ *is a set and* $\tau \subset 2^X$ *is a collection of subsets of* $X$*, satisfying the following axioms:*

1. $X \in \tau$ *and* $\emptyset \in \tau$*;*
2. *For any index* $I$ *over* $\tau$*:* $\cup_{i \in I} U_i \in \tau$*;*
3. *For any* $U, V \in \tau$ *the intersection* $U \cap V \in \tau$

$\tau$ is known as the *open set*s of $X$.

There is particular kind of *Topological space*, the *metric space*s; where the *open set*s are defined as follows.

**Definition 2.2.3.2** (*metric space*)**.** *A non-empty set* $X$ *with a map* $d : X \times X \to R$ *is called a metric space if the map* $d$ *has the following properties [29]:*

$MS1 : d(x, y) \geq 0 | \forall x \in X \forall y \in X$

$MS2 : d(x, y) = 0 \iff x = y$

$MS3 : d(x, y) = d(y, x) \forall x \in X \forall y \in X$

[36] A surface of revolution obtained by rotating a closed plane curve about an axis parallel to the plane which does not intersect the curve. The simplest toroid is the torus [159].

*MS4* : $d(x,y) \geq d(x,z) + d(z,y) | \forall x \in X \forall y \in X \forall \in X$

The map $d$ is called the metric on $X$ or sometimes the **distance function** on $X$. The phrase "$(X,d)$ is a *metric space*" means that $d$ is a metric on the set X. Property MS4 is often called the **triangle inequality** [29; 133].

**Definition 2.2.3.3** (*distance defined by a function*)**.** *For a given function $f : X \to \mathbb{R}$, and $i \in \mathbb{N}$ let the $d_f : X \times X \to \mathbb{R}_0^+$ the* distance defined by *$f$ as:*

$$d_f(a,b) = |f(a) - f(b)| \tag{2.23}$$

**Definition 2.2.3.4** (*equivalent class of a function*)**.** *For a given sets $X, Y$, and a function $f : X \to Y$, the equivalent class $[x]_f$ are defined as:*

$$[y]_f = \{x \in X | f(x) = y\} \forall y \in f(X) \tag{2.24}$$

*And all the equivalent classes $[X]_f$ as:*

$$[X]_f = \{[f(x)]_f \forall x \in X\} \tag{2.25}$$

**Theorem 2.2.3.1.** $([X]_f, d_f)$ *is a* metric space *For any given* ***non-empty*** *set $X$, and a function $f : X \to \mathbb{R}$; $([X]_f, d_f)$ is a* metric space.

*Proof.* For all $[x]_f, [y]_f, [z]_f \in [X]_f$:

MS1 The codomain of $d_f$ is $\mathbb{R}_0^+$ by definition.

MS2

- $\Leftarrow$:

$$d_f([x]_f, [x]_f) = |x - x| = 0 \tag{2.26}$$

- $\Rightarrow$:

$$d_f([x]_f, [y]_f) = |x - y| = 0 \implies [x]_f = [y]_f \tag{2.27}$$

MS3

$$d_f([x]_f, [y]_f) = |x - y| = |y - x| = d_f([y]_f, [x]_f) \tag{2.28}$$

MS4

$$\begin{aligned} d_f([x]_f, [z]_f) &= |x - z| && (2.29)\\ &= |x - y + y - z| && (2.30)\\ &\leq |x - y| + |y - z| && (2.31)\\ &= d_f([x]_f, [y]_f) + d_f([y]_f, [z]_f) && (2.32) \end{aligned}$$

□

**Definition 2.2.3.5** (*ball*)**.** *in a* metric space

*For a given metric space* $(E, d)$*, a point* $p \in E$ *and a radius* $r \in \mathbb{R}^+$*, the (open)* ball $B(p; r)$ *is defined as*

$$B(p; r) = \{q \in E | d(p, q) < r\} \tag{2.33}$$

**Definition 2.2.3.6** (*open set*)**.** *in a* metric space

*For a given metric space* $(E, d)$*,* $A \subset E$ *is an* open set *if and only if:*

$$\forall p \in A \exists r \in \mathbb{R}^+ | B(p; r) \subset A \tag{2.34}$$

*The* $\emptyset$ *and* $E$ *are* open set*s by definition.*

**Definition 2.2.3.7** (*neighbourhood*)**.** *of a set in a* metric space

*If* $A \neq \emptyset \in E$*, an* open neighbourhood *of* $A$ *is an* open set *that contains* $A$*. If* $A = \{p\}$*, we can speak of the* open neighbourhood *of* $p$ *(instead of the set* $\{p\}$*).*

**Definition 2.2.3.8** (*interior*)**.** *of a set The interior of a set* $A$ *is largest* open set *that is contained in* $A$*; ant it is denoted as* $\dot{A}$*.*

**Definition 2.2.3.9** (*closed set*)**.** *in a* metric space *For a given* metric space $(E, d)$*,* $K \subset E$ *is an* closet set *if and only if* $E - K$ *is an* open set*.*

**Definition 2.2.3.10** (*cluster point*)**.** *of a set*

*A* cluster point *of* $A \subset E$ *is a point* $p \in E$ *that every* neighbourhood *has a* ***non-empty*** *intersection with* $A$*. All the* cluster point*s of* $A$ *are denoted as* $\bar{A}$*.*

**Definition 2.2.3.11** (*continuous mapping*)**.** *Let* $(E, d)$ *and* $(E', fd')$ *two* metric space*s. A mapping* $f : E \to E'$ *is said to be* continuous *at point* $x_0 \in E$ *if, for every* neighbourhood *of* $V'$ *of* $f(x_0) \in E$‘ *there is a* neighbourhood $V$ *of* $x_0$ *such that* $f(V) \subset V'$*.* $f$ *is said to be* continuous *in* $E$ *(or simple* continuous*) if it is* continuous *at* ***every point*** *of* $E$*.*

*In order to be* continuous *at* $x_0 \in E$*, a necessary and sufficient condition is that for every* neighbourhood $V'$ *of* $f(x_0)$ *in* $E'$*,* $f^{-1}(V')$ *be a* neighbourhood *of* $x_0 \in E$*.*

*Another, necessary and sufficient condition (only for* metric space*s) is that:*

$$\forall \epsilon > 0 \quad \exists \delta > 0 \quad d(x_0, x) < \delta \implies d'(f(x_0), f(x)) < \epsilon \tag{2.35}$$

*And all of these properties are equivalent:*

1. $f$ *is* continuous*;*
2. *for every* open set $U'$ *in* $E'$*,* $f^{-1}(U')$ *is* open set *in* $E$*;*
3. *for every* closed set $K'$ *in* $E'$*,* $f^{-1}(K')$ *is* closed set *in* $E$*;*
4. *for every* open set $A$ *in* $E$*,* $f(\bar{A}) \subset \overline{f(A)}$*.*

**Definition 2.2.3.12** (*lower semi-continuous mapping*)**.** *We say that* $f$ *is lower semi-continuous at* $x_0$ *if for every* $\epsilon > 0$ *there exists a neighbourhood* $U$ *of* $x_0$ *such that* $f(x) \geq f(x_0) - \epsilon$ *for all* $x$ *in* $U$ *when* $f(x_0) < +\infty$*, and* $f(x)$ *tends to* $+\infty$ *as* $x$ *tends*

*towards* $x_0$ *when* $f(x_0) = +\infty$*. Equivalently, in the case of a metric space, this can be expressed as:*

$$\liminf_{x \to x_0} f(x) \geq f(x_0) \tag{2.36}$$

*where* lim inf *is the limit inferior (of the function* $f$ *at point* $x_0$*).*

**Definition 2.2.3.13** (*limit*)**.** *of a function Let* $(E, d)$ *be a* metric space*,* $A \subset E$*, a a* cluster point *of* $A$*. Suppose that* $a \notin A$*. If* $f$ *is a mapping of* $A$ *in to a* metric space $E'$*. we say that* $f(x)$ *has a limit* $a' \in E'$ *where* $x \in A$ *tends to* $a$*. It the mapping* $g$ *of* $A \cup a \subset E'$ *defined by taking* $g(x) = f(x)$ *for every* $x \in A$*,* $g(a) = a'$*, is* continuous mapping *at point* $a$*. It is denoted as:*

$$\lim_{x \to a, x \in A} f(x) \tag{2.37}$$

*In order that* $a' \in E'$ *be a limit of* $f(x)$ *when* $x \in A$ *tends to* $a$*, a necessary and sufficient condition is that, for every* neighbourhood $V'$ *of* $a'$ *in* $E'$*, there exists a* neighbourhood $V$ *of* $a$ *in* $E$ *such that* $f(V \cap A) \subset V'$*.*

*Another (only for metric spaces), necessary and sufficient condition is that:*

$$\forall \epsilon > 0 \quad \exists \delta > 0 \quad x \in A, d(x, a) < \delta \implies d'(a', f(x)) < \epsilon \tag{2.38}$$

**Definition 2.2.3.14** (*Cauchy sequence*)**.** *In a* metric space $(E, d)$ *a* Cauchy sequence *is a infinte sequence* $\{x_n\}$ *such that:*

$$\forall \epsilon > 0 \quad \exists n_0 \in \mathbb{N} \quad \forall n_0 \leq n \in \mathbb{N} \quad d(x_{n_0}, x_n) < \epsilon \tag{2.39}$$

## 2.3. Computer Science

As part of the review of the state of the art, we present some examples of how the techniques presented in Mathematical Fundamentals, especially Topology, were applied in Computer Science to solve similar problems.

Important note:

These techniques were not applied to this research, but they were considered.

This section intends to be illustrative.

### 2.3.1. Nearest Neighbour Search

The problem of the NNS consists in given a set $S$ of points $n$ data points in a *metric space* $X$ (see item 2.2.3, Topology); the task is to process these points so that, given a *query point* $q \in X$ the data nearest to $q$ can be reported quickly [7].

#### 2.3.1.1. Space Partitioning

Space partitioning is one of NNS techniques where the *search space* is divided recursively and the search time is usually in order of $O(log(n))$ (where $n$ is the number elements in the *search space*).

#### 2.3.1.2. R-Tree

A **R-tree** is a height-balanced tree similar to a **B-tree**. Proposed by Guttman in 1984, **R-tree** is one of the most efficient methods that supports range queries. **R-tree** found significant use in both theoretical and applied contexts [58]. In figure 2.12 there is an example of a **R-tree** for 2D rectangles.

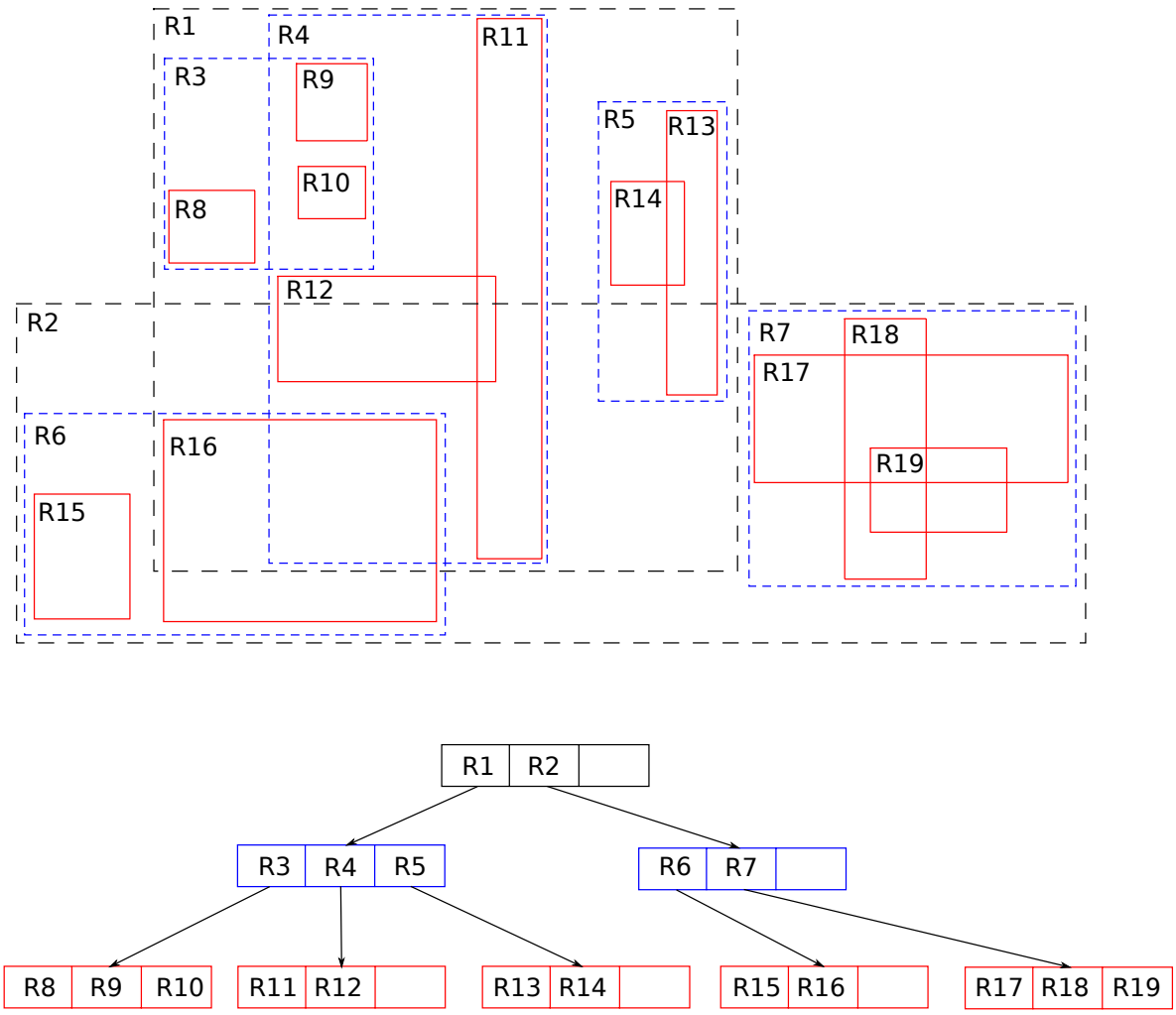


**Figure 2.12:** Simple Example of an R-tree for 2D Rectangles from Guttman Original Paper [58]

### 2.3.2. Locality Sensitive Hashing

LSH[37] is a family of techniques that reduces the dimensionality of high-dimensional data. The techniques of *space partitioning* work very well for scalar and vectorial values where the dimensionality is low. But, in documents, this approach is not so effective. Rajaraman and Ullman mentions that the most effective way to represent documents as sets, for the purpose of identifying lexically similar documents is to construct from the document the set of short strings that appear within it [118].

#### 2.3.2.1. Jaccard Similarity of Sets

**Definition 2.3.2.1.** *The* Jaccard index*, also known as the* Jaccard similarity coefficient*, is a statistic used for comparing the similarity and diversity of sample sets. The* Jaccard coefficient *measures similarity between finite sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets [118]:*

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{2.40}$$

*In the case when $A$ and $B$ are both empty the value is 1: $J(\emptyset,\emptyset) = 1$ [118].*

**Definition 2.3.2.2.** *The* Jaccard distance*, is complementary to the* Jaccard similarity coefficient*, and it can be defined as [118]:*

$$d_J(A,B) = 1 - J(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|} \tag{2.41}$$

#### 2.3.2.2. Min Hash

**Min hash** is a technique that takes a document as a string of characters. It defines a $k$-shingle for a document to be any sub-string of length $k$ found within the document. Then, associating with each document, the set of $k$-shingles that appear one or more

[37] Locality Sensitive Hashing

times within that document [118].

The probability that the *minhash* function for a random permutation of rows produces the same value for two sets equals the *Jaccard similarity* of those sets [118].

## 2.4. Related Work

Tools and investigations about MS are presented in this section. Some of them will serve as a comparison point and some other will help elaborate the research.

### 2.4.1. MS-GF+

Kim and Pevzner presented database search tool: **MS-GF**+ that is sensitive, (it identifies more peptides than most other database search tools, and universal, it works well for diverse types of spectra, different configurations of MS instruments and different experimental protocols [82].

Kim and Pevzner did a benchmark of **MS-GF**+ using diverse spectral data sets:

1. spectra of varying fragmentation methods;
2. spectra of multiple enzyme digests;
3. spectra of phosphorylated peptides;
4. and spectra of peptides with unusual fragmentation propensities produced by a novel alpha-lytic protease.

For all these data sets, **MS-GF**+ increased the number of identified peptides compared with commonly used methods for peptide identifications [82].

Kim and Pevzner emphasised that although **MS-GF**+ *is not specifically designed for any particular experimental* set-up, it improves the performance of tools specifically designed for these applications [82].

**MS-GF**+ database search tools used a scoring function $Score(P, S)$ to evaluate a PSM[38] of a *peptide* $P$ and a *spectrum* $S$, and further compute statistical significance of the resulting PSMs. Kim and Pevzner used $E$-values to evaluate statistical significance of individual PSM's and the target-decoy approach to estimate fdr[39]s [82].

[38]Peptide Spectrum Match
[39]False Discovery Rate

In this research **MS-GF+** will be used to evaluate the accuracy of the thecnique proposed.

### 2.4.2. Mascot

Mascot Server is live on a website for both Peptide Mass Fingerprint and MS/MS database searches. The sequence databases that can be searched on the **Matrix Science** are free. The public **Mascot** server provides [92]:

1. **SwissProt** is a high quality, curated protein database. These sequences are non-redundant [92].

2. **NCBInr** is a comprehensive, non-identical protein database maintained by **NCBI** [92].

3. **EMBL EST divisions** contain "single-pass" cDNA sequences, or Expressed Sequence Tags, from a number of organisms [92]. **Contaminants** is a database of common contaminants compiled by Max Planck Institute of Biochemistry [92].

4. **cRAP** is a database of common contaminants compiled by the Global Proteome Machine Organization [92].

According to Domselaar, **Mascot** uses a "probability-based MOWSE" algorithm to estimate the significance of a match. This method can be used to judge whether a result is significant or not. This metric allows to compare scores from different searches and on different databases [31].

This research will not take in consideration all of those repositories. As mentioned in limitations, this research will only take in consideration MS data produced by Orbitrap. This tool is not open, there is a practical way to compare the results of this research against **Mascot Server**.

### 2.4.3. HiXCorr

Kim et al. [81] presented a portable high-speed XCorr[40] engine, which does not create a mass bin array altogether, instead, it calculates XCorr directly from the peak list. Thus, it runs in $O(p)$ time where $p$ is the number of peaks in a *spectrum*, while all the previous engines are based on *XCorr* algorithms running in $O(m/f)$ time where $m$ is the *precursor mass* and $f$ is the *fragment tolerance*. Furthermore, Kim et al. engine is easily portable to almost every machine because it was developed in C. Optimising the engine for x86 machines. By embedding a x86 machine code can be a future research topic since XCorr score is widely used in peptide identification. Finally, Kim et al. did not trade correctness for efficiency. XCorr engine calculates the same XCorr score as Tide and Comet did [81].

### 2.4.4. Sailfish

**Sailfish** is not a MS database search tool. It is a RNA-seq identification tool. Patro et al. introduced **Sailfish** as a computational method for quantifying the abundance of previously annotated RNA isoforms from RNA-seq data. Sailfish entirely avoids mapping reads, which is a time-consuming step in all current methods. **Sailfish** provides quantification estimates much faster than existing approaches without loss of accuracy [112]. **Sailfish** is an example of how some computer techniques can help to reduce the execution time. In **Sailfish**, the fundamental unit of transcript coverage is the $k$-mer [112]. This technique is really similar to Min Hash presented in Section 2.3.2.2. The main idea behind it is to reduce the dimensionality of the search space, in order to facilitate the search dividing the document (or in this case the RNA sequence) in small pieces of length $k$.

Using a *perfect hash* function over the $k$-mer as a pillar, **Sailfish** builds an index where the search and updates are done in constant time and without locking for writing

[40] Cross Correlation

[112].

The idea of using $k$-mer for the identifications of DNA is widely used; but these techniques were not be applied to the identification of peptides. The idea of this research is take advantage of the structure of the spectra of MS to reduce the execution time.

### 2.4.5. SEQUEST

The **SEQUEST** program was the first and remains one of the most widely used tools for assigning a peptide sequence within a database to a MS/MS as Eng et al. [46] indicated. The XCorr score is the primary **score** function implemented within **SEQUEST** and it is this **score** that makes the tool particularly sensitive. Unfortunately, this score is computationally expensive to calculate, and thus, to make the score manageable. Eng et al. also indicated that SEQUEST uses a less *sensitive* but fast preliminary score and restricts the cross correlation to just the top 500 peptides returned by the preliminary score.

Classically, the XCorr score has been calculated using FFT[41] to generate the full correlation function. Eng et al. described an alternate method of calculating the XCorr score that does not require FFTs and can be computed in a fraction of the time. The fast calculation allows all candidate peptides to be scored by the XCorr function [46].

Also, Eng et al. suggests that potentially mitigating the need for the preliminary score, and enables an $E - value$ significance calculation based on the XCorr score distribution calculated on all candidate peptide sequences obtained from a sequence database [46].

But, with the increase of the size of the search database applying a function like this, it is prohibitive in time. This is opposite to the goal of this research.

[41] Fast Fourier Transforms

### 2.4.6. Tempest

Peptide spectral matching of MS/MS *spectra* to translate genomic sequence databases is the preferred method for identifying peptides in a *proteomics* experiment. However, recent advancements in *MS* instrumentation have enabled the generation of very high numbers of MS/MS spectra per experiment, which has resulted in greater attention to the speed of peptide spectral matching algorithms [102].

Milloy et al. presented a database searching program, **Tempest**, that combines efficient database digestion and MS/MS spectral indexing with fast similarity scoring on a GPU[42]. In Milloy et al. implementation, the entire similarity score, as well as other data processing tasks, are conducted on the GPU. **Tempest** uses the classical **SEQUEST XCorr** score as a primary metric for evaluating *similarity*. When scoring MS/MS spectra collected at high resolution, **Tempest** uses an "Accelerated Score" based on a dot product that closely recapitulates the more computationally expensive legacy XCorr without sacrificing score accuracy.

In Milloy et al. experience, **Tempest** provides compute-cluster level performance in an affordable desktop computer [102]. In general, unit resolution scoring still takes as much as three times longer than database digestion, which suggests that increased parallelism though the use of additional CPU cores or GPUs could further improve performance.

On the other hand, this research will assume that the the target computer has enough main memory. Part of the main idea is having all the index in memory in order to reduce execution time.

### 2.4.7. OpenMS

**OpenMS** is an open-source software C++ library for LC-MS data management and analysis, provided by Sturm et al.. It offers infrastructure for the rapid development

[42]Graphics Processor Unit

of mass spectrometry related software that will be used in this research. **OpenMS** is free software available under the three clause BSD license and runs under Windows, MacOSX and Linux according to their website [140].

### 2.4.8. PepNovo

Frank and Pevzner presented a scoring method for *de novo* interpretation of peptides from MS/MS data. Frank and Pevzner scoring method uses a probabilistic network whose structure reflects the chemical and physical rules that govern peptide fragmentation. Frank and Pevzner used a likelihood ratio hypothesis test to determine whether the peaks observed in the mass spectrum are more likely to have been produced under our fragmentation model than under a model that treats peaks as random events. Frank and Pevzner tested the *de novo* algorithm **PepNovo** on ion trap data and achieved results that were superior to popular *de novo* peptide sequencing algorithms [52].

### 2.4.9. Other Investigations

There many other investigations on these topics, we reviewed: [8; 23; 26; 50; 55; 57; 75; 88; 144; 145]. In particular Chen et al. [23] and Bandeira et al. [8] were very useful to this research. **PEAKS DB** it is a remarkable tool in the community, but it was not possible to get a license [166].

Chapter 3

# Research Methodology

"There are tumults of the mind, when, like the great convulsions of Nature, all seems anarchy and returning chaos; yet often, in those moments of vast disturbance, as in the strife of Nature itself, some new principle of order, or some new impulse of conduct, develops itself, and controls, and regulates, and brings to an harmonious consequence, passions and elements which seem only to threaten despair and subversion.".

William Gibson, The differential engine

**Table 3.1:** Objectives, Activities, and Deliverables

| Specific Objective | Status | Deliverable |
|---|---|---|
| Index Creation | Completed | Indexation Prototype |
| Search Engine Creation | Completed | Search Engine Prototype |
| Evaluation of the Metrics | Completed | Metrics review |

This chapter presents the objectives of this research and results obtained. The first point given is the analysis of the objectives in Section 3.1, Objectives. For Specific Objectives Table 3.1, Objectives, Activities, and Deliverables shows the executive summary of each one. Details of the deliverables are listed in Section 4.2.2.1, Analysis of Metrics. Table 4.1, Deliverables provides deliverables associated with each Specific objective.

## 3.1. Objectives

We established following objectives to develop this research:

### 3.1.1. Main Objective

"Verify if the application of NNS techniques can reduce the peptide identification ET Metric from MS with no reduction of the AMB Metric and not effect of COV Metric."

Conclusions presented in Metrics review show that this Main Objective was accomplished. The NNS techniques based on a topological analysis provided a formal and a practical way to approach the problem. Notwithstanding the fact that the original guided approach was replaced by a *de novo* one; the same techniques prevailed. At the end, we found better results.

### 3.1.2. Specific Objectives

#### 3.1.2.1. Index Creation

"Define an indexation strategy of peptides that allows fast searches applying Nearest Neighbour Search techniques."

The indexation strategy provided by the Build Index enables a constant time look-up (depending on the Operative System Library resolution) for the validation of the *admissibility* of a set of peaks.

#### 3.1.2.2. Search Engine Creation

"Define a **search strategy** using the data previously indexed in Index Creation that allows a fast responses with small sacrifices to the AMB Metric. "

The search strategy provided by the Build Index enables a fast response depending of the number of peaks. One important point to mention is that there is a 100% AMB Metric because each solution of the algorithm is *admissible* for that given *error*. Another important point is that the *search* is made on the whole search space since the analysis follows a *de novo* strategy.

#### 3.1.2.3. Evaluation of the Metrics

"Evaluating the metrics of the peptide search using the technique defined in Search Engine Creation."

The **search strategy** was analysed in a formal way in Algorithm 5, $\text{Search}_\varepsilon$ Complexity and Algorithm 5, $\text{Search}_\varepsilon$ Correctness.

## 3.2. Proposed Approach

Originally, this research tried to apply an experimental approach using Section 3.2.1, Analysis of Variance; but, regarding the size of the problem, a more formal analytic

path was followed.

### 3.2.1. Analysis of Variance

As Montgomery mentioned, experimentation is a vital part of the scientific and the engineering method. Now; there are certainly situations where the scientific phenomena are so well understood that useful results including mathematical models can be developed directly by applying these well-understood principles [104].

In general, experiments are used to study performance of processes and systems. Visualisation can be usually conceptualised as a combination of operations, machines, methods, people, and other resources that transforms some input into an output that has one or more observable response variables. Some of the process variables and material properties $x_1, x_2, \ldots, x_p$ are controllable, whereas other variables $z_1, z_2, z_3, \ldots, z_q$ are uncontrollable [104].

The objectives of the experiment may include the following:

1. Determining which variables are most influential on the response $y$ [104].
2. Determining where to set the influential $x$'s so that y is almost always near the desired nominal value [104].
3. Determining where to set the influential $x$'s so that variability in $y$ is small [104].
4. Determining where to set the influential $x$'s so that the effects of the uncontrollable variables $z_1, z_2, \ldots, z_q$ are minimised.

### 3.2.2. Results Analysis

The ANOVA[1] verifies the variation of the means of different groups associated to a set of treatments. ANOVA allows to accept, or reject an hypothesis with a probability of error (hopefully very low) with statistical evidence. [157].

[1] Analysis of Variance

It is necessary to establish a ANOVA hypothesis, an alternative hypothesis, and to accomplish three assumptions.

#### 3.2.2.1. Alternative and Null Hypotheses

The hypotheses of interest in ANOVA are as follows:

$$H_0 : \quad \mu_1 = \mu_2 = \mu_3 = ... = \mu_k \tag{3.1}$$

$$H_1 : \quad \exists \mu_j \neq \mu \qquad j = 1, 2, ..., k \tag{3.2}$$

The *null* hypothesis in ANOVA (equation 3.1) is always that there is no difference in means and the alternative hypothesis (equation 3.2) is always that the means are not all equal.

### 3.2.3. Original Metrics

#### 3.2.3.1. FDR Metric

The *false discovery rate* (FDR) of MS-GF+ is 1% according to Kim and Pevzner [82]. The *accuracy* will be measured as the number of PSM of *hirief-search* that are common to MS-GF+ divided by the total of PSM reported by MS-GF+.

#### 3.2.3.2. COV Metric

The *coverage* will be measured as the total number of PSM divided by the total number of samples.

#### 3.2.3.3. ET Metric

The *execution time* will be measured as the time when the execution ends minus the time when the execution starts in milliseconds.

### 3.2.4. Original Hypothesis

One of the most important contributions of the NNS is that it usually reduces the search time from linear time to logarithmic (even constant time) related to the search space size. This is the main reason why the hypothesis are oriented towards the reduction of the time. The original hypotheses of this investigation are presented below:

$H_1$ The application of NNS applied to peptide identification from MS produce a reduction in the *ET Metric* by 50% with no impact in the *AMB Metric* ($\pm$5%) and no impact of the *COV Metric* ($\pm$5%).

$H_2$ If $H_1$ is true then the *ET Metric* follow a $O(n \cdot log(s))$ behaviour where $s$ is the size of the search space and $n$ the number of peptides to search.

Chapter 4

# Results and discussion

Sometimes, the problems that we face are overwhelming; and, sometimes the problem is that we do not understand the problem.

The original approach used was not successful because the size of the search space was overwhelming. One of the first approaches considered was space partition [58] based on the precursor mass and the Isoelectric point [15; 60; 142]; but this approach can only consider the data that was given for building the this index. So it was not able to extend the results to modifications of the given peptides.

## 4.1. *De novo* Approach

The firsts attempts of *de novo* sequencing were done by Hamm et al. and Sakurai et al. in 1986 and 1984 respectively. The approach was to match each spectrum to every possible spectrum. Regardless of how, these comparision are prohibitive [59; 122]. Please refer to Figure 4.1, Index Visualisation to see the amount of possibilities for each molecular weight.

Later, another approach named *subsequencing* was followed. This approach takes small parts of a peptide. Then, those matches are extended to one residue at the time trying to find the best match [12; 67; 74; 134].

At the same time, Johnson and Biemann presented a graphical tool for displaying the spectral data. That tool connects with lines the peaks which distance corresponds with an amino acid residue. This method was very useful for manual sequencing, but failed for high-throughput. [74].

The approach proposed by Bartels is the most successful nowadays. It applies Graph theory (see Definition 2.1.8.4, for details) to the detection of peptides. Many tools use this technique [9; 26; 49; 145; 166].

This section presents the final approach that this research followed. The first topic presented is a introduction to the notation used all over this document in Section 4.1.1, Notation and Definitions.

Later in Section 4.1.2, Indexation we present the proposed algorithm for the

indexation of the data; the underlying formal concepts are present in Section 4.1.1, Notation and Definitions. In Section 4.1.3, Search Algorithm we present the basis for searching a solution together with correctness and complexity proofs..

### 4.1.1. Notation and Definitions

This section presents the definitions and theorems developed in this research. Most of these definitions have a direct translation into source code (see Chapter 6, Prototypes).

**Definition 4.1.1.1** (*Molecular Mass* of an *Amino Acid Residue*)**.** $mw_\alpha : \mathcal{A} \to \mathbb{R}^+$*: See definition in Table 2.1, Aminoacids Codes, Molecular Mass, and Composition.*

**Definition 4.1.1.2** (*Amino Acid Residue*)**.** $\mathcal{A}$*: The set of* Amino acids *is* ***sorted*** *based on the* Molecular Mass *of an* Amino Acid Residue*. For reference see Table 2.1, Aminoacids Codes, Molecular Mass, and Composition:*

$$\mathcal{A} = \{\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n\} \text{ where } mw_\alpha(\alpha_i) < mw_\alpha(\alpha_j) \quad \forall i, j \in \mathbb{N}, 1 \leq i < j \leq n \quad (4.1)$$

$$|\mathcal{A}| = n \quad (4.2)$$

**Definition 4.1.1.3** (*Peptide*)**.** $\pi$*: Let define a* peptide $\pi$ *as a* string *based on the* Amino Acid Residue *alphabet.*

$$\pi = \langle \pi_1, \pi_2, \pi_3, \ldots, \pi_m \rangle \in \mathcal{A}^* \quad (4.3)$$

$$|\pi| = m \quad (4.4)$$

**Definition 4.1.1.4** (*Molecular Weight* of a *Peptide*)**.** $mw_\pi : \mathcal{A}^* \to \mathbb{R}_0^+$*:*

$$mw_\pi(\pi) = \sum_{i=1}^{|\pi|} mw_\alpha(\pi_i) \tag{4.5}$$

**Definition 4.1.1.5** (*Combination* of *Amino Acid Residue*)**.** $\mathbb{N}_0^{\mathcal{A}}$*:*

$$\mathbb{N}_0^{\mathcal{A}} = \{c : \mathcal{A} \to \mathbb{N}_0\} \tag{4.6}$$

**Theorem 4.1.1.1** ($(\mathbb{N}_0^{\mathcal{A}}, +)$)**.** $(\mathbb{N}_0^{\mathcal{A}}, +)$ *is a* commutative monoid *The binary function* $+ : \mathbb{N}_0^{\mathcal{A}} \times \mathbb{N}_0^{\mathcal{A}} \to \mathbb{N}_0^{\mathcal{A}}$ *is defined as:*

$$(a + b)(\alpha) = a(\alpha) + b(\alpha) \quad \forall \alpha \in \mathcal{A} \tag{4.7}$$

*Proof.* $\forall a, b \in \mathbb{N}_0^{\mathcal{A}}, \alpha \in \mathcal{A}$:

CM1 Associative:

$$\begin{aligned} ((a + b) + c)(\alpha) &= (a + b)(\alpha) + c(\alpha) && (4.8)\\ &= a(\alpha) + b(\alpha) + c(\alpha) && (4.9)\\ &= a(\alpha) + (b + c)(\alpha) && (4.10)\\ &= ((a + (b + c))(\alpha) && (4.11) \end{aligned}$$

CM2 Neutral element: $\exists c_0 \in \mathbb{N}_0^{\mathcal{A}}; c_0(\alpha) = 0$

$$(a + c_0)(\alpha) = a(\alpha) + c_0(\alpha) \tag{4.12}$$
$$= a(\alpha) + 0 \tag{4.13}$$
$$= a(\alpha) \tag{4.14}$$
$$= 0 + a(\alpha) \tag{4.15}$$
$$= c_0(\alpha) + a(\alpha) \tag{4.16}$$
$$= (c_0 + a)(\alpha) \tag{4.17}$$

CM3 Commutative:

$$(a + b)(\alpha) = a(\alpha) + b(\alpha) \tag{4.18}$$
$$= b(\alpha) + c(\alpha) \tag{4.19}$$
$$= (b + a)(\alpha) \tag{4.20}$$

□

**Theorem 4.1.1.2** $(G)$**.** $\langle G \rangle = \mathbb{N}_0^{\mathcal{A}}$ *The set* $G \subset \mathbb{N}_0^{\mathcal{A}}$ *finitely generates* $\mathbb{N}_0^{\mathcal{A}}$.

$$g_i(\alpha) = \begin{cases} 1 & \alpha = \alpha_i \\ 0 & \alpha \neq \alpha_i \end{cases} \quad \forall 1 \leq i \leq |\mathcal{A}| \tag{4.21}$$

$$G = \{g_i \quad | \quad \forall 1 \leq i \leq |\mathcal{A}|\} \tag{4.22}$$

*Proof.*

$$c = \sum_{i=1}^{|A|} \sum_{j=1}^{c(\alpha_i)} g_i \quad \forall c \in \mathbb{N}_0^{\mathcal{A}} \tag{4.23}$$

□

Conclusion: Search Space is Exponential

**Lemma 4.1.1.1** (Size of Search space: $\Omega(exp(|A|+|s|))$**.** $|\{\pi \in \mathcal{A}^*; |\pi| \leq n\}| \in \Omega(exp(|\mathcal{A}|+n)) \quad \forall n \in \mathbb{N}$ *for a give size of a spectrum* $|s| = n$.

*Proof.* $\forall n \in \mathbb{N}$

$$|\{s \in \mathcal{A}^*; |s| \leq n\}| = \sum_{i=0}^{n} |\{s \in \mathcal{A}^*; |s| = n\}| \tag{4.24}$$

$$= \sum_{i=0}^{n} \begin{pmatrix} |A|+i \\ |A| \end{pmatrix} \tag{4.25}$$

$$= \sum_{i=0}^{n} \frac{(|A|+i)!}{|A|!i!} \tag{4.26}$$

$$= \sum_{i=0}^{n} \frac{|A|! \prod_{j=|A|+1}^{|A|+i} j}{|A|!i!} \tag{4.27}$$

$$= \sum_{i=0}^{n} \frac{\prod_{j=|A|+1}^{|A|+i} j}{i!} \tag{4.28}$$

$$= \sum_{i=0}^{n} \prod_{j=1}^{i} \frac{|A|+j}{j} \tag{4.29}$$

$$< \sum_{i=0}^{n} \frac{(|A|+i)^i}{i!} \tag{4.30}$$

$$< \sum_{i=0}^{n} \frac{(|A|+n)^i}{i!} \tag{4.31}$$

$$\lim_{n\to\infty} \sum_{i=0}^{n} \frac{(|A|+n)^i}{i!} = exp(|A|+n) \tag{4.32}$$

□

**Definition 4.1.1.6** (*Molecular Weight* of a *Combination* of *Amino Acid Residue*)**.** $mw_\theta : \mathbb{N}_0^{\mathcal{A}} \to \mathbb{R}_0^+$*:*

$$mw_\theta(c) = \sum_{i=1}^{|\mathcal{A}|} c(\alpha_i) \cdot mw_\alpha(\alpha_i) \tag{4.33}$$

**Definition 4.1.1.7** (*Ambiguity* of a *Combination* of *Amino Acid Residue*)**.** $amb_\theta : \mathbb{N}_0^{\mathcal{A}} \to \mathbb{N}$*:*

$$amb_\theta(c) = \frac{(\sum_{i=1}^{|\mathcal{A}|} c(\alpha_i)!}{\prod_{i=1}^{|\mathcal{A}|} c(\alpha_i)!} \tag{4.34}$$

**Definition 4.1.1.8** (Resolution $\mathcal{R}$)**.** *: Let $\mathcal{R} \in \mathbb{R}^+$ be the resolution used to solve the peptide sequence problem. The scale function $s_\mathcal{R} : \mathbb{N}_0^{\mathcal{A}} \to \mathbb{N}_0$ value for a* Combination *of* Amino Acid Residue *is defined as:*

$$s_\mathcal{R}(c) = \lfloor \mathcal{R} \cdot mw_\theta(c) \rfloor \tag{4.35}$$

**Definition 4.1.1.9** (*Combinations* of a *Distance* at Resolution $\mathcal{R}$)**.** $[\cdot]_\mathcal{R} : \mathbb{N}_0 \to 2^{\mathbb{N}_0^{\mathcal{A}}}$

$$[d]_\mathcal{R} = \{c \in N_0^{\mathcal{A}} \text{ where } s_\mathcal{R}(c) = d\} \tag{4.36}$$

**Definition 4.1.1.10** (*Ambiguity* of a *Distance* at Resolution $\mathcal{R}$)**.** $amb_\mathcal{R} : \mathbb{N}_0 \to \bar{\mathbb{N}}$

$$amb_\mathcal{R}(d) = \begin{cases} \sum_{c \in [d]_\mathcal{R}} amb_\theta(c) & [d]_\mathcal{R} \neq \emptyset \\ \infty & [d]_\mathcal{R} = \emptyset \end{cases} \tag{4.37}$$

### 4.1.2. Indexation

Algorithm 0, Build Index traverses the *Cayley Graph* of $(\mathbb{N}_0^{\mathcal{A}}, +)$ finitely generated by $G$.

The implementation of Build Index is presented in Section 6.1.3.3, Implementation of Algorithm 0, Build Index on page 115.

**Algorithm 4** Build Index

```
1: procedure BuildIndex(M, R)
2:     I[0] ← {c_0}
3:     for i = 0 to R · M do
4:         for j = 1 to |A| do
5:             I[i + s_R(g_j)] ← I[i + s_R(g_j)] ∪ (I[i] + g_j)
```

Conclusion: Build Index is Exponential

**Theorem 4.1.2.1** (Algorithm 0, Build Index Computational Complexity)**.** *Algorithm 0, Build Index computes $\mathcal{I}$ in:*

$$\textsc{BuildIndex}(\mathcal{R},\mathcal{M}) \in \Omega\left(\mathcal{R}\cdot\mathcal{M} + exp(|A| + \lfloor\frac{\mathcal{M}}{mw_\alpha(\alpha_1)})\rfloor\right) \text{ in space} \quad (4.38)$$

$$\textsc{BuildIndex}(\mathcal{R},\mathcal{M}) \in \Omega\left(\mathcal{M}\cdot\left(\mathcal{R} + \cdot exp(|A| + \lfloor\frac{\mathcal{M}}{mw_\alpha(\alpha_1)}\rfloor)\right)\right) \text{ in time} \quad (4.39)$$

**Lemma 4.1.2.1** (Size of search space: $\Omega(exp(|A| + |s|))$)**.** $|\{\pi \in \mathcal{A}^*; |\pi| \leq n\}| \in \Omega(exp(|\mathcal{A}| + n)) \quad \forall n \in \mathbb{N}$ *for a give size of a spectrum* $|s| = n$

*Proof.* Let $W = \lfloor\frac{\mathcal{M}}{mw_\alpha(\alpha_1)}\rfloor$ the length of longest string that can be constructed with the smallest amino acid. From Lemma 4.1.2.1, Size of search space: $\Omega(exp(|A| + |s|))$ is known that the total number combinations is in the order of $e^{|A|+W}$. Since $|\mathcal{A}|$ is a constant in the algorithm, the complexity is in the order of the highest loop $\mathcal{R}\cdot\mathcal{M}$ and the total number of combinations.

$$\textsc{BuildIndex}(\mathcal{R},\mathcal{M}) \in max\{\Omega(\mathcal{R}\cdot\mathcal{M}), \Omega(exp(|A| + W))\} \quad (4.40)$$

$$\in \Omega\left(\mathcal{R}\cdot\mathcal{M} + exp(|A| + W)\right) \quad (4.41)$$

$$\therefore \textsc{BuildIndex}(\mathcal{R},\mathcal{M}) \in \Omega\left(\mathcal{R}\cdot\mathcal{M} + exp(|A| + W)\right) \text{ in space} \quad (4.42)$$

Taking the average of combination by cell as:

$$\frac{e^{|A|+W}}{\mathcal{R} \cdot \mathcal{M}} \tag{4.43}$$

Using a modern set implementations that allows $n \cdot ln(n)$ time for the union operation:

$$\mathcal{R} \cdot \mathcal{M} \cdot \frac{exp(|A| + W)}{\mathcal{R} \cdot \mathcal{M}} \cdot ln \frac{exp(|A| + W)}{\mathcal{R} \cdot \mathcal{M}} \tag{4.44}$$

$$exp(|A| + W) \cdot (|A| + W - ln(\mathcal{R} \cdot \mathcal{M})) \tag{4.45}$$

$$W \cdot exp(|A| + W) \tag{4.46}$$

$$\textsc{BuildIndex}(\mathcal{R}, \mathcal{M}) \in max\{\Omega(\mathcal{R} \cdot \mathcal{M}), \Omega(W \cdot exp(|A| + W))\} \tag{4.47}$$

$$\in \Omega\left(\mathcal{R} \cdot \mathcal{M} + \lfloor \frac{\mathcal{M}}{mw_{\alpha}(\alpha_1)} \rfloor exp(|A| + W)\right) \tag{4.48}$$

$$\in \Omega\left(\mathcal{M} \cdot (\mathcal{R} + exp(|A| + W))\right) \tag{4.49}$$

$$\therefore \textsc{BuildIndex}(\mathcal{R}, \mathcal{M}) \in \Omega\left(\mathcal{M} \cdot (\mathcal{R} + exp(|A| + W))\right) \text{ in time} \tag{4.50}$$

□

**Lemma 4.1.2.2** $(O(p(n)) \in nO(1))$**.** *For a given E equivalences matrix computed with Algorithm 0, Build Index, p can be computed* $O(1)$*.*

Let define a mass spectra peak, or simply a *peak p* as a pair of *mass/charge* $(mz)$ and *intensity* $(t)$.

$$p = (mz, t) | mz \in \mathbb{R}, t \in \mathbb{R} \tag{4.51}$$

### 4.1.3. Search Algorithm

**Definition 4.1.3.1** ($\mathcal{I}_{\mathcal{R}}^{\mathcal{M}}$)**.** *Index for at $\mathcal{R}$ resolution to $\mathcal{M}$ maximum: Is the result of the Algorithm 0, Build Index.*

**Definition 4.1.3.2** (*spectrum*)**.** *Let define a mass spectrum, or simply a* spectrum *as a* string *over the $\mathbb{R}^+$ alphabet where:*

$$s = \langle m_1, m_2, m_3, \ldots, m_K \rangle \tag{4.52}$$

$$m_i < m_j \quad \forall i, j \in \mathbb{N}, i < j \leq K \tag{4.53}$$

$$|s| = K \tag{4.54}$$

*For convenience the value $\langle s \rangle_0 = m_0 = 0$ is assumed to be present in all spectrum, but is not counted.*

**Definition 4.1.3.3** (*$\varepsilon$ error tolerance*)**.** *From now, $\varepsilon \in \mathbb{N}_0$ is known as the* error tolerance*.*

**Definition 4.1.3.4** (*admissibility* of a *distance* at *$\varepsilon$ error tolerance*)**.** $adm_{\varepsilon}^{\delta} : \mathbb{N}_0 \to 2$ *For a given $\varepsilon$* error tolerance *and distance $d \in \mathbb{N}_0$:*

$$adm_{\varepsilon}^{\delta}(d) \iff [B(d, \varepsilon)]_{\mathcal{R}} \neq \emptyset \vee d > \mathcal{M} \tag{4.55}$$

**Definition 4.1.3.5** (Explanations of a equivalent class related to a distance)**.** $explain_{\varepsilon} : \mathbb{N}_0 \to \bar{\mathbb{N}} \times \mathbb{N}_0 \times \mathbb{N}_0$

*The permutations $p_{\varepsilon}(d)$, error $e_{\varepsilon}(d)$, and, equivalent combinations $c_{\varepsilon}(d)$ of $d$ with $\varepsilon$ are defined as:*

$$(p_\varepsilon(d), e_\varepsilon(d), c_\varepsilon(d)) \mapsto \begin{cases} min_{c \in B(d,\varepsilon)}\{(amb_\varepsilon(c), |d-c|, c)\} & d \leq \mathcal{M} \\ (\frac{d}{\mathcal{M}} \cdot p_{\mathcal{M}}(0), 0, d) & d > \mathcal{M} \end{cases} \tag{4.56}$$

*The ball* $B(d, \varepsilon)$ *is over the* $(\mathbb{N}_0, d_1)$ *metric space using the* lexicographical order *relation.*

**Definition 4.1.3.6** (*admissible spectrum* at $\varepsilon$ *error tolerance*)**.** $adm^{\varsigma}_{\varepsilon} : \mathbb{R}^{+*} \to 2$*:*

$$adm^{\varsigma}_{\varepsilon}(s) \iff \begin{cases} adm^{\delta}_{\varepsilon}(s_{\mathcal{R}}(\langle s \rangle_i - \langle s \rangle_{i-1})) \quad \forall i \in \mathbb{N}, i \leq |s| & |s| > 0 \\ 1 & |s| = 0 \end{cases} \tag{4.57}$$

**Definition 4.1.3.7** (*ambiguity* of an *admissible spectrum* at $\varepsilon$ *error tolerance*)**.** $amb^{\varsigma}_{\varepsilon} : \mathbb{R}^{+*} \to \bar{\mathbb{R}^{+}}$*:*

$$amb^{\varsigma}_{\varepsilon}(s) = \begin{cases} \infty & |s| \leq 1 \\ \frac{1+\sum_{i=1}^{|s|} log(p_\varepsilon(\langle s \rangle_i - \langle s \rangle_{i-1}))}{2^{|s|}} & |s| > 1 \end{cases} \tag{4.58}$$

**Definition 4.1.3.8** (*spectrum key*)**.** $key_S : \mathbb{R}^{+*} \to 2^K_S$*: For a given non empty* spectrum $S$ *and a sub-*spectrum $s \subset S$*:*

$$\langle key_S(s) \rangle_i \iff \langle s \rangle_i \in S \quad \forall i \in \mathbb{N}, i \leq |s| \tag{4.59}$$

**Definition 4.1.3.9** (Succession of values of $s.s$)**.** $\{s.s_n\} \in 2^K_S, \{s.s_n\} : \mathbb{N} \to 2^K_S$*:*

$$s.s_n = \textit{ value of } s.s \textit{ after } n \textit{ executions of } \textbf{\textit{Step}}(s) \tag{4.60}$$

**Definition 4.1.3.10** (Succession of values of $s.b$)**.** $\{s.b_n\} \in 2^K_S, \{s.b_n\} : \mathbb{N} \to 2^K_S$*:*

**Algorithm 5** Search$_\varepsilon$

```
1:  procedure STEP(s)
2:      e ← 0
3:      if |s.p| > 0 then
4:          t ≪ p
5:          for i = 1 to k do
6:              n ← t + 1⃗_i
7:              if n ∉ s.f then
8:                  s.f ← s.f ∪ n
9:                  if admissible_ε(n) then
10:                     n ≫ s.p
11:                     e ← 1
12:         if e = 0 then
13:             if t <_ε^amb b then
14:                 b ← t
15:             else
16:                 e ← 1
        return e
17: procedure INIT(s)
18:     s.s ← 0⃗
19:     s.b ← 0⃗
20:     1⃗ ≫ s.p
21: procedure SEARCH(s)
22:     while step(s) = 1 do
23:         print(s.s)
        return s.b
24: procedure BEST(s)
25:     INIT(s)
26:     while |s.p| > 0 do
27:         Search(s)
        return s.b
```

$$s.b_n = \textit{ value of } s.b \textit{ after } n \textit{ executions of } \textbf{\textit{Search}}(s) \tag{4.61}$$

**Definition 4.1.3.11** ($\cdot^{amb}_{\varepsilon} : 2^K_S \times 2^K_S \to 2$)**.** *Is $\cdot$ ambiguous than: for $u, v \in 2^K_S$ keys of the given spectrum $S$:*

$$u <^{amb}_{\varepsilon} v \iff amb^{\varsigma}_{\varepsilon}(key^{-1}_S(u)) < amb^{\varsigma}_{\varepsilon}(key^{-1}_S(v)) \tag{4.62}$$

$$u \leq^{amb}_{\varepsilon} v \iff amb^{\varsigma}_{\varepsilon}(key^{-1}_S(u)) \leq amb^{\varsigma}_{\varepsilon}(key^{-1}_S(v)) \tag{4.63}$$

$$u =^{amb}_{\varepsilon} v \iff amb^{\varsigma}_{\varepsilon}(key^{-1}_S(u)) = amb^{\varsigma}_{\varepsilon}(key^{-1}_S(v)) \tag{4.64}$$

$$u \geq^{amb}_{\varepsilon} v \iff amb^{\varsigma}_{\varepsilon}(key^{-1}_S(u)) \geq amb^{\varsigma}_{\varepsilon}(key^{-1}_S(v)) \tag{4.65}$$

$$u >^{amb}_{\varepsilon} v \iff amb^{\varsigma}_{\varepsilon}(key^{-1}_S(u)) > amb^{\varsigma}_{\varepsilon}(key^{-1}_S(v)) \tag{4.66}$$

**Definition 4.1.3.12** (Open sets of $S$)**.** $\tau^S_{\varepsilon}$*: For a given $S$* spectrum *and $\varepsilon$* error tolerance*:*

$$U_{\infty} = \emptyset \tag{4.67}$$

$$U_0 = 2^K_S \tag{4.68}$$

$$U_r = \{s \textit{ where } s \in 2^K_S \wedge adm^{\varsigma}_{\varepsilon}(s) \wedge amb^{\varsigma}_{\varepsilon}(s) > r\} \tag{4.69}$$

$$\tau_{\varepsilon}^{S} = \{U_r \forall r \in \bar{\mathbb{R}}_0^+\} \cup \{U_0, U_\infty\} \tag{4.70}$$

**Lemma 4.1.3.1** ($(\tau_{\varepsilon}^{S}, \vee, \wedge, U_0, U_\infty)$ is a *bounded lattice*)**.** *With the following definitions:*

$\vee : \tau_{\varepsilon}^{S} \times \tau_{\varepsilon}^{S} \to \tau_{\varepsilon}^{S}$

$$U_r \vee U_s = U_{r \wedge s} \tag{4.71}$$

$\wedge : \tau_{\varepsilon}^{S} \times \tau_{\varepsilon}^{S} \to \tau_{\varepsilon}^{S}$*:*

$$U_r \wedge U_s = U_{r \vee s} \tag{4.72}$$

**Lemma 4.1.3.2** ($(2_S^K, \tau_{\varepsilon}^{S})$)**.** *is a topological space*

*Proof.*

$$\emptyset, 2_S^K \in \tau_{\varepsilon}^{S} \text{ by definition} \tag{4.73}$$

For any $U_r, U_s \in \tau_{\varepsilon}^{S}$:

$$U_r \cap U_s = U_r \wedge U_s \tag{4.74}$$

For any index $I$ over $\tau_{\varepsilon}^{S}$:

$$\bigcup_{r \in I} U_r = \bigvee_{r \in I} (U_r) \tag{4.75}$$

□

**Lemma 4.1.3.3** ($(\{s.b_n\}, \geq_{\varepsilon}^{amb}) \in (2_S^K, \tau_{\varepsilon}^{S})$)**.** *is a net:*

*Proof.* For any $1 \leq i < j \leq |admissible_{\varepsilon}(2_S^K)|$ holds:

$$s.b_i >_{\varepsilon}^{amb} s.b_j \tag{4.76}$$

Because line 18 of Algorithm 5, Search$_\varepsilon$ is the only place where a change in $s.b$ is done and the update is make if and only if the new value has less ambiguity than the last one.

For any $|admissible_\varepsilon(2_S^K)| \leq i < j$ holds:

$$s.b_i =_\varepsilon^{amb} s.b_j \tag{4.77}$$

After the Algorithm 5, Search$_\varepsilon$ found all the admissible solutions the value of $s.b$ is not updated, so $s.b_i = s.b_j$.

□

**Theorem 4.1.3.1** (Algorithm 5, Search$_\varepsilon$ Correctness)**.** $(\{s.b_n\}, \geq_\varepsilon^{amb}) \to min(admissible_\varepsilon(2_S^K), \leq_\varepsilon^{amb})$ *in* $(2_S^K, \tau_\varepsilon^S)$

*Proof.*

$$M = min(admissible_\varepsilon(2_S^K), \leq_\varepsilon^{amb}) \tag{4.78}$$

The only 2 neighbourhoods of $M$ are $U_\Omega$ and $U_{ambiguity_\varepsilon(M)}$.

**By contradiction**:

$$\exists N \in \{U_\Omega, U_{ambiguity_\varepsilon(M)}\} \forall L \in \mathbb{N} \text{ where } s.b_i \notin N \forall L \leq i \in \mathbb{N} \tag{4.79}$$

$$s.b_i \in U_\Omega \forall i \in \mathbb{N} \implies U_i = U_{ambiguity_\varepsilon(M)} \tag{4.80}$$

But there is a contradiction with:

$$U_\alpha \subset \cdots \subset U_i \subset \cdots \subset U_j \subset \cdots \subset U_M \subset U_\omega \quad \forall 0 \leq i < j \leq ambiguity_\varepsilon(M) \tag{4.81}$$

In other case, there is $U_N$ that has less ambiguity than $M$; so M is not the one with

minimum ambiguity.

Finally:

$$\therefore (\{s.b_n\}, \geq_{\varepsilon}^{amb}) \to M \tag{4.82}$$

□

**Definition 4.1.3.13** ($G_S(V, E)$)**.** *Search space graph: for a given spectrum $S$ the search space directed graph:*

$$V = 2_S^K \tag{4.83}$$

$$(i, j) \in E \iff admissible_{\varepsilon}(i) \wedge admissible_{\varepsilon}(j) \wedge |i \cap j| = 1 \wedge i >_{\varepsilon}^{amb} j \tag{4.84}$$

**Definition 4.1.3.14** ($h_S : 2_S^K \times 2_S^K \to 2$)**.** *Heuristic search function:*

$$h(u, v) = \begin{cases} 1 & |keys_S^{-1}(u)| < |keys_S^{-1}(v)| \\ 0 & |keys_S^{-1}(u)| > |keys_S^{-1}(v)| \\ 1 & u <_{\varepsilon}^{amb} v \\ 0 & u >_{\varepsilon}^{amb} v \\ u < v & otherwise \end{cases} \tag{4.85}$$

**Lemma 4.1.3.4.** *Algorithm 5, Search$_{\varepsilon}$ is an implementation of Dijkstra's algorithm: Using a Fibonacci Heap with the heuristic $h$.*

**Theorem 4.1.3.2** ($O(\textsc{Step}(s)) \in O(K^2)$)**.**

*Proof.* Every operation of a $v \in 2_S^K$ takes $K = |s|$ to be completed. There are 2 cycles in lines 9 and 22 bounded by $K$.

$$\therefore O(\textsc{Step}(s)) \in O(K^2) \tag{4.86}$$

□

Conclusion: Search time was reduced to $O(n^3)$

**Theorem 4.1.3.3** ($O(\textsc{Search}(s)) \in O(|s|^3)$)**.**

*Proof.* Every call to $O(\textsc{Step}(s))$ in line 29 $O(|s|^2)$ operations, and every operation produces a new entry in the top of $s.p$ that have one more peak than its predecessor. Since the maximum numbers of peaks is $K$:

$$\therefore O(\textsc{Search}(s)) \in O(K^3) \tag{4.87}$$

□

**Theorem 4.1.3.4** (Algorithm 5, Search$_\varepsilon$ Complexity)**.** $O(\textsc{Best}(s)) \in O(|s|(|E|+2^{|s|}))$

*Proof.* The Dijkstra's algorithm [123] using a Fibonacci Heap [53] has a time complexity of:

$$O((|E| + |V|) \cdot log(|V|)) \tag{4.88}$$

$$O((|E| + 2^{|s|}) \cdot log(2^{|s|})) \tag{4.89}$$

$$O(K(|E| + 2^{|s|})) \tag{4.90}$$

□

**Theorem 4.1.3.5** ($s_{\mathcal{R}}$ : is a *continuous mapping*)**.** *from* $([\mathbb{N}_0^{\mathcal{A}}]_{mw_\theta}, d_{mw_\theta})$ *to* $(\mathbb{R}_0^+, d_1)$.

*Proof.* Let any $I$ index family over $(\mathbb{R}_0^+, d_1)$ *open set*s:

$$U' = \cup_{i \in I} B(c_i, r_i) \tag{4.91}$$

Any *open set* is conformed as the union of *ball*.

$$s_{\mathcal{R}} \text{ is a } \textit{continuous mapping} \iff s_{\mathcal{R}}^{-1}(U') \text{ is a } \textit{open set} \tag{4.92}$$

$$s_{\mathcal{R}}^{-1}(U') = s_{\mathcal{R}}^{-1}(\cup_{i \in I} B(c_i, r_i)) \tag{4.93}$$

$$= \cup_{i \in I} s_{\mathcal{R}}^{-1}(B(c_i, r_i)) \tag{4.94}$$

$$= \cup_{i \in I} \cup_{c_j \in s_{\mathcal{R}}^{-1}(c_i)} (B(c_j, \lceil \frac{r_i}{\mathcal{R}} \rceil)) \tag{4.95}$$

And, any *open set* is conformed as the union of *ball*s, in this base over the $c_j$ index family.

□

**Theorem 4.1.3.6** ($s_R^{-1}$ is a **??**)**.** *from* $(\mathbb{R}_0^+, d_1)$ *to* $([\mathbb{N}_0^{\mathcal{A}}]_{mw_\theta}, d_{mw_\theta})$.

*Proof.* Let any $J$ index family over $([\mathbb{N}_0^{\mathcal{A}}]_{mw_\theta}, d_{mw_\theta})$ *open set*s:

$$U = \cup_{j \in J} B(c_j, r_j) \tag{4.96}$$

Any *open set* is conformed as the union of *ball*.

$$s_{\mathcal{R}}^{-1} \text{ is a } \textit{continuous mapping} \iff s_{\mathcal{R}}(U) \text{ is a } \textit{open set} \tag{4.97}$$

$$s_{\mathcal{R}}(U) = s_{\mathcal{R}}(\cup_{j\in J} B(c_j, r_j)) \tag{4.98}$$

$$= \cup_{j\in J} s_{\mathcal{R}}(B(c_j, r_j)) \tag{4.99}$$

$$= \cup_{j\in J} B(s_{\mathcal{R}}(c_j), \mathcal{R}r_j) \tag{4.100}$$

And, any *open set* is conformed as the union of *ball*. □

## 4.2. Deliverable List

This is the list of deliverables that this research produced:

### 4.2.1. Source Code

#### 4.2.1.1. Indexation Prototype

The design of the Index Creation is presented in Build Index and the prototype source code is attached in Index Prototype Source Code .

#### 4.2.1.2. Search Engine Prototype

The design of the Search Engine Creation is presented in Search$_\varepsilon$ and the prototype source code is attached in Search Prototype Source Code.

### 4.2.2. Mathematical

#### 4.2.2.1. Analysis of Metrics

The evaluation of metrics was elaborated with Algorithm 5, Search$_\varepsilon$ Complexity and Algorithm 5, Search$_\varepsilon$ Correctness. Both theorems show that the results provided by the Search Engine Creation are correct; they also evaluate the execution time.

**Table 4.1:** Deliverables

| Specific Objective | Deliverable |
|---|---|
| Indexation Prototype | Algorithm 5, Search$_\varepsilon$ |
| | Index Prototype Source Code |
| Search Prototype | Algorithm 0, Build Index |
| | Search Prototype Source Code |
| Evaluation of metrics | Theorem 4.1.3.4, Algorithm 5, Search$_\varepsilon$ Complexity |
| | Theorem 4.1.3.1, Algorithm 5, Search$_\varepsilon$ Correctness |

## 4.3. Indexation

Figure 4.1, Index Visualisation was generated with Section 6.1, Index Prototype Source Code using a resolution $\mathcal{R} = 100$ to a maximum of $\mathcal{M} = 1000da$. The abscissa axis remains with linear scale meanwhile the ordinate use a logarithmic scale, and the colour scale remains logarithmic. We can appreciate how the ambiguity of combinations grows in exponential way according to Theorem 4.1.2.1, Algorithm 0, Build Index Computational Complexity tells us. Originally, the parallel implementation of the algorithm was not part of the research proposal. But, for connivance during the developing stage it was implemented.

The complexity analysis of the parallel implementation was not done as a part of this research. Nevertheless; our guess is that $P \in O(\mathcal{R} \times mw_\alpha(\alpha_1))$ according to Brent's Theorem [1] [80] as we can see in Figure 4.2, Duration of Index Generation on Different Machines, the duration of the generation of the index source code is really fast (less than 10 minutes, in a MacOS machine, and about 1 minute on a Linux machine).

[1] The Brent's Theorem affirms that a parallel computer with each processor can perform an arithmetic operation in unit time. Further, if we assume that the computer has exactly enough processors to exploit the maximum concurrency in an algorithm with $N$ operations, such that $T$ time steps suffice. It says that a similar computer with fewer processors, $P$, can perform the algorithm in time $T_P \leq T + \frac{N-T}{P}$ [80]

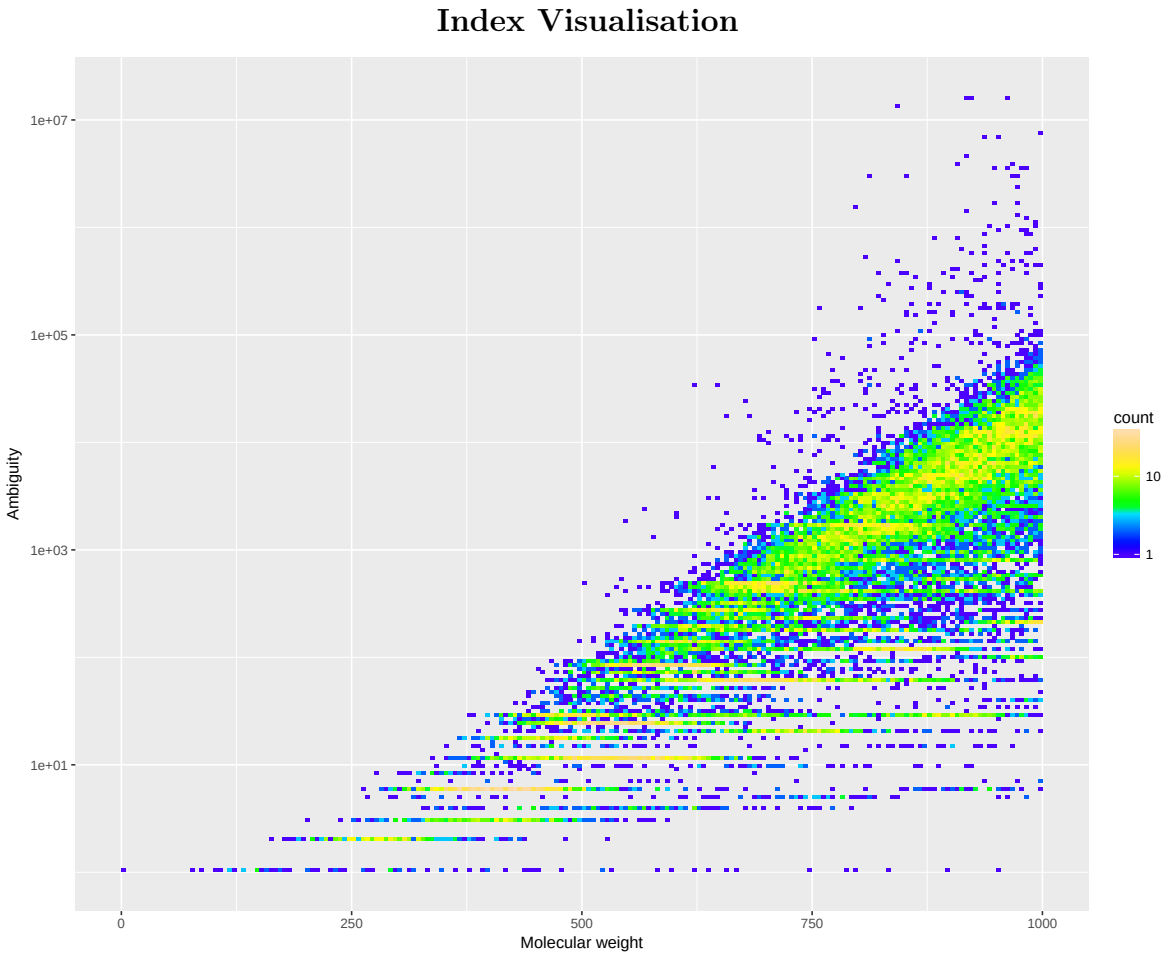


**Figure 4.1:** Index Visualisation, with parameters $\mathcal{R} = 100, \mathcal{M} = 1000$.

## 4.4. Search

In Figure 4.3, Spectrum Interpretation Example 1 and Figure 4.4, Spectrum Interpretation Example 2, we can observe examples of interpretations of a couple of spectra.

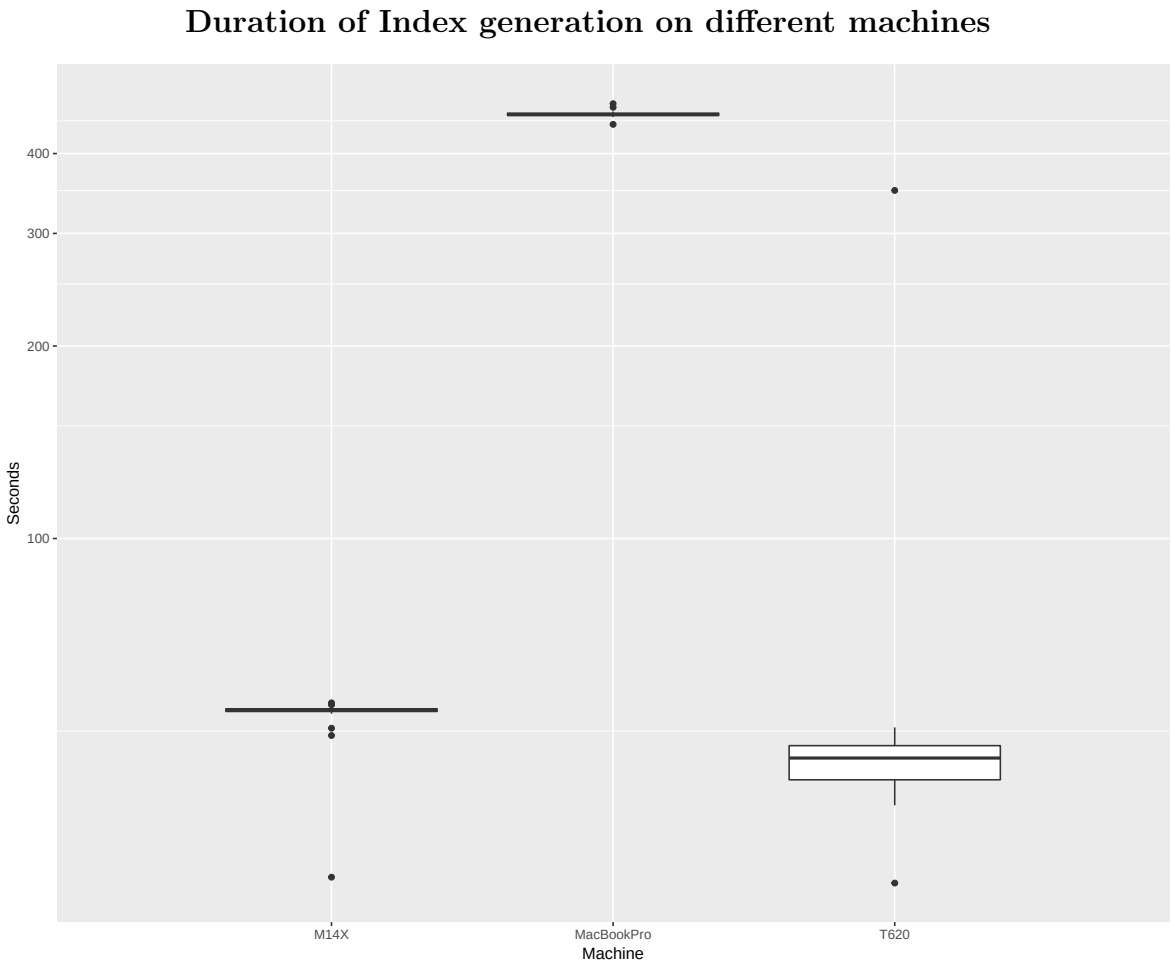


**Figure 4.2:** Index generation time with $\mathcal{R} = 100, \mathcal{M} = 1000$. MacBookPro is a 2.2 GHz i7, threads, MacOs High Sierra 10.3.3; M14X is an Alienware M14X R2 with Intel(R) Core(TM) i7-4700MQ CPU @ 2.40GHz, 8 threads, Linux 4.9.0-3-amd64 1 SMP Debian; and T620 is a Dell PowerEdge T620 with two Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz, 32 threads, Linux 4.4.0-45-generic 66-Ubuntu. There were 30 executions in Section 6.1, Index Prototype Source Code. These times did not contemplate the compilation time of the generated code.

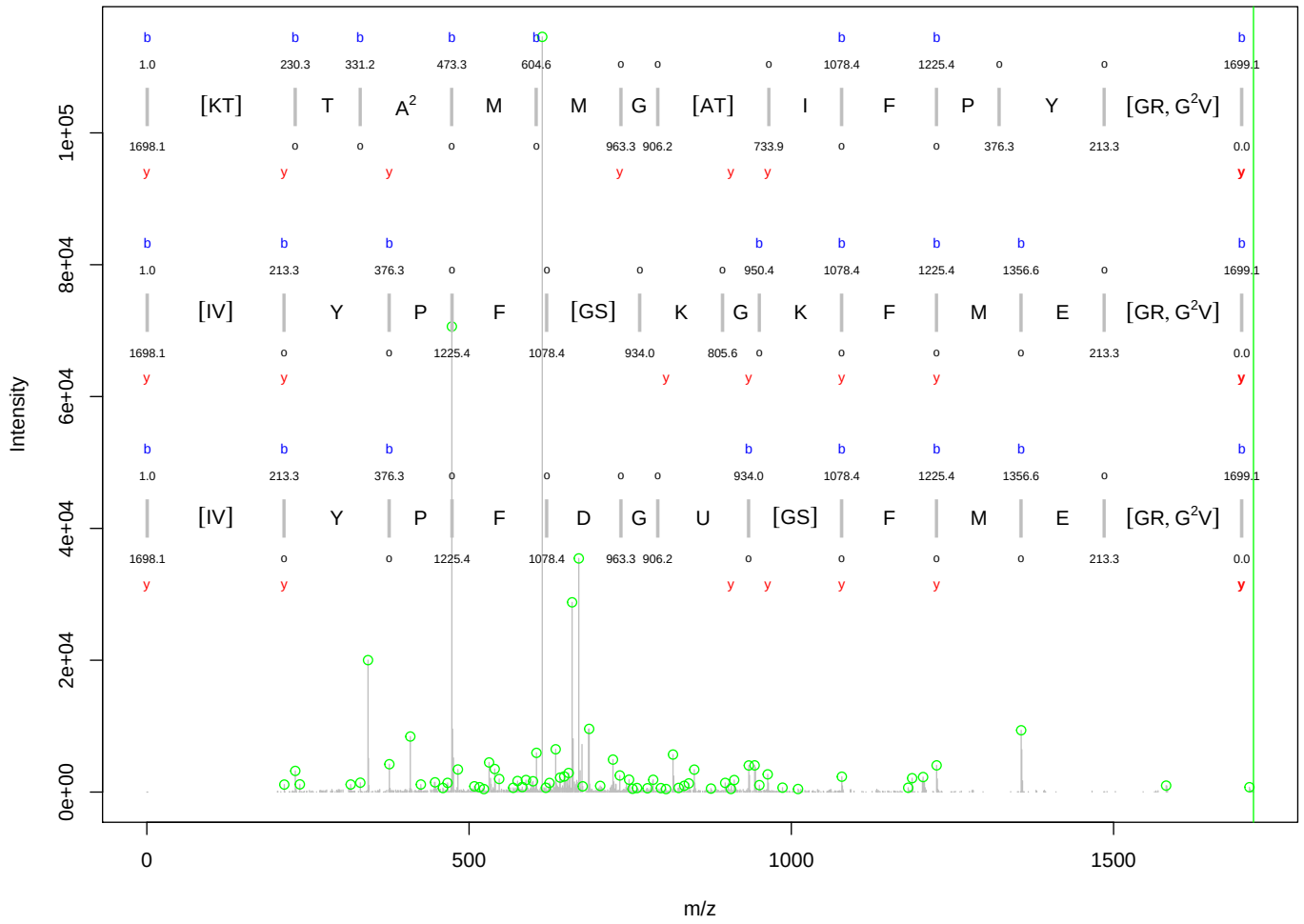


**Figure 4.3:** Spectrum interpretation example 1, generated with the `MS3TMT10_01022016_32917-33481.mzML.gz` file provided by the MSDATA project [109]. 70 peaks were selected from file, those with green circles, and 1M iterations were executed for Search$_\varepsilon$ using the Search Engine Prototype, with an $\epsilon = 0.25da$. Only the top 3 solutions are presented by default. To read the file we used the MZR package provided by Chambers et al. [22].

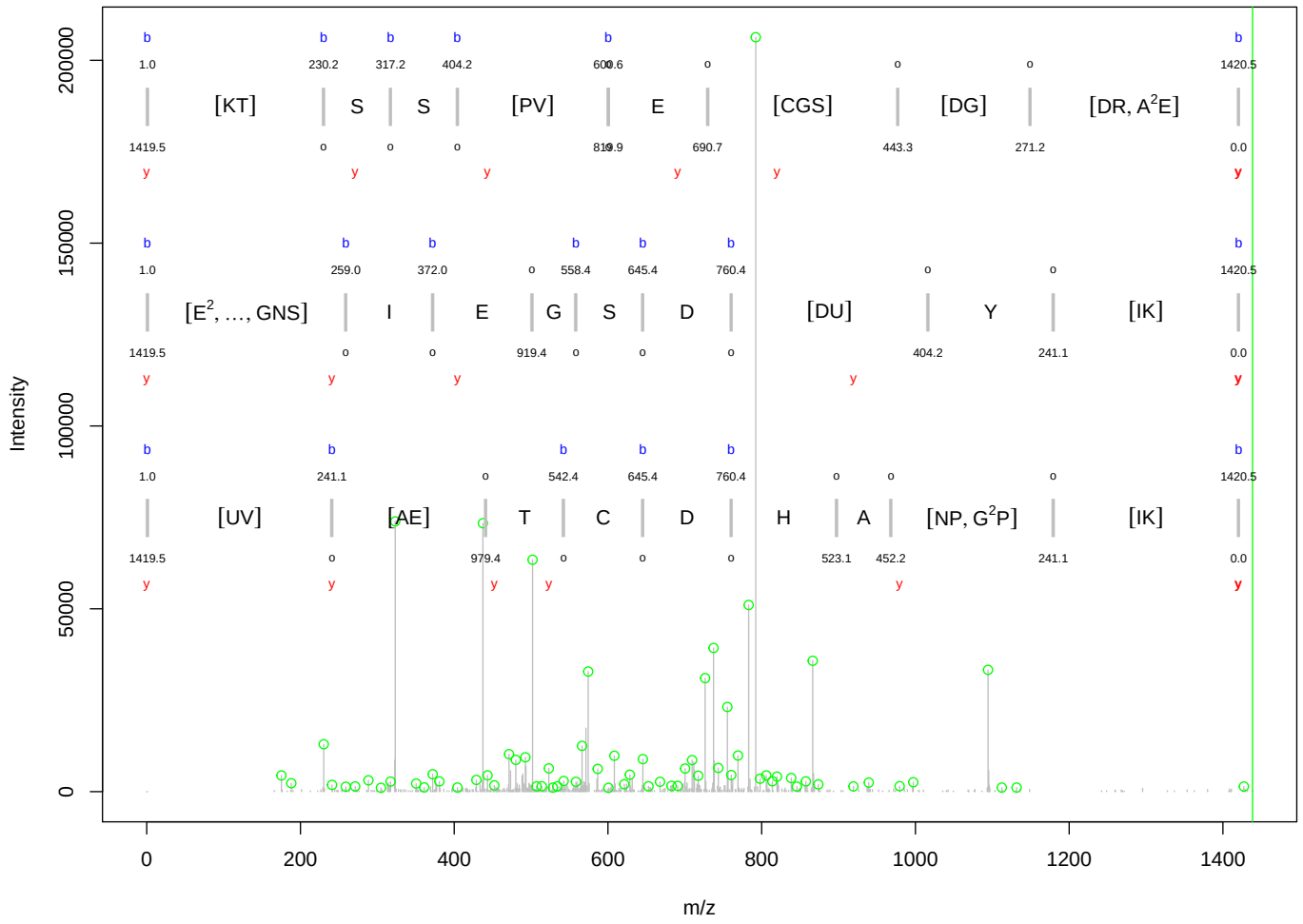


**Figure 4.4:** Spectrum interpretation example 2. This example follows the same characterisation than the example 1 but we selected a different scan.

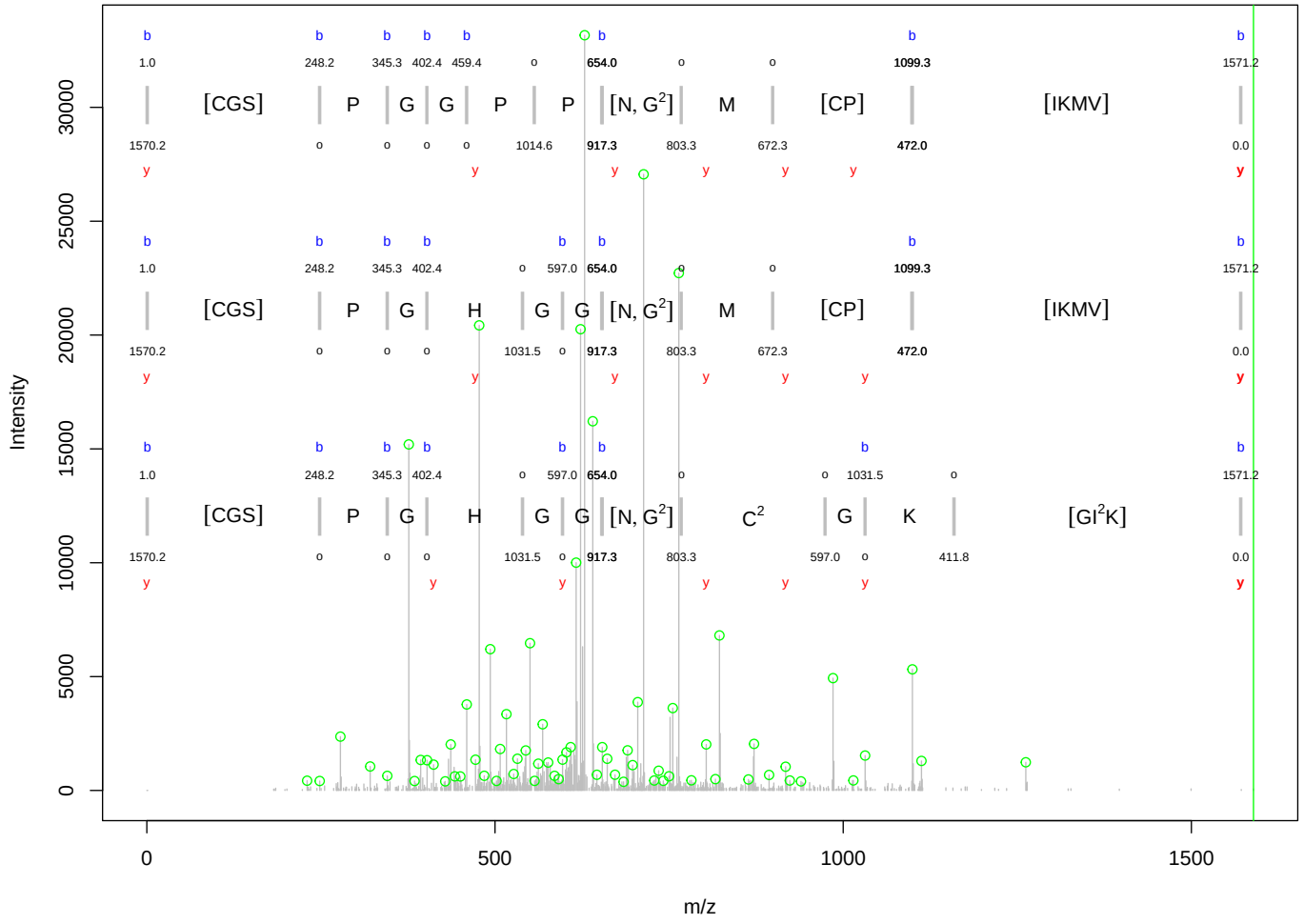


**Figure 4.5:** Spectrum interpretation example 3. This example follows the same characterisation than the example 1 and 2.

**Table 4.2:** Original Metrics Analysis

| Metric | Value | Analysis |
|---|---|---|
| FDR Metric | 0% | Algorithm 5, Search$_\varepsilon$ always produces a feasible solution (see Theorem 4.1.3.1, Algorithm 5, Search$_\varepsilon$ Correctness) |
| COV Metric | 100% of $exp(\|\mathcal{A}\| + \|s\|)$ | Algorithm 5, Search$_\varepsilon$traverses in the whole search space. |
| ET Metric | $O(\text{SEARCH}(s)) \in O(\|s\|^3)$ | The execution time does not depend on the amount of amino acids nor search space size. It depends on the number of peaks. (See Theorem 4.1.3.4, Algorithm 5, Search$_\varepsilon$ Complexity) |

## 4.4.1. Metrics review

A review of the original metrics is presented. We offer a summary of the metrics conclusions in Table 4.2, Original Metrics Analysis.

### 4.4.1.1. AMB Metric

The original FDR Metric ( Section 3.2.3.1) defined based in the possibility that the solution given for the implemented algorithm is wrong. But, in the way the Algorithm 5, Search$_\varepsilon$ works, it only moves on feasible solutions, there is no chance that a wrong solution is produced. This is proven in Theorem 4.1.3.1, Algorithm 5, Search$_\varepsilon$ Correctness. Since this fact, the FDR Metric will be always 0%.

### 4.4.1.2. COV Metric

In the same way that the Algorithm 5, Search$_\varepsilon$ will produce 0% FDR Metric, the original COV Metric will be fixed in 100%. Always the search is *de novo*; it takes in consideration the whole search space, no matter the size of any given reference genome. The order of the size of the search space is calculated in Lemma 4.1.2.1, Size of search space: $\Omega(exp(|A| + |s|))$.

#### 4.4.1.3. ET Metric

The ET Metric was originally defined was depending on the size of the search space (see Lemma 4.1.2.1, Size of search space: $\Omega(exp(|A|+|s|))$). Nevertheless, according to Theorem 4.1.3.4, Algorithm 5, Search$_\varepsilon$ Complexity, it does not depend on size of the search space. It depends on the size of the given spectrum and the amount of feasible solutions wanted only.

### 4.4.2. Error at High Resolution

The error for any given resolution is bounded by Equation 4.102, Error at High Resolution. Since $mw_\theta$ is a *homomorphism* from $(\mathbb{N}_0^A, +)$ to $(\mathbb{R}_0, +)$, and $S_\mathcal{R}$ is a *lower semi-continuous mapping*:

$$s_\mathcal{R}(a+b) \leq s_\mathcal{R}(a) + s_\mathcal{R}(b) \leq s_\mathcal{R}(a+b) + 1 \tag{4.101}$$

$$\lfloor \mathcal{R}mw_\theta(a+b) \rfloor \leq \lfloor \mathcal{R}mw_\theta(a) \rfloor + \lfloor \mathcal{R}mw_\theta(b) \rfloor \leq \lfloor \mathcal{R}mw_\theta(a+b) \rfloor + 1 \tag{4.102}$$

Whem using a high resolution (see Equation 4.103, Error at High Resolution), it is expected that error tends to 0.

$$\lim_{\mathcal{R}\to\infty} \frac{1}{\mathcal{R}} = 0 \tag{4.103}$$

Chapter 5

# Conclusions, recommendations, and future work

"All the speed he took, all the turns he'd taken and the corners he'd cut in Night City, and still he'd see the matrix in his sleep, bright lattices of logic unfolding across that colorless void...".

William Gibson, Neuromancer

## 5.1. Conclusions and Recommendations

First; we found that a mixed approach where multiple disciplines involved showed to be the most successful. In this case, Computer Science and Mathematics, Topology, in particular, worked together to provide a solution to a Molecular Biology problem. Although it seems to be a straightforward recommendation, it is necessary to include sources and experts in these topics to construct a more comprehensive view of the problem.

The formal examination of the problem from the Topological point of view helped to understand succinctly. This understanding was easy translated to a computer program. The reader will perceive how every definition in the Section 4.1, *De novo* Approach is transparently translated to a piece of code in Chapter 6, Prototypes. We suggest to follow a formal description of the problem and the computer program to solve it in the most direct way possible. Maybe, using some formal definition as Z [69] will be too complicated, but having the necessary definitions and theorems will prove to be necessary.

In this particular case, the formal analysis done by other authors suggests that Graph theory is the right approach. But, we extended it using Topology. We extended the Definition 2.1.8.4, in Definition 4.1.3.13, $G_S(V, E)$ applying our concepts and reducing the complexity of graph as well as adding the error and admissibility tests. These test can be executed in **constant time** with the provided index for any pair of peaks and in **linear time** for a full interpretation. At the end, we conclude with a search/interpretation algorithm in an acceptable time as with we see in Theorem 4.1.3.2, $O(\textsc{Step}(s)) \in O(K^2)$ and Theorem 4.1.3.3, $O(\textsc{Search}(s)) \in O(|s|^3)$.

Proving that this problem is part of the ExpSpace [1] was not part of this research.

[1] ExpSpace is the set of all decision problems solvable by a deterministic Turing machine in $O(2^{p(n)})$ space, where $p(n)$ is a polynomial function of $n$ [135].

Additionally; Lemma 4.1.2.1, Size of search space: $\Omega(exp(|A| + |s|))$ strongly suggests that this is the case. We strongly recommend avoiding to apply a brute force approach as was done in the past [59; 122] in 1986; 1984.

#### 5.1.0.1. Understanding the "Shape" of the Problem

Avoiding to apply a brute force approach.

Part of the solution that helps to reduce the index generation time to avoid intractability of this problem was the use of the properties of Definition 4.1.3.5, Explanations of a equivalent class related to a distance. Taking advantage of these properties, we were not forced to traverse every possible combination in the search space. The usage of a Definition 2.2.2.7, *Cayley Graph* as a traverse strategy ensures that there are not missing elements and we were not forced to visit elements outside the bounds given to the algorithm.

It is important to mention that the graphical display of the results helps in every stage of the process. The work of Johnson and Biemann in 1989 was visionary because it is really useful for the developer and for the reader of the spectra to get an immediate response of the results. In the early stages, the lack of a graphical output delayed validation of results. After the implementation of Section 6.3, R extension package: `tms2R` was done the validation, and correction of the algorithm had a speed up.

After the application of small optimisation like Section 6.1.1.1 Memoization speed up [page 107], and the application of parallel programming in the Section 6.1.3.3 Implementation of Algorithm 0, Build Index [page 115], the execution time of the index generation was reduced.

## 5.2. Future Work

There are several topics that we want to explore and more questions than we had with the original one. We enumerate the points that we want to continue evaluating.

### 5.2.1. Extension of the Model

There are many topics that we did not explore and they will help to produce better solutions.

#### 5.2.1.1. Extension of the Amino Acid Alphabet

One step further is the inclusion of all peptides modification in the same way that Kim and Pevzner did [82]. This will allow to discover new scenarios that were not contemplated in this first attempt.

#### 5.2.1.2. Results Refinement

The prototypes keep track of the metric $SS_{error}$ that is not contemplated originally to sort the results and provide one with lowest $SS_{error}$. Also; the algorithm finds only one ion interpretation for the given spectrum. It can be extended as a complement to find another ion interpretation from complement of the found solution [104].

#### 5.2.1.3. Other Ion Types Detection

This research only applies to the detection of only one ion type; $b-ion$ or $y-ion$; but other ions types can be included in the model. Adding other ion types [2] will provide more evidence to first provide the most probable interpretations.

[2] considering: $a$, $a^*$, $a^o$, $b^*$, $b^o$, $c^*$, $d$, $v$, $w$, $y^*$, $y^o$, and $z-ion$.

#### 5.2.1.4. Consideration of Fragmentation Methods

At this moment; the algorithm does not take into consideration the fragmentation method in the same way that Kim and Pevzner do [82]. The inclusion of this method will reduce the number of admissible solution to a number that is human consumable [6; 60; 128; 129]. Right now, We only consider Tripsina [3] for this research.

### 5.2.2. Optimisation

#### 5.2.2.1. Reduction of Index Building Time

The Algorithm 0, Build Index has a parallel level of $\lfloor min(mw(A)) \rfloor$. Taking advantage of modern Multiple cores processors, this total index building time can be reduced; especially if the suggestion Section 5.2.1.1, Extension of the Amino Acid Alphabet is taken [14; 93].

#### 5.2.2.2. Reduction of Search Running Time

In the same way that Algorithm 0, Build Index can be executed in parallel, the Algorithm 5, Search$_\varepsilon$ can be extended to a parallel version [14; 93] that will reduce the executing time. In this case, the pending queue, visited dictionary, and, solutions are the only one structure to protect with semaphores.

### 5.2.3. Variable Behaviour

Our implementation allows the customisation of many variables such fragmentation tolerance, overall tolerance, interpretation ordering, and many orders. One parameter that we will need to study is the number of peaks, and the signal to noise ratio. Small variations on these paremeters can reduce the number of results.

[3] Tripsina cuts on K or R at the **N-terminal**.

#### 5.2.3.1. Study of how an Admissible Error affects Running Time and Ambiguity

This research keeps the admissible error as a constant; but, it is for sure that it affects the running time of Algorithm 5, Search$_\varepsilon$ and the ambiguity of the reported results. These questions can be addressed using an experimental approach [103; 104].

### 5.2.4. Large-Scale and Complex Data Sets and Evaluation

At this moment, we limited the application of this *de novo* technique to a small number of Proteomic datasets, but we plan to extend the use of our approach to a large-scale and complex data set, such as Devabhaktuni and Elias did [28].

#### 5.2.4.1. Database Assisted Interpretation

Zhang et al. offer a private software package that using a genome references produces better results [166]. We suspect that applying a filter based on a genome reference will provide more accurate results.

#### 5.2.4.2. Solution Discrimination

Right now, we face the same problem that Taylor and Johnson did [144] because we report the best solutions that are very similar among themselves. We want to avoid this behaviour, but we do not have an answer for this situation now.

#### 5.2.4.3. Performance and Accuracy Analysis with more *de novo* Software Packages

We plan to make a full comparison with others *de novo* interpretation software packages in the similar way that Sergey Pevtsov et al. did [132]. We were comparing our results to `Lutefisk` software (see Figure 5.1, Execution of `Lutefisk` software with the default parameters.). Nevertheless, there is no straight way to compare our results.

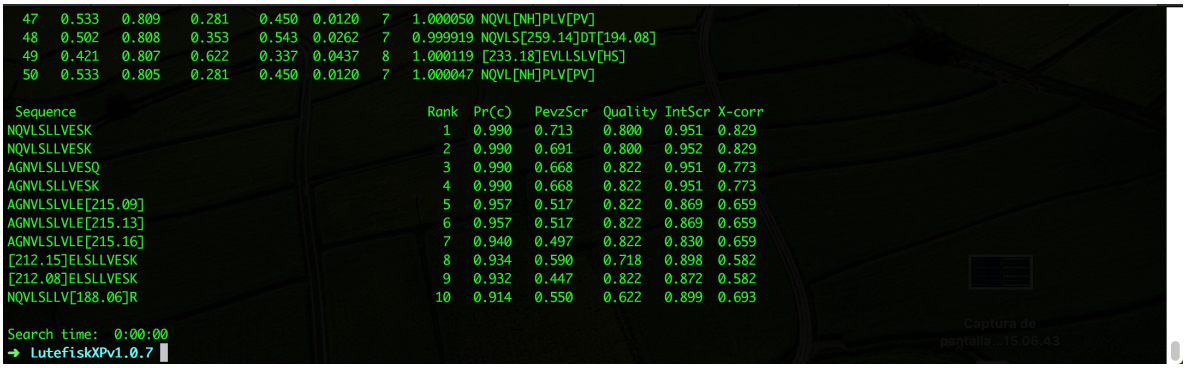


**Figure 5.1:** Execution of `Lutefisk` software with the default parameters.

Chapter 6

# Prototypes

## 6.1. Index Prototype Source Code

The Index Prototype Source Code was developed using C++14 standard [139], but the generated code will only need C++11 standard [131; 138]. For portability reasons the Boost Program Options [150] and Boost Filesystem Library Version 3 [10] were used to facilitate the port from Linux and Mac systems. CMake 3.5 [13] was used for compiling. The main implementation details of the indexation prototype is presented in this section. There was added some support for parallel programming [14; 80; 85; 93] on Listing 6.1.2, Multi-threading support.

The Level of parallelism allowed on Listing 163, Index Prototype Source Code on page 109 must not surpass the half of `ARG_MAX` of the `POSIX` standard library. [147].

**Listing 6.1:** `tms2index.h`: Index header

```
1 //
```

## 2 6.1.1. Index generation header

```
3  #ifndef TMS2_TMS2INDEX_H
4  #define TMS2_TMS2INDEX_H
5  #include <map>
6  #include <iomanip>
7  #include <iterator>
8  #include <fstream>
9  #include <thread>
10 #include <mutex>
11 #include <exception>
12 #include <boost/filesystem.hpp>
13 #include <boost/program_options.hpp>
14 namespace tms2 {
15 inline namespace indexgen {
16 namespace fs = boost::filesystem;
17 namespace po = boost::program_options;
18 //
```

### 6.1.1.1. Memoization speed up

A memoization approach was used to speed up the calculus of theambiguity of a point [1].

```
19 class memoizer_factorial {
20   mutable ::std::vector<::std::size_t> result_;
21 public:
22   memoizer_factorial() {
23     result_.push_back(1);
24   }
25   inline ::std::size_t operator()(::std::size_t f) const {
26     for (::std::size_t l = result_.size(); l <= f; l++) {
27       result_.push_back(l * result_[l - 1]);
28     }
29     return result_[f];
30   }
31 };
32 extern const memoizer_factorial fact;
33 // Using float as the best match for Molecular Mass of an Amino Acid Residue.
34 typedef float mw_t;
35 // Reference to Amino Acid Residue.
36 typedef const ::std::map<::std::string, mw_t> aa_list_t;
37 // Reference to Amino Acid Residue.
38 extern const aa_list_t aa_list;
39 //
```

### 40 6.1.1.2. Program arguments

```
41 class arguments {
42 public:
43   int silent;
44   int verbose;
45   // Argument R for Algorithm 0, Build Index. See Definition 4.1.1.8, Resolution R.
46   int R;
47   // Argument M for Algorithm 0, Build Index
48   int M;
49   //
50   int B;
51   // Working directory where the source code files for the index is goinggoing to be generated.
52   fs::path wd;
53   // Additional argument for make program. It defines the root directorywhere to install the target library.
54   fs::path prefix;
55   // Additional argument, if define it will copy all the source code filesto the give directory. This allow to create
         a self contained Rextension.
56   fs::path tms2R;
57   // See Equation 4.35, Resolution R
58   inline ::std::size_t operator()(mw_t mw) const { return mw * R; }
```

```
59 |   // See Equation 4.35, Resolution ℛ
60 |   inline mw_t operator()(::std::size_t mw) const {
61 |     return ((mw_t) mw) / R;
62 |   }
63 | };
64 | //
```

### 6.1.1.3. Implementation of *Combination* of *Amino Acid Residue*

This class is the implementation of Definition 4.1.1.5, *Combination* of *Amino Acid Residue*.

```
65 | class combination {
66 |   // The best fit for ℕ0 is ::std::size_t. And the mapping for each Amino Acid Residue is done by positional
         access based on the order.
67 |   ::std::vector<::std::size_t> V;
68 | public:
69 |   // Defaults to the empty combination with 0 amino acids.
70 |   inline combination() : V(aa_list.size(), 0uL) {}
71 |   // Default copy constructor.
72 |   inline combination(const combination &other) : V(other.V) {}
73 |   // Default destructor.
74 |   inline ~combination() {}
75 |   // Implementation of Definition 4.1.1.6, Molecular Weight of a Combination of Amino Acid Residue.
76 |   inline mw_t mw(void) const {
77 |     mw_t result = 0;
78 |     ::std::size_t i = 0;
79 |     for (const auto &aa : aa_list) result += V[i++] * aa.second;
80 |     return result;
81 |   }
82 |   // Gets the number of amino acids on this combination.
83 |   inline ::std::size_t count(void) const {
84 |     mw_t result = 0;
85 |     ::std::size_t i = 0;
86 |     for (const auto &aa : aa_list) result += V[i++];
87 |     return result;
88 |   }
89 |   // Implementation of Definition 4.1.1.7, Ambiguity of a Combination of Amino Acid Residue.
90 |   inline ::std::size_t ambiguity(void) const {
91 |     ::std::size_t result = fact(count());
92 |     for (int i = 0; i < aa_list.size(); i++) result /= fact(V[i]);
93 |     return result;
94 |   }
95 |   // Support for Definition 2.2.2.7, Cayley Graph and Definition 2.2.2.6, generation setIt is based on
         Theorem 4.1.1.2, G.
96 |   void step(::std::vector<combination> &next) const {
97 |     ::std::vector<::std::size_t> V(this->V);
98 |     for (::std::size_t i = 0; i < aa_list.size(); i++) {
99 |       if (i) V[i - 1]--;
100|       V[i]++;
101|       next.push_back(V);
102|     }
103|   }
104|   // Support operator for comparison
105|   bool operator==(const combination &other) const { return V == other.V; }
106|   // Support operator for comparison
107|   bool operator!=(const combination &other) const { return V != other.V; }
108|   // Prints the current spectrum in R list format.
109|   template<typename charT>
110|   friend ::std::basic_ostream<charT> &
111|   operator<<(::std::basic_ostream<charT> &out, const combination &c) {
112|     for (int i = 0; i < aa_list.size(); i++) {
113|       if (i) out << ' ';
114|       out << c.V[i];
115|     }
116|     return out;
117|   }
118|   // File output printing as a single line of numbers.
119|   template<typename charT>
120|   friend ::std::basic_ostream<charT> &
121|   operator%(::std::basic_ostream<charT> &out, const combination &c) {
122|     out << '{' << ' ';
123|     out << c.count() << ',' << ' ';
```

```
124 |     out << c.ambiguity() << ',' << ' ';
125 |     out << '{' << ' ';
126 |     for (int i = 0; i < aa_list.size(); i++) {
127 |       if (i) out << ',' << ' ';
128 |       out << c.V[i];
129 |     }
130 |     out << ' ' << '}';
131 |     out << ' ' << '}';
132 |     return out;
133 |   }
134 |   // Prints the current spectrum in R list format.
135 |   template<typename charT>
136 |   friend ::std::basic_istream<charT> &
137 |   operator>>(::std::basic_istream<charT> &in, combination &c) {
138 |     if (in.good() && !in.eof())
139 |       for (int i = 0; i < aa_list.size(); i++)
140 |         in >> c.V[i];
141 |     if (in.good() && !in.eof()) in.get();
142 |     return in;
143 |   }
144 | private:
145 |   // Support for step method.
146 |   inline combination(::std::vector<::std::size_t> v) : V(v) {}
147 | };
148 | // Support for step method, sorting based on the amount of amino acids.
149 | typedef struct {
150 |   bool operator()(const combination &a, const combination &b) const {
151 |     return a.count() < b.count();
152 |   }
153 | } combination_sort_by_count;
154 | //
```

### 155 | 6.1.1.4. Implementation of *Combinations* of a *Distance* at Resolution $\mathcal{R}$

```
156 | class point {
157 | public:
158 |   typedef ::std::set<combination, combination_sort_by_count> combinations_t;
159 |   typedef typename combinations_t::const_iterator const_iterator;
160 |   static constexpr ::std::size_t src_base() {
161 |     return sizeof(src_guards) / sizeof(src_guards[0]);
162 |   };
163 |   // Level of parallelism allowed for the generation of files.
164 |   static ::std::mutex src_guards[100];
165 |   // Constructor with the scaled mw
166 |   explicit inline point(
167 |       const arguments &arguments,
168 |       ::std::size_t scaled_mw) :
169 |       arguments_(arguments),
170 |       scaled_mw_(scaled_mw),
171 |       mw_(arguments_(scaled_mw)),
172 |       data_(get_data_path(arguments, scaled_mw_)),
173 |       src_(get_src_path(arguments, scaled_mw_)),
174 |       loaded_(false) {}
175 |   // Default constructor to 0da as scaled mw.
176 |   explicit inline point(const arguments &arguments, mw_t mw = 0) :
177 |       arguments_(arguments),
178 |       scaled_mw_(arguments(mw)),
179 |       mw_(mw),
180 |       data_(get_data_path(arguments, scaled_mw_)),
181 |       src_(get_src_path(arguments, scaled_mw_)),
182 |       loaded_(false) {}
183 |   // Prints the current spectrum in R list format.
184 |   friend point &operator<<(point &p, const combination &c) {
185 |     int index = 0;
186 |     {
187 |       ::std::fstream out(
188 |           p.data_.c_str(),
189 |           ::std::fstream::out | ::std::fstream::app);
```

```
190 |       ::std::lock_guard<::std::mutex>
191 |           lock(src_guards[p.scaled_mw_ % src_base()]);
192 |       out << ' ' << ::std::endl << c;
193 |     }
194 |     return p;
195 |   }
196 |   // Support for source file generation, target file name.
197 |   inline const fs::path &src(void) const { return src_; }
198 |   // Print the current point to the target file, and updates the mins.
199 |   void print_code(const ::std::vector<::std::size_t> &mins);
200 |   // Support for iterator traverse
201 |   inline const_iterator begin() const {
202 |     load_();
203 |     return combinations_.cbegin();
204 |   }
205 |   // Support for iterator traverse
206 |   inline const_iterator end() const {
207 |     load_();
208 |     return combinations_.cend();
209 |   }
210 |   // Implementation of Definition 4.1.1.10, Ambiguity of a Distance at Resolution R.
211 |   inline ::std::size_t permutations(void) const {
212 |     load_();
213 |     return permutations_;
214 |   }
215 |   // Implementation of Definition 4.1.1.10, Ambiguity of a Distance at Resolution R.
216 |   inline ::std::size_t combinations(void) const {
217 |     load_();
218 |     return combinations_.size();
219 |   }
220 | private:
221 |   const arguments &arguments_;
222 |   const ::std::size_t scaled_mw_;
223 |   const mw_t mw_;
224 |   const fs::path data_;
225 |   const fs::path src_;
226 |   mutable bool loaded_;
227 |   mutable combinations_t combinations_;
228 |   mutable ::std::size_t permutations_;
229 |   // Opens the data file and load all combinations on a set.
230 |   inline void load_() const {
231 |     if (!loaded_) {
232 |       loaded_ = true;
233 |       ::std::fstream in(data_.string(), ::std::fstream::in);
234 |       while (in.good() && !in.eof()) {
235 |         combination c;
236 |         in >> c;
237 |         combinations_.insert(c);
238 |       }
239 |       permutations_ = 0;
240 |       for (const combination &c : combinations_) {
241 |         permutations_ += c.ambiguity();
242 |       }
243 |     }
244 |   }
245 |   // Construction of the data file name.
246 |   static inline fs::path
247 |   get_data_path(const arguments &arguments, ::std::size_t scaled_mw) {
248 |     static const int base = 10;
249 |     fs::path result = arguments.wd / ::std::string("data");
250 |     for (; scaled_mw > 0; scaled_mw /= base) {
251 |       result /= ::std::to_string(scaled_mw % base);
252 |     }
253 |     result /= ::std::to_string(0);
254 |     fs::create_directories(result);
255 |     result /= ::std::string("data.json");
256 |     return result;
257 |   }
258 |   // Construction of the target source file name.
259 |   static inline fs::path
```

```
260 |   get_src_path(const arguments &arguments, ::std::size_t scaled_mw) {
261 |     fs::path result = arguments.wd / ::std::string("src");
262 |     result /= ::std::to_string(scaled_mw % (src_base())) + ".cc";
263 |     return result;
264 |   }
265 |};
266 |}
267 |}
268 |#endif //TMS2_TMS2INDEX_H
```

**Listing 6.2:** `tpool.h`: Thread pool header

```
1 |//
```

## 2 | 6.1.2. Multi-threading support

```
3 |#ifndef TMS2_TPOOL_H
4 |#define TMS2_TPOOL_H
5 |#include <vector>
6 |#include <queue>
7 |#include <memory>
8 |#include <thread>
9 |#include <mutex>
10|#include <condition_variable>
11|#include <future>
12|#include <functional>
13|#include <stdexcept>
14|namespace tms2 {
15|inline namespace threading {
16|class tpool {
17|public:
18|  // Constructor with the number of threads.
19|  tpool(::std::size_t);
20|  // Enqueue tasks
21|  template<class F, class... Args>
22|  auto
23|  enqueue(F &&f, Args &&... args)
24|  -> ::std::future<typename ::std::result_of<F(Args...)>::type>;
25|  // Default destructor
26|  ~tpool();
27|private:
28|  // Need to keep track of threads so we can join them
29|  ::std::vector<::std::thread> workers;
30|  // Task queue
31|  ::std::queue<::std::function<void()> > tasks;
32|  // Synchronization
33|  ::std::mutex queue_mutex;
34|  ::std::condition_variable condition;
35|  bool stop;
36|};
37|// The constructor just launches some amount of workers
38|inline tpool::tpool(::std::size_t threads) : stop(false) {
39|  for (::std::size_t i = 0; i < threads; ++i)
40|    workers.emplace_back(
41|        [this] {
42|          for (;;) {
43|            ::std::function<void()> task;
44|            {
45|              ::std::unique_lock<std::mutex> lock(
46|                  this->queue_mutex);
47|              this->condition.wait(
48|                  lock,
49|                  [this] {
50|                    return this->stop ||
51|                           !this->tasks.empty();
52|                  });
53|              if (this->stop && this->tasks.empty()) return;
54|              task = ::std::move(this->tasks.front());
55|              this->tasks.pop();
56|            }
57|            task();
58|          }
59|        }
60|    );
61|}
62|// add new work item to the pool
63|template<class F, class... Args>
```

```
64 |auto tpool::enqueue(F &&f, Args &&... args)
65 |-> ::std::future<typename ::std::result_of<F(Args...)>::type> {
66 |  using return_type = typename ::std::result_of<F(Args...)>::type;
67 |  auto task = ::std::make_shared<std::packaged_task<return_type()> >(
68 |      ::std::bind(::std::forward<F>(f),
69 |                  ::std::forward<Args>(args)...)
70 |  );
71 |  ::std::future<return_type> res = task->get_future();
72 |  {
73 |    ::std::unique_lock<std::mutex> lock(queue_mutex);
74 |    // don't allow enqueueing after stopping the pool
75 |    if (stop) throw ::std::runtime_error("enqueue on stopped tpool");
76 |    tasks.emplace([task]() { (*task)(); });
77 |  }
78 |  condition.notify_one();
79 |  return res;
80 |}
81 |// the destructor joins all threads
82 |inline tpool::~tpool() {
83 |  {
84 |    ::std::unique_lock<::std::mutex> lock(queue_mutex);
85 |    stop = true;
86 |  }
87 |  condition.notify_all();
88 |  for (::std::thread &worker: workers) worker.join();
89 |}
90 |}
91 |}
92 |#endif
```

**Listing 6.3:** `tms2index.cpp`: Index generation source

```
//
```

### 6.1.3. Index generation source

```
#include <unistd.h>
#include <iostream>
#include <thread>
#include <tms2/printer.h>
#include <tms2/tms2index.h>
#include <tms2/resource.h>
#include <tms2/tpool.h>
#include <tms2/trie.h>
using namespace tms2;
using namespace tms2::cpp;
using namespace tms2::indexgen;
using namespace tms2::threading;
static const auto aav = "aav"_type;
static const auto aap = "aap"_type;
static const auto size_t_name = "::std::size_t"_type;
static const auto ostream = "::std::basic_ostream<charT>"_type;
static const auto other = "other"_local;
static const fs::path root = "tms2";
static const fs::path da_h = root / "da.h";
void progress_bar(float progress) {
  int barWidth = 70;
  ::std::cerr << "# [";
  int pos = barWidth * progress;
  for (int i = 0; i < barWidth; ++i) {
    if (i < pos) ::std::cerr << "=";
    else if (i == pos) ::std::cerr << ">";
    else ::std::cerr << " ";
  }
  ::std::cerr << "] " << ::std::fixed << ::std::setw(7)
              << ::std::setprecision(3)
              << progress * 100.0 << " %\r";
  ::std::cerr.flush();
}
//
```

#### 6.1.3.1. `project` source code generator

```
class project {
  const arguments arg_;
  const ::std::size_t limit_;
  ::std::fstream aah_h;
  ::std::fstream cmakelists_txt;
  ::std::fstream index_h;
  ::std::fstream index_cc;
  ::std::set<fs::path> sources_;
public:
  project(const arguments &arg) :
      arg_(arg),
      limit_(arg.M + arg.B + 1) {
  }
  //
```

#### 6.1.3.2. Project file definition

```
  void init() {
    // Remove previous execution if there is one.
    fs::remove_all(arg_.wd);
    // Create folder is it does not exist.
    fs::create_directories(arg_.wd);
    fs::create_directories(arg_.wd / "data");
    fs::create_directories(arg_.wd / "include");
    fs::create_directories(arg_.wd / "include" / "tms2");
    fs::create_directories(arg_.wd / "src");
    fs::create_directories(arg_.wd / "bin");
```

```
61 |     // Change to temporary folder.
62 |     chdir(arg_.wd.c_str());
63 |     ::std::vector<::std::pair<::std::fstream &, ::fs::path>> files = {
64 |         {cmakelists_txt, arg_.wd / "CMakeLists.txt"},
65 |         {aah_h,          arg_.wd / "include" / da_h},
66 |         {index_h,        arg_.wd / "src" / "index.h"},
67 |         {index_cc,       arg_.wd / "src" / "index.cc"}
68 |     };
69 |     for (const ::std::pair<::std::fstream &, ::fs::path> &file : files) {
70 |       file.first.open(file.second.c_str(),
71 |                       ::std::fstream::out | ::std::fstream::app);
72 |       if (!file.first.good()) {
73 |         ::std::cerr << "Unable to create aah.h file" << ::std::endl;
74 |       }
75 |     }
76 |     preCMakeFile();
77 |   }
78 |   //
```

#### 6.1.3.3. Implementation of Algorithm 0, Build Index

```
79 |   void caleyWalk() {
80 |     float minMw = ::std::numeric_limits<float>::max();
81 |     for (const auto &aa : aa_list) {
82 |       minMw = ::std::min(aa.second, minMw);
83 |     }
84 |     const ::std::size_t minMwInt = arg_(minMw);
85 |     typedef enum {
86 |       start,
87 |       data,
88 |       code
89 |     } progress_t;
90 |     ::std::vector<progress_t> mw_progress(limit_ + minMwInt + 1, start);
91 |     ::std::vector<::std::size_t> mins(
92 |         limit_ + 1,
93 |         ::std::numeric_limits<::std::size_t>::max());
94 |     // Start point Init the zero.
95 |     {
96 |       combination czero;
97 |       point pzero(arg_);
98 |       pzero << czero;
99 |       mins[0] = 0;
100|       for (::std::size_t mw = 0; mw <= minMwInt; mw++)
101|         mw_progress[mw] = progress_t::code;
102|     }
103|     {
104|       ::tms2::tpool pool(::std::thread::hardware_concurrency());
105|       ::std::cerr << "# Generating data " << ::std::endl;
106|       progress_bar(0.0);
107|       float progress = 0.0;
108|       for (::std::size_t mw = 0; mw <= limit_; mw++) {
109|         while (mw_progress[mw] == progress_t::start)
110|           ::std::this_thread::yield();
111|         pool.enqueue(
112|             [this, mw, minMwInt, &mw_progress, &mins] {
113|               point p(arg_, mw);
114|               for (const combination &i : p) {
115|                 ::std::vector<combination> next;
116|                 i.step(next);
117|                 for (const combination &j: next) {
118|                   point q(this->arg_, j.mw());
119|                   q << j;
120|                 }
121|               }
122|               const ::std::size_t permutations = p.permutations();
123|               if (permutations) {
124|                 mins[mw] = permutations;
125|               }
126|               mw_progress[mw + minMwInt + 1] = progress_t::data;
127|               mw_progress[mw] = progress_t::code;
```

```
128 |               for (::std::size_t i =
129 |                   (size_t) ::std::max(0L, (long) mw - 2 * arg_.B + 1);
130 |                    i < mw; i++)
131 |                 while (mw_progress[i] != progress_t::code)
132 |                   ::std::this_thread::yield();
133 |               p.print_code(mins);
134 |             });
135 |         //
136 |         for (float p = ::std::exp(-(2.0 + limit_ - mw) / minMwInt);
137 |              p - progress > 0.001;
138 |              p = progress) {
139 |           progress_bar(p);
140 |         }
141 |       }
142 |     }
143 |     progress_bar(1.0);
144 |     ::std::cerr << ::std::endl << "# Done " << ::std::endl;
145 |   }
146 |   //
```

#### 6.1.3.4. da header file generation

```
147 |   void print_aah_h() {
148 |     ::std::size_t level = 0;
149 |     const ::std::string guard = "__tms2_aa__";
150 |     Setup(aah_h, level);
151 |     Out << _IFNDEF << guard << ::std::endl;
152 |     Out << _DEFINE << guard << ::std::endl;
153 |     Out << ::std::endl;
154 |     for (const ::std::string file :
155 |         {"cstring", "new", "cstring", "new", "cmath", "limits",
156 |          "algorithm", "set", "iostream", "iomanip"}) {
157 |       Out << _INCLUDE;
158 |       SBrks { Out << file; }
159 |       Out << ::std::endl;
160 |     }
161 |     Out << ::std::endl;
162 |     Out << _namespace << "tms2";
163 |     Brcs {
164 |       print_aa();
165 |       print_aav();
166 |       print_aap();
167 |       print_da();
168 |     };
169 |     Out << ::std::endl;
170 |     Out << _ENDIF << ::std::endl;
171 |     if (!arg_.prefix.empty()) {
172 |       aah_h.close();
173 |       const fs::path from = arg_.wd / "include" / da_h;
174 |       const fs::path to = arg_.prefix / "include" / da_h;
175 |       create_directories(to.parent_path());
176 |       fs::copy_file(from, to, fs::copy_option::overwrite_if_exists);
177 |     }
178 |   }
179 |   //
```

#### 6.1.3.5. da class generation

```
180 |   void print_da() {
181 |     ::std::size_t level = 0;
182 |     Setup(aah_h, level);
183 |     const auto ida = "ida"_type;
184 |     const auto fda = "fda"_type;
185 |     const auto da = "da"_type;
186 |     const auto fresolution = "fresolution"_member_public;
187 |     const auto iresolution = "iresolution"_member_public;
188 |     const auto fmin = "fmin"_member_public;
189 |     const auto fmax = "fmax"_member_public;
190 |     const auto imin = "imin"_member_public;
191 |     const auto imax = "imax"_member_public;
192 |     const auto iepsilon = "iepsilon"_member_public;
193 |     const auto fepsilon = "fepsilon"_member_public;
194 |     const auto mradius = "mradius"_member_public;
```

```
195     const auto __min_minimuns = "min_minimuns"_member;
196     const auto __max_minimuns = "max_minimuns"_member;
197     const auto min_minimuns = "min_minimuns"_member_public;
198     const auto max_minimuns = "max_minimuns"_member_public;
199     const auto ivalue = "ivalue"_member_public;
200     const auto fvalue = "fvalue"_member_public;
201     const auto tcombinations = "tcombinations"_member_public;
202     const auto tpermutations = "tpermutations"_member_public;
203     const auto ccombinations = "ccombinations"_member_public;
204     const auto cpermutations = "cpermutations"_member_public;
205     const auto lcombinations = "lcombinations"_member_public;
206     const auto minimuns = "minimuns"_member_public;
207     const auto zero = "zero"_member_public;
208     const auto other = "other"_local;
209     const auto index = "index"_member;
210     Stmt(_typedef << _int << ida);
211     Stmt(_typedef << _float << fda);
212     Tab << _struct << da;
213     Brcs {
214       Tab << _static << _constexpr << fda << fresolution;
215       Pars {}
216       Brcs { Stmt(_return << 1.0 / arg_.R); }
217       Tab << _static << _constexpr << ida << iresolution;
218       Pars {}
219       Brcs { Stmt(_return << 1 * arg_.R); }
220       Tab << _static << _constexpr << fda << fmin;
221       Pars {}
222       Brcs { Stmt(_return << 0.0f); }
223       Tab << _static << _constexpr << fda << fmax;
224       Pars {}
225       Brcs { Stmt(_return << (float) arg_.M); }
226       Tab << _static << _constexpr << ida << imin;
227       Pars {}
228       Brcs { Stmt(_return << 0); }
229       Tab << _static << _constexpr << ida << imax;
230       Pars {}
231       Brcs { Stmt(_return << arg_.M); }
232       Tab << _static << _constexpr << fda << fepsilon;
233       Pars {}
234       Brcs { Stmt(_return << 1.0 / arg_.R); }
235       Tab << _static << _constexpr << ida << iepsilon;
236       Pars {}
237       Brcs { Stmt(_return << 1); }
238       Tab << _static << _constexpr << ida << mradius;
239       Pars {}
240       Brcs { Stmt(_return << arg_.B); }
241       Tab << _static << _const << ida << __min_minimuns;
242       Brks { ArgN << arg_.B + 1; }
243       Out << _semicolon;
244       Tab << _static << _const << ida << __max_minimuns;
245       Brks { ArgN << arg_.B + 1; }
246       Out << _semicolon;
247       Tab << _static << _const << ida << '*' << min_minimuns;
248       Pars {}
249       Brcs {
250         Stmt(_return << '(' << ida << '*' << ')' << __min_minimuns);
251       }
252       Tab << _static << _const << ida << '*' << max_minimuns;
253       Pars {}
254       Brcs {
255         Stmt(_return << '(' << ida << '*' << ')' << __max_minimuns);
256       }
257       Tab << _static << _const << da << '*' << index;
258       Brks { Out << arg_.M + arg_.B + 2; };
259       Out << _semicolon;
260       Stmt(_const << ida << ivalue);
261       Stmt(_const << fda << fvalue);
262       Stmt(_const << size_t_name << tcombinations);
263       Stmt(_const << size_t_name << tpermutations);
264       Stmt(_const << size_t_name << ccombinations);
265       Stmt(_const << size_t_name << cpermutations);
```

```
266 |       Stmt(_const << aap << '*' << _const << lcombinations);
267 |       Stmt(_const << ida << '*' << _const << minimuns);
268 |       // Default constructor
269 |       Tab << _inline << da;
270 |       Pars {};
271 |       Out << _space << _noexcept << _colon;
272 |       Init {
273 |         ArgNL << ivalue;
274 |         Pars { Out << 0; }
275 |         ArgNL << fvalue;
276 |         Pars { Out << 0; }
277 |         ArgNL << tcombinations;
278 |         Pars { Out << 1; }
279 |         ArgNL << tpermutations;
280 |         Pars { Out << 1; }
281 |         ArgNL << ccombinations;
282 |         Pars { Out << 1; }
283 |         ArgNL << cpermutations;
284 |         Pars { Out << 1; }
285 |         ArgNL << lcombinations;
286 |         Pars { Out << _bitand << aap << _scope << "zero"; }
287 |         ArgNL << minimuns;
288 |         Pars {
289 |           Out << min_minimuns;
290 |           Pars {}
291 |         }
292 |       };
293 |       Brcs {}
294 |       // Value constructor
295 |       Tab << _constexpr << _inline << da;
296 |       Pars {
297 |         ArgNL << _const << ida << ivalue.to_local();
298 |         ArgNL << _const << fda << fvalue.to_local();
299 |         ArgNL << _const << size_t_name << tcombinations.to_local();
300 |         ArgNL << _const << size_t_name << tpermutations.to_local();
301 |         ArgNL << _const << size_t_name << ccombinations.to_local();
302 |         ArgNL << _const << size_t_name << cpermutations.to_local();
303 |         ArgNL << _const << aap << '*' << _const
304 |               << lcombinations.to_local();
305 |         ArgNL << _const << ida << '*' << _const
306 |               << minimuns.to_local();
307 |       };
308 |       Out << _space << _noexcept << _colon;
309 |       Init {
310 |         ArgNL << ivalue;
311 |         Pars { Out << ivalue.to_local(); }
312 |         ArgNL << fvalue;
313 |         Pars { Out << fvalue.to_local(); }
314 |         ArgNL << tcombinations;
315 |         Pars { Out << tcombinations.to_local(); }
316 |         ArgNL << tpermutations;
317 |         Pars { Out << tpermutations.to_local(); }
318 |         ArgNL << ccombinations;
319 |         Pars { Out << ccombinations.to_local(); }
320 |         ArgNL << cpermutations;
321 |         Pars { Out << cpermutations.to_local(); }
322 |         ArgNL << lcombinations;
323 |         Pars { Out << lcombinations.to_local(); }
324 |         ArgNL << minimuns;
325 |         Pars { Out << minimuns.to_local(); }
326 |       };
327 |       Brcs {}
328 |       // Copy constructor
329 |       Tab << _constexpr << _inline << da;
330 |       Pars {
331 |         ArgNL << _const << da << _bitand << other;
332 |       };
333 |       Out << _space << _noexcept << _colon;
334 |       Init {
335 |         ArgNL << ivalue;
```

```
336 |         Pars { Out << other << _dot << ivalue; }
337 |         ArgNL << fvalue;
338 |         Pars { Out << other << _dot << fvalue; }
339 |         ArgNL << tcombinations;
340 |         Pars { Out << other << _dot << tcombinations; }
341 |         ArgNL << tpermutations;
342 |         Pars { Out << other << _dot << tpermutations; }
343 |         ArgNL << ccombinations;
344 |         Pars { Out << other << _dot << ccombinations; }
345 |         ArgNL << cpermutations;
346 |         Pars { Out << other << _dot << cpermutations; }
347 |         ArgNL << lcombinations;
348 |         Pars { Out << other << _dot << lcombinations; }
349 |         ArgNL << minimuns;
350 |         Pars { Out << other << _dot << minimuns; }
351 |       };
352 |       Brcs {}
353 |       // Float constructor
354 |       const auto f = "f"_local;
355 |       const auto scale = "scale"_member_public;
356 |       Tab << _inline << da;
357 |       Pars {
358 |         ArgNL << _const << fda << f;
359 |       };
360 |       Out << _space << _noexcept << _colon;
361 |       Init {
362 |         ArgNL << ivalue;
363 |         Pars { Out << scale << par(f); }
364 |         ArgNL << fvalue << par(f);
365 |         ArgNL << tcombinations;
366 |         Pars {
367 |           Out << index;
368 |           Brks {
369 |             Out << _scope << "std" << _scope << "min";
370 |             Pars {
371 |               ArgN << ivalue;
372 |               ArgN << imax << par();
373 |             };
374 |           }
375 |           Out << _access << tcombinations;
376 |         }
377 |         ArgNL << tpermutations;
378 |         Pars {
379 |           Out << index;
380 |           Brks {
381 |             Out << _scope << "std" << _scope << "min";
382 |             Pars {
383 |               ArgN << ivalue;
384 |               ArgN << imax << par();
385 |             };
386 |           }
387 |           Out << _access << tpermutations;
388 |         }
389 |         ArgNL << ccombinations;
390 |         Pars {
391 |           Out << index;
392 |           Brks {
393 |             Out << _scope << "std" << _scope << "min";
394 |             Pars {
395 |               ArgN << ivalue;
396 |               ArgN << imax << par();
397 |             };
398 |           }
399 |           Out << _access << ccombinations;
400 |         }
401 |         ArgNL << cpermutations;
402 |         Pars {
403 |           Out << index;
404 |           Brks {
405 |             Out << _scope << "std" << _scope << "min";
406 |             Pars {
```

```
407 |               ArgN << ivalue;
408 |               ArgN << imax << par();
409 |             };
410 |           }
411 |           Out << _access << cpermutations;
412 |         }
413 |         ArgNL << lcombinations;
414 |         Pars {
415 |           Out << index;
416 |           Brks {
417 |             Out << _scope << "std" << _scope << "min";
418 |             Pars {
419 |               ArgN << ivalue;
420 |               ArgN << imax << par();
421 |             };
422 |           }
423 |           Out << _access << lcombinations;
424 |         }
425 |         ArgNL << minimuns;
426 |         Pars {
427 |           Out << index;
428 |           Brks {
429 |             Out << _scope << "std" << _scope << "min";
430 |             Pars {
431 |               ArgN << ivalue;
432 |               ArgN << imax << par();
433 |             };
434 |           }
435 |           Out << _access << minimuns;
436 |         }
437 |       }
438 |       Brcs {}
439 |       // Int constructor
440 |       const auto i = "i"_local;
441 |       Tab << _inline << da;
442 |       Pars {
443 |         ArgNL << _const << ida << i;
444 |       };
445 |       Out << _space << _noexcept << _colon;
446 |       Init {
447 |         ArgNL << ivalue << par(i);
448 |         ArgNL << fvalue;
449 |         Pars { Out << scale << par(i); }
450 |         ArgNL << tcombinations;
451 |         Pars {
452 |           Out << index;
453 |           Brks {
454 |             Out << _scope << "std" << _scope << "min";
455 |             Pars {
456 |               ArgN << ivalue;
457 |               ArgN << imax << par();
458 |             };
459 |           }
460 |           Out << _access << tcombinations;
461 |         }
462 |         ArgNL << tpermutations;
463 |         Pars {
464 |           Out << index;
465 |           Brks {
466 |             Out << _scope << "std" << _scope << "min";
467 |             Pars {
468 |               ArgN << ivalue;
469 |               ArgN << imax << par();
470 |             };
471 |           }
472 |           Out << _access << tpermutations;
473 |         }
474 |         ArgNL << ccombinations;
475 |         Pars {
476 |           Out << index;
```

```
477 |           Brks {
478 |             Out << _scope << "std" << _scope << "min";
479 |             Pars {
480 |               ArgN << ivalue;
481 |               ArgN << imax << par();
482 |             };
483 |           }
484 |           Out << _access << ccombinations;
485 |         }
486 |         ArgNL << cpermutations;
487 |         Pars {
488 |           Out << index;
489 |           Brks {
490 |             Out << _scope << "std" << _scope << "min";
491 |             Pars {
492 |               ArgN << ivalue;
493 |               ArgN << imax << par();
494 |             };
495 |           }
496 |           Out << _access << cpermutations;
497 |         }
498 |         ArgNL << lcombinations;
499 |         Pars {
500 |           Out << index;
501 |           Brks {
502 |             Out << _scope << "std" << _scope << "min";
503 |             Pars {
504 |               ArgN << ivalue;
505 |               ArgN << imax << par();
506 |             };
507 |           }
508 |           Out << _access << lcombinations;
509 |         }
510 |         ArgNL << minimuns;
511 |         Pars {
512 |           Out << index;
513 |           Brks {
514 |             Out << _scope << "std" << _scope << "min";
515 |             Pars {
516 |               ArgN << ivalue;
517 |               ArgN << imax << par();
518 |             };
519 |           }
520 |           Out << _access << minimuns;
521 |         }
522 |       }
523 |       Brcs {}
524 |       // Scale
525 |       const auto value = "value"_local;
526 |       const auto index = "index"_member;
527 |       Tab << _static << _constexpr << _inline << ida << scale;
528 |       Pars { Out << _const << _double << value; }
529 |       Brcs { Stmt(_return << value << '/' << fresolution << par()); }
530 |       Tab << _static << _constexpr << _inline << ida << scale;
531 |       Pars { Out << _const << fda << value; }
532 |       Brcs { Stmt(_return << value << '/' << fresolution << par()); }
533 |       Tab << _static << _constexpr << _inline << fda << scale;
534 |       Pars { Out << _const << ida << value; }
535 |       Brcs { Stmt(_return << value << '*' << fresolution << par()); }
536 |       // Operator =
537 |       Tab << _inline << da << _bitand << _operator << _set;
538 |       Pars { Out << _const << da << _bitand << other; }
539 |       Brcs {
540 |         Stmt(_return << '*' << _new << par("this") << da << par(other));
541 |       }
542 |       Tab << _inline << da << _bitand << _operator << _set;
543 |       Pars { Out << _const << fda << _bitand << other; }
544 |       Brcs {
545 |         Stmt(_return << '*' << _new << par("this") << da << par(other));
546 |       }
```

```
547 |       Tab << _inline << da << _bitand << _operator << _set;
548 |       Pars { Out << _const << ida << _bitand << other; }
549 |       Brcs {
550 |         Stmt(_return << '*' << _new << par("this") << da << par(other));
551 |       }
552 |       // Aritmethic operators
553 |       const ::std::vector<::std::string> operators = {"+", "-", "/", "*"};
554 |       const auto a = "a"_local;
555 |       const auto b = "b"_local;
556 |       for (const auto op : operators) {
557 |         Tab << _friend << da << _operator << op;
558 |         Pars {
559 |           ArgNL << da << a;
560 |           ArgNL << _const << da << _bitand << b;
561 |         }
562 |         Brcs { Stmt(_return << a << op << '=' << b); }
563 |         Tab << _inline << da << _bitand << _operator << op << '=';
564 |         Pars { ArgN << _const << da << _bitand << other; }
565 |         Brcs {
566 |           Tab << _return << '*' << _this << _set << '*' << index;
567 |           Brks {
568 |             Out << scale;
569 |             Pars {
570 |               ArgN << fvalue << op << other << _dot << fvalue;
571 |             };
572 |           }
573 |           Out << _semicolon;
574 |         }
575 |       }
576 |       // Operator bool()
577 |       Tab << _inline << _operator << _bool << par() << _const;
578 |       Brcs {
579 |         Stmt(_return << lcombinations << _or << fvalue << _gte << fmax
580 |                      << par());
581 |       }
582 |       // Operator []
583 |       Tab << _inline << da << _operator;
584 |       Brks {};
585 |       Pars { Out << ida << value; }
586 |       Out << _const;
587 |       Brcs {
588 |         Tab << _return << '*' << index;
589 |         Brks {
590 |           Out << index;
591 |           Brks {
592 |             Out << _scope << "std" << _scope << "min";
593 |             Pars {
594 |               ArgN << ivalue << '+' << mradius << par();
595 |               ArgN << imax << par();
596 |             }
597 |           }
598 |           Out << _access << minimuns;
599 |           Brks {
600 |             Out << _scope << "std" << _scope << "min";
601 |             Pars {
602 |               ArgN << value;
603 |               ArgN << mradius << par();
604 |             }
605 |           }
606 |         }
607 |         Out << _semicolon;
608 |       }
609 |       // Operator ida, fda
610 |       Tab << _inline << _operator << fda << par() << _const;
611 |       Brcs { Stmt(_return << fvalue); }
612 |       Tab << _inline << _operator << ida << par() << _const;
613 |       Brcs { Stmt(_return << ivalue); }
614 |       // Operator <<
615 |       Tab << _template;
616 |       const auto charT = "charT"_type;
```

```
617 |       SBrks { Out << _typename << charT; }
618 |       Tab << _friend << _scope << "std" << _scope << "basic_ostream";
619 |       SBrks { Out << charT; }
620 |       Out << _bitand << _operator << _ltlt;
621 |       Pars {
622 |         ArgNL << _scope << "std" << _scope << "basic_ostream";
623 |         SBrks { Out << charT; }
624 |         Out << _bitand << "out";
625 |         ArgNL << _const << da << _bitand << value;
626 |       }
627 |       Brcs {
628 |         Stmt("out" << _ltlt << value << _dot << fvalue);
629 |         Stmt(_return << "out");
630 |       }
631 |       // Zero, min, max
632 |       Stmt(_static << _const << da << zero);
633 |       Stmt(_static << _const << da << "min");
634 |       Stmt(_static << _const << da << "max");
635 |     }
636 |     Out << _semicolon << ::std::endl;
637 |     {
638 |       const auto v = "v"_local;
639 |       Tab << _constexpr << ida << _operator << '"' << '"' << '_' << 'd' << 'a';
640 |       Pars { Out << _long << _double << v; }
641 |       Brcs {
642 |         Tab << _return << da << _scope << "scale";
643 |         Pars { Out << '(' << _float << ')' << v; };
644 |         Out << _semicolon;
645 |       }
646 |     }
647 |     Out << ::std::endl;
648 |   }
649 |   void print_aa() {
650 |     ::std::size_t level = 0;
651 |     Setup(aah_h, level);
652 |     Tab << _typedef << _enum;
653 |     Brcs Init {
654 |         ArgNL << 'n' << 'o' << 'n' << 'e' << _set << 0;
655 |         for (const auto &aa: aa_list) {
656 |           ArgNL << ' ';
657 |           for (const char c : aa.first) {
658 |             if (::std::isalnum(c)) {
659 |               Out << c;
660 |             } else {
661 |               Out << '_';
662 |             }
663 |           }
664 |         }
665 |       }
666 |     Out << "residue_t" << _semicolon;
667 |     Stmt(_typedef << _typename << "::std::basic_string<residue_t> "
668 |                   << "peptide_t");
669 |   }
670 |   //
```

#### 6.1.3.6. Code generation for Definition 4.1.1.5, *Combination* of *Amino Acid Residue*

```
671 |   void print_aav() {
672 |     ::std::size_t level = 0;
673 |     Setup(aah_h, level);
674 |     Tab << _struct << aav;
675 |     Brcs {
676 |       Tab << _public << _colon << ::std::endl;
677 |       Ident {
678 |         for (const aa_list_t::value_type &aa: aa_list) {
679 |           Stmt(_const << _short << _public_member(aa.first));
680 |         }
681 |         Out << ::std::endl;
682 |         // Default constructor
683 |         Tab << _constexpr << _inline << aav;
```

```
684 |         Pars {}
685 |         Out << _space << _noexcept << _colon;
686 |         Init {
687 |           for (const aa_list_t::value_type &aa: aa_list) {
688 |             ArgNL << _public_member(aa.first) << par(0);
689 |           }
690 |         }
691 |         Brcs {}
692 |         Out << ::std::endl;
693 |         // Copy constructor
694 |         Tab << _constexpr << _inline << aav;
695 |         Pars { Out << _const << aav << _bitand << other; }
696 |         Out << _space << _noexcept << _colon;
697 |         Init {
698 |           for (const aa_list_t::value_type &aa: aa_list) {
699 |             ArgNL << _public_member(aa.first);
700 |             Pars {
701 |               Out << other << _dot << _public_member(aa.first);
702 |             }
703 |           }
704 |         }
705 |         Brcs {}
706 |         // Value constructor
707 |         Tab << _constexpr << _inline << aav;
708 |         Pars {
709 |           for (const aa_list_t::value_type &aa: aa_list) {
710 |             ArgNL << _short << _local(aa.first);
711 |           }
712 |         }
713 |         Out << _space << _noexcept << _colon;
714 |         Init {
715 |           for (const aa_list_t::value_type &aa: aa_list) {
716 |             ArgNL << _public_member(aa.first);
717 |             Pars {
718 |               Out << _local(aa.first);
719 |             }
720 |           }
721 |         }
722 |         Brcs {}
723 |         Out << ::std::endl;
724 |         Tab << _inline << aav << _bitand << _operator << _set;
725 |         Pars { Out << _const << aav << _bitand << other; }
726 |         Brcs {
727 |           Tab << _return << '*' << _new;
728 |           Pars { Out << _this; }
729 |           Out << aav;
730 |           Pars { Out << other; };
731 |           Out << _semicolon;
732 |         }
733 |         Out << ::std::endl;
734 |         Tab << _template;
735 |         SBrks { Out << _typename << "charT"; }
736 |         Tab << _friend << ostream << _bitand << _operator << _ltlt;
737 |         Pars {
738 |           ArgNL << ostream << _bitand << "out";
739 |           ArgNL << _const << aav << _bitand << "v";
740 |         }
741 |         Brcs {
742 |           Stmt(_int << "index" << _set << 0);
743 |           Stmt("out" << _ltlt << "'{'");
744 |           for (const aa_list_t::value_type &aa: aa_list) {
745 |             If("v" << _dot << aa.first) {
746 |               If("index" << _inc) {
747 |                 Stmt("out" << _ltlt << "'*'");
748 |               }
749 |               Stmt("out" << _ltlt << "'{'");
750 |               for (const auto &c : aa.first) {
751 |                 Stmt("out" << _ltlt << "'" << c << "'");
752 |               }
```

```
753 |               If("v" << _dot << aa.first << _gt << 1) {
754 |                 Stmt("out" << _ltlt << "'^'"
755 |                            << _ltlt
756 |                            << "v" << _dot
757 |                            << aa.first);
758 |               }
759 |               Stmt("out" << _ltlt << "'}'");
760 |             }
761 |           }
762 |           Stmt("out" << _ltlt << "'}'");
763 |           Stmt(_return << "out");
764 |         }
765 |       }
766 |     }
767 |     Out << _semicolon;
768 |     Out << ::std::endl;
769 |   }
770 |   void print_aap() {
771 |     ::std::size_t level = 0;
772 |     Setup(aah_h, level);
773 |     Tab << _struct << aap;
774 |     Brcs {
775 |       const auto mSymbols = "symbols"_member_public;
776 |       const auto aSymbols = "symbols"_local;
777 |       const auto mPermutations = "permutations"_member_public;
778 |       const auto aPermutations = "permutations"_local;
779 |       const auto mVector = "vector"_member_public;
780 |       const auto aVector = "vector"_local;
781 |       const auto other = "other"_local;
782 |       const auto zero = "zero"_member_public;
783 |       Tab << _public << _colon;
784 |       Ident {
785 |         Stmt(_const << size_t_name << mSymbols);
786 |         Stmt(_const << size_t_name << mPermutations);
787 |         Stmt(_const << aav << _space << mVector);
788 |         Out << ::std::endl;
789 |         Tab << _constexpr << _inline << aap;
790 |         Pars {
791 |           ArgNL << _const << size_t_name << aSymbols;
792 |           ArgNL << _const << size_t_name << aPermutations;
793 |           ArgNL << _const << aav << _space << aVector;
794 |         }
795 |         Out << _space << _noexcept << _colon;
796 |         Init {
797 |           ArgNL << mSymbols;
798 |           Pars { Out << aSymbols; }
799 |           ArgNL << mPermutations;
800 |           Pars { Out << aSymbols; }
801 |           ArgNL << mVector;
802 |           Pars { Out << aVector; }
803 |         }
804 |         Brcs {}
805 |         Out << ::std::endl;
806 |         Tab << _constexpr << _inline << aap;
807 |         Pars { Out << _const << aap << _bitand << other; }
808 |         Out << _space << _noexcept << _colon;
809 |         Init {
810 |           ArgNL << mSymbols;
811 |           Pars { Out << other << _dot << mSymbols; }
812 |           ArgNL << mPermutations;
813 |           Pars { Out << other << _dot << mSymbols; }
814 |           ArgNL << mVector;
815 |           Pars { Out << other << _dot << mVector; }
816 |         }
817 |         Brcs {}
818 |         Out << ::std::endl;
819 |         Tab << _constexpr << _inline << aap;
820 |         Pars {};
821 |         Out << _space << _noexcept << _colon;
822 |         Init {
823 |           ArgNL << mSymbols;
```

```
824 |           Pars { Out << 0; }
825 |           ArgNL << mPermutations;
826 |           Pars { Out << 1; }
827 |           //ArgNL << mVector;
828 |           //Pars { Out << aav << _scope << zero; }
829 |         }
830 |         Brcs {}
831 |         Out << ::std::endl;
832 |         Stmt(_static << _const << aap << _space << zero);
833 |         Out << ::std::endl;
834 |       }
835 |     }
836 |     Out << _semicolon;
837 |   }
838 |   //
```

#### 6.1.3.7. Support for the R extension

```
839 |   void tms2R() {
840 |     if (arg_.tms2R.empty()) {
841 |       ::std::cerr << "# Not copying to tsm2R" << std::endl;
842 |       return;
843 |     }
844 |     ::std::cerr << "# Copying to " << arg_.tms2R << std::endl;
845 |     for (::std::size_t i = 0; i < point::src_base(); i++) {
846 |       point p(arg_, i);
847 |       const fs::path from = p.src();
848 |       const fs::path to =
849 |           arg_.tms2R / "src" / (::std::to_string(i) + ".cc");
850 |       create_directories(to.parent_path());
851 |       fs::copy_file(from, to, fs::copy_option::overwrite_if_exists);
852 |     }
853 |     ::std::map<fs::path, fs::path> to_copy = {
854 |         {arg_.wd / "include" / "tms2" / "da.h",
855 |             arg_.tms2R / "src" / "tms2" / "da.h"},
856 |         {arg_.wd / "src" / "index.cc",
857 |             arg_.tms2R / "src" / "index.cc"},
858 |         {arg_.wd / "src" / "index.h",
859 |             arg_.tms2R / "src" / "index.h"}
860 |     };
861 |     for (const auto &resource : to_copy) {
862 |       create_directories(resource.second.parent_path());
863 |       fs::copy_file(resource.first, resource.second,
864 |                     fs::copy_option::overwrite_if_exists);
865 |     }
866 |     ::std::map<fs::path, const ::tms2::resource &> resources = {
867 |         {arg_.tms2R / "src" / "tms2" / "bitree.h",
868 |             LOAD_RESOURCE(bitree_h)},
869 |         {arg_.tms2R / "src" / "tms2" / "spectrum.h",
870 |             LOAD_RESOURCE(spectrum_h)},
871 |         {arg_.tms2R / "src" / "spectrum.cpp",
872 |             LOAD_RESOURCE(spectrum_cpp)}
873 |     };
874 |     for (const auto &resource : resources) {
875 |       create_directories(resource.first.parent_path());
876 |       ::std::fstream out(resource.first.string(), ::std::fstream::out);
877 |       out << resource.second;
878 |     }
879 |   }
880 |   //
```

#### 6.1.3.8. Code generation CMakeLists.txt project file

```
881 |   void preCMakeFile() {
882 |     ::std::size_t level = 0;
883 |     Setup(cmakelists_txt, level);
884 |     Out << "# Auto generated code" << ::std::endl;
885 |     Out << "cmake_minimum_required(VERSION 3.8)" << ::std::endl;
886 |     Out << std::endl;
887 |     Out << "include(CheckCXXCompilerFlag)" << ::std::endl;
888 |     //Out << "set(CMAKE_CXX_FLAGS \"${CMAKE_CXX_FLAGS}\")" << ::std::endl;
```

```
889     Out << "set(CMAKE_POSITION_INDEPENDENT_CODE ON)" << ::std::endl;
890     Out << "set(CMAKE_CXX_STANDARD 11)" << ::std::endl;
891     Out << "set(CMAKE_BUILD_TYPE Debug)" << ::std::endl;
892     Out
893         << "list(APPEND CMAKE_MODULE_PATH \"${CMAKE_CURRENT_LIST_DIR}/CMake\")"
894         << ::std::endl;
895     Out << "# uninstall target\n"
896         "if(NOT TARGET uninstall)\n"
897         "    configure_file(\n"
898         "            \"${CMAKE_CURRENT_SOURCE_DIR}/CMake/cmake_uninstall.cmake.in
                \"\n"
899         "            \"${CMAKE_CURRENT_BINARY_DIR}/CMake/cmake_uninstall.cmake\"\n"
900         "            IMMEDIATE @ONLY)\n"
901         "\n"
902         "    add_custom_target(uninstall\n"
903         "            COMMAND ${CMAKE_COMMAND} -P "
904         "${CMAKE_CURRENT_BINARY_DIR}/CMake/cmake_uninstall.cmake)\n"
905         "endif()" << ::std::endl;
906     Out << std::endl;
907     Out << "project(tms2index)" << ::std::endl;
908     Out << std::endl;
909     Out << "include_directories(include)" << ::std::endl;
910     Out << std::endl;
911     Out << "set(SOURCES src/index.cc)" << ::std::endl;
912     Out << std::endl;
913     fs::path src = "src";
914     for (::std::size_t i = 0; i < point::src_base(); i++) {
915       sources_.insert(src / (::std::to_string(i) + ".cc"));
916       point p(arg_, i);
917       ::std::fstream out(p.src().c_str(),
918                          ::std::fstream::out | ::std::fstream::app);
919       ::std::size_t level = 0;
920       out << _INCLUDE << fs::path("index.h") << ::std::endl;
921       out << ::std::endl;
922     }
923     ::std::map<fs::path, const ::tms2::resource &> resources = {
924         {arg_.wd / "CMake" / "cmake_uninstall.cmake.in",
925             LOAD_RESOURCE(cmake_uninstall_cmake_in)}
926     };
927     for (const auto &resource : resources) {
928       create_directories(resource.first.parent_path());
929       ::std::fstream out(resource.first.string(),
930                          ::std::fstream::out | ::std::fstream::app);
931       out << resource.second;
932     }
933   }
934   void postCMakeFile() {
935     ::std::size_t level = 0;
936     Setup(cmakelists_txt, level);
937     for (const fs::path &source : sources_) {
938       Out << "list(APPEND SOURCES " << source.string() << ")";
939       Out << ::std::endl;
940     }
941     Out << "add_library(tms2index STATIC ${SOURCES})" << ::std::endl;
942     Out << "install(TARGETS tms2index" << ::std::endl;
943     Out << "\tRUNTIME DESTINATION bin" << ::std::endl;
944     Out << "\tLIBRARY DESTINATION lib" << ::std::endl;
945     Out << "\tARCHIVE DESTINATION lib)" << ::std::endl;
946     Out << "install(DIRECTORY include/ DESTINATION include)" << ::std::endl;
947     Out << ::std::endl;
948   }
949   //
```

#### 6.1.3.9. Code generation INDEX.H private header

```
950   void print_index_h() {
951     ::std::size_t level = 0;
952     Setup(index_h, level);
953     const ::std::string guard = "__tms2_index_h__";
954     Tab;
955     Out << _IFNDEF << guard << ::std::endl;
956     Out << _DEFINE << guard << ::std::endl;
957     Out << ::std::endl;
```

```
958 |     Out << _INCLUDE;
959 |     SBrks { Out << da_h.string(); };
960 |     Out << ::std::endl;
961 |     Stmt(_using << _namespace << "tms2");
962 |     for (::std::size_t da = 0; da <= limit_; da++) {
963 |       Stmt(_extern << _const << "da"_type << _da(da));
964 |     }
965 |     Out << ::std::endl;
966 |     Out << _ENDIF << ::std::endl;
967 |   }
968 |   //
```

#### 6.1.3.10. Code generation for $\mathcal{I}$

```
969 |   void print_index_cc() {
970 |     const ::std::size_t limit = limit_ + 1;
971 |     const auto index = "index"_member;
972 |     const auto zero = "zero"_member_public;
973 |     const auto da = "da"_type;
974 |     ::std::size_t level = 0;
975 |     Setup(index_cc, level);
976 |     Out << _INCLUDE << fs::path("index.h") << ::std::endl;
977 |     Out << ::std::endl;
978 |     Tab << _const << da << '*' << da << _scope << index;
979 |     Brks { Out << limit; };
980 |     Out << _set;
981 |     Brcs Init {
982 |         for (::std::size_t da = 0; da < limit; da++) {
983 |           ArgNL << '&' << _da(da);
984 |         }
985 |         Out << ::std::endl;
986 |       }
987 |     Out << _semicolon << ::std::endl;
988 |     Out << _const << "ida"_type << "da"_type << _scope
989 |         << "min_minimuns"_member;
990 |     Brks { ArgN << arg_.B + 1; }
991 |     Out << _set;
992 |     Brcs Init {
993 |         for (::size_t i = 0; i < arg_.B; i++) {
994 |           if (i % 10 == 0) {
995 |             ArgNL << ::std::setw(9) << ::std::setfill(' ')
996 |                   << 0;
997 |           } else if (i) {
998 |             ArgN << ::std::setw(9) << ::std::setfill(' ')
999 |                  << 0;
1000|           }
1001|         }
1002|         Out << ::std::endl;
1003|       }
1004|     Out << _semicolon << ::std::endl;
1005|     Out << _const << "ida"_type << da << _scope
1006|         << "max_minimuns"_member;
1007|     Brks { ArgN << arg_.B + 1; }
1008|     Out << _set;
1009|     Brcs Init {
1010|         for (::size_t i = 0; i < arg_.B; i++) {
1011|           if (i % 10 == 0) {
1012|             ArgNL << ::std::setw(9) << ::std::setfill(' ')
1013|                   << arg_.M;
1014|           } else if (i) {
1015|             ArgN << ::std::setw(9) << ::std::setfill(' ')
1016|                  << arg_.M;
1017|           }
1018|         }
1019|         Out << ::std::endl;
1020|       }
1021|     Out << _semicolon << ::std::endl;
1022|     Stmt(_const << aap << aap << _scope << zero);
1023|     Stmt(_const << da << da << _scope << zero << _set << _da(0));
1024|     Out << ::std::endl;
1025|   }
1026 | };
```

```
::std::mutex point::src_guards[100];
static inline
::std::size_t bound(::std::size_t lower, long value, ::std::size_t upper) {
  return ::std::max((long) lower, ::std::min((long) upper, value));
}
//
```

#### 6.1.3.11. Code generation for Definition 4.1.1.9, *Combinations* of a *Distance* at Resolution $\mathcal{R}$

```
void point::print_code(const ::std::vector<::std::size_t> &mins) {
  ::std::lock_guard<::std::mutex>
  lock(src_guards[scaled_mw_ % src_base()]);
  ::std::fstream out(src_.c_str(), ::std::fstream::out | ::std::fstream::app);
  ::std::size_t level = 0;
  Setup(out, level);
  Out << _static << _constexpr << "aap"_type
      << _da(scaled_mw_, ::std::string("_comb"));
  Brks {
    Out << ::std::setw(9) << ::std::setfill(' ')
        << ::std::max(1uL, combinations());
  };
  Out << _set;
  Brcs Init {
      bool empty = true;
      for (const combination &c : combinations_) {
        ArgNL;
        out % c;
        empty = false;
      }
      if (empty) {
        ArgNL;
        combination empty;
        out % empty;
      }
      Out << ::std::endl;
    }
  Out << _semicolon << ::std::endl << ::std::endl;
  Out << _static << _constexpr << "ida"_type
      << _da(scaled_mw_, ::std::string("_mins"));
  Brks {
    Out << ::std::setw(9) << ::std::setfill(' ') << arguments_.B + 1;
  };
  Out << _set;
  Brcs Init {
      const long center = (long) scaled_mw_ - arguments_.B;
      ::std::size_t best = ::std::max(0L, center);
      for (long i = 0; i < (long) arguments_.B; i++) {
        const ::std::size_t upper = bound(0uL, center - i, scaled_mw_);
        const ::std::size_t lower = bound(0uL, center + i, scaled_mw_);
        if (mins[best] > mins[upper]) best = upper;
        if (mins[best] > mins[lower]) best = lower;
        const ::std::size_t inf = ::std::numeric_limits<::std::size_t>::max();
        if (i % 10 == 0) {
          ArgNL;
        } else if (i) {
          ArgN;
        }
        Out << ::std::setw(9) << ::std::setfill(' ') << best;
#ifdef TRACE
        Out << '/' << '*' << '-' << lower << '=' << (mins[lower] == inf ? 0 : mins[
            lower]) << ' ';
        Out << '+' << upper << '=' << (mins[upper] == inf ? 0 : mins[upper]) << '*'
             << '/';
#endif
      }
      Out << ::std::endl;
    }
  Out << _semicolon << ::std::endl;
  Tab << _constexpr << "da"_type << _da(scaled_mw_) << _set;
  Brcs Init {
```

```
1092 |       ArgNL << _formati << scaled_mw_;
1093 |       ArgNL << _formatf << mw_;
1094 |       ArgNL << _formati << combinations() << 'u' << 'L';
1095 |       ArgNL << _formati << permutations() << 'u' << 'L';
1096 |       ArgNL << _formati << combinations() << 'u' << 'L';
1097 |       ArgNL << _formati << permutations() << 'u' << 'L';
1098 |       ArgNL << _da(scaled_mw_, ::std::string("_comb"));
1099 |       ArgNL << _da(scaled_mw_, ::std::string("_mins"));
1100 |       Out << ::std::endl;
1101 |     };
1102 |   Out << _semicolon << ::std::endl << ::std::endl;
1103 |}
1104 |//
```

#### 6.1.3.12. Main program

```
1105 |bool getArguments(int argc, const char *const *argv, arguments &arg);
1106 |int main(int argc, const char **argv) {
1107 |   arguments arg;
1108 |   if (getArguments(argc, argv, arg)) {
1109 |     project project(arg);
1110 |     project.init();
1111 |     project.print_aah_h();
1112 |     project.print_index_h();
1113 |     project.caleyWalk();
1114 |     project.print_index_cc();
1115 |     project.postCMakeFile();
1116 |     project.tms2R();
1117 |     ::std::cout << "cd " << arg.wd / "bin" << ::std::endl;
1118 |     ::std::cout << "cmake ";
1119 |     //if (arg.prefix.size()) {
1120 |     //     ::std::cout << "-DCMAKE_INSTALL_PREFIX=\"\" ";
1121 |     //}
1122 |     ::std::cout << arg.wd << ::std::endl;
1123 |     if (!arg.prefix.empty()) {
1124 |       ::std::cout << "DESTDIR=" << arg.prefix << " ";
1125 |     }
1126 |     ::std::cout << "make --jobs " << ::std::thread::hardware_concurrency();
1127 |     ::std::cout << ::std::endl;
1128 |     ::std::cout << " " << std::endl;
1129 |     if (!arg.prefix.empty()) {
1130 |       ::std::cout << "DESTDIR=" << arg.prefix << " ";
1131 |     } else {
1132 |       ::std::cout << "echo 'Execute: sudo ";
1133 |     }
1134 |     ::std::cout << "make --jobs " << ::std::thread::hardware_concurrency();
1135 |     ::std::cout << " -C " << arg.wd / "bin" << " ";
1136 |     ::std::cout << " install'" << ::std::endl;
1137 |   }
1138 |   return EXIT_SUCCESS;
1139 |}
1140 |//
```

#### 6.1.3.13. Command line argument parsing

```
1141 |bool getArguments(int argc, const char *const *argv, arguments &arg) {
1142 |   const boost::filesystem::path tmpRoot(
1143 |       boost::filesystem::temp_directory_path());
1144 |   const boost::filesystem::path projectFileName("tms2");
1145 |   const boost::filesystem::path wd = tmpRoot / projectFileName;
1146 |   // Default values.
1147 |   arg.silent = 0;
1148 |   arg.verbose = 0;
1149 |   arg.wd = wd;
1150 |   ::std::string prefix;
1151 |   try {
1152 |     // Declare a group of options that will be allowed both on command lineand in config file
1153 |     boost::program_options::options_description config("Configuration");
1154 |     config.add_options()
1155 |         ("version,v", "print version string")
1156 |         ("help,h", "produce help message")
1157 |         ("resolution",
```

```
1158 |          boost::program_options::value<int>(&arg.R)->default_value(10),
1159 |          "resolution to use")
1160 |         ("maximum",
1161 |          boost::program_options::value<int>(&arg.M)->default_value(350),
1162 |          "maximum value to calculate in da")
1163 |         ("window",
1164 |          boost::program_options::value<int>(&arg.B)->default_value(3),
1165 |          "window size")
1166 |         ("prefix",
1167 |          boost::program_options::value<::std::string>()->default_value(
1168 |              ""),
1169 |          "installation dir")
1170 |         ("tms2R",
1171 |          boost::program_options::value<::std::string>()->default_value(
1172 |              ""),
1173 |          "tms2R source dir")
1174 |         ("include-path,I",
1175 |          boost::program_options::value<::std::vector<::std::string> >()->composing
     |              (),
1176 |          "include path");
1177 |     boost::program_options::variables_map vm;
1178 |     store(parse_command_line(argc, argv, config), vm);
1179 |     notify(vm);
1180 |     if (vm.count("help")) {
1181 |       ::std::cout << config << ::std::endl;
1182 |       return false;
1183 |     }
1184 |     if (vm.count("version")) {
1185 |       ::std::cout << config << ::std::endl;
1186 |       return false;
1187 |     }
1188 |     arg.R = vm["resolution"].as<int>();
1189 |     arg.B = vm["window"].as<int>();
1190 |     arg.M = vm["maximum"].as<int>();
1191 |     arg.prefix = vm["prefix"].as<::std::string>();
1192 |     arg.tms2R = vm["tms2R"].as<::std::string>();
1193 |   } catch (const boost::program_options::error &ex) {
1194 |     std::cerr << ex.what() << '\n';
1195 |     return false;
1196 |   }
1197 |   if (!arg.silent) {
1198 |     ::std::cerr << "# Resolution: " << arg.R << ::std::endl;
1199 |     ::std::cerr << "# Maximum: " << arg.M << ::std::endl;
1200 |     ::std::cerr << "# Window:  " << arg.B << ::std::endl;
1201 |     ::std::cerr << "# Output: " << arg.wd << ::std::endl;
1202 |     ::std::cerr << "# Install: " << arg.prefix << ::std::endl;
1203 |   }
1204 |   arg.M *= arg.R;
1205 |   arg.B *= arg.R;
1206 |   return true;
1207 | }
1208 | //
```

#### 6.1.3.14. Implementation of Definition 4.1.1.2, *Amino Acid Residue*

```
1209 | constexpr mw_t H = 001.0079;
1210 | constexpr mw_t C = 012.0107;
1211 | constexpr mw_t N = 014.0067;
1212 | constexpr mw_t O = 015.9994;
1213 | constexpr mw_t H2O = H * 2 + O;
1214 | const tms2::indexgen::aa_list_t tms2::indexgen::aa_list = {
1215 |     {"A",   089.094000 - H2O},
1216 |     {"C",   121.154000 - H2O},
1217 |     {"E",   147.131000 - H2O},
1218 |     {"D",   133.104000 - H2O},
1219 |     {"G",   075.067000 - H2O},
1220 |     {"F",   165.192000 - H2O},
1221 |     {"I",   131.175000 - H2O},
1222 |     {"H",   155.156000 - H2O},
1223 |     {"K",   146.189000 - H2O},
1224 |     {"M",   149.208000 - H2O},
```

```
1225 |     {"O",   273.325000 - H20},
1226 |     {"N",   132.119000 - H20},
1227 |     {"Q",   146.146000 - H20},
1228 |     {"P",   115.132000 - H20},
1229 |     {"S",   105.093000 - H20},
1230 |     {"R",   174.203000 - H20},
1231 |     {"U",   159.065000 - H20},
1232 |     {"T",   119.120000 - H20},
1233 |     {"W",   204.228000 - H20},
1234 |     {"V",   117.148000 - H20},
1235 |     {"Y",   181.191000 - H20},
1236 |     // These are modified amino acids added later for extend range ofinterpretations
1237 |//     {"Ccm", 161.175500 - H20},
1238 |//     {"Cpe", 208.284000 - H20},
1239 |//     {"Cpm", 174.217600 - H20},
1240 |//     {"Mso", 147.192000 - H20},
1241 |//     {"Tpo", 202.212600 - H20},
1242 |//     {"Hsl", 100.097140 - H20}
1243 |};
1244 |const tms2::indexgen::memoizer_factorial tms2::indexgen::fact;
```

**Listing 6.4:** `resource.h`: Resource header

```
1 //
```

## 2 6.1.4. Embedded resources support header

```
3  #ifndef TMS2_RESOURCE_H
4  #define TMS2_RESOURCE_H
5  #include <string>
6  #include <ostream>
7  namespace tms2 {
8  // TODO: Add reference to the original source code.
9  class resource {
10 public:
11   inline constexpr resource(const char *start, const ::std::size_t len) :
12       resource_data(start),
13       data_len(len) {}
14   inline const char *const &data() const { return resource_data; }
15   inline const ::std::size_t &size() const { return data_len; }
16   inline const char *begin() const { return resource_data; }
17   inline const char *end() const { return resource_data + data_len; }
18   inline ::std::string toString() { return ::std::string(data(), size()); }
19   template<typename _CharT, typename _Traits>
20   friend inline
21   ::std::basic_ostream<_CharT, _Traits> &operator<<(
22       ::std::basic_ostream<_CharT, _Traits> &out,
23       const resource &r) {
24     for (const char &c: r) {
25       out << c;
26     }
27     return out;
28   }
29 private:
30   const char *resource_data;
31   const ::std::size_t data_len;
32 };
33 }
34 #define LOAD_RESOURCE(RESOURCE) ([]() { \
35     extern const char __resource_##RESOURCE##_start_[]; \
36     extern const ::std::size_t __resource_##RESOURCE##_len_; \
37     return ::tms2::resource(__resource_##RESOURCE##_start_, __resource_##RESOURCE##
         _len_); \
38 })()
39 #endif
```

**Listing 6.5:** `printer.h`: C++ pretty printer header

```
1 //
```

## 2 6.1.5. C++ pretty printing support

```
3  #ifndef TMS2_PRINTER_H
4  #define TMS2_PRINTER_H
5  #include <cstddef>
6  #include <iostream>
7  #include <typeinfo>
8  #include <new>
9  #ifndef _TMS2_INLINE_VISIBILITY
10 #define _TMS2_INLINE_VISIBILITY \
11 __attribute__((__visibility__("hidden"), __always_inline__))
12 #endif
13 namespace tms2 {
14 inline namespace cpp {
15 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
16 class ident {
17 public:
18   inline _TMS2_INLINE_VISIBILITY
19   ident(
20       ::std::basic_ostream<_CharT, _Traits> &out,
21       ::std::size_t &level) :
22       out_(out),
23       level_(level) {
24     level_++;
25   }
26   virtual ~ident() { level_--; }
27   friend inline _TMS2_INLINE_VISIBILITY
28   ::std::basic_ostream<_CharT, _Traits> &operator<<(
29       ::std::basic_ostream<_CharT, _Traits> &__out,
30       const ident &__c) {
31     __out << '\n';
32     __out.fill(' ');
33     __out.width(2 * __c.level_ - 1);
34     __out << ' ';
35     __out.flush();
36     return __out;
37   }
38 //     ::std::basic_ostream<_CharT, _Traits> &operator<<(
39 //             ::std::basic_ostream<_CharT, _Traits> &(*m)(::std::basic_ostream<
     _CharT, _Traits> &)) {
40 //         (*m)(out_);
41 //         return out_;
42 //     }
43 protected:
44   ::std::basic_ostream<_CharT, _Traits> &out_;
45   ::std::size_t &level_;
46 private:
47   ident() = delete;
48   ident(const ident &) = delete;
49   ident &operator=(const ident &) = delete;
50 };
51 template<typename _CharT = char, typename _Traits=::std::char_traits<_CharT> >
52 class initialize_ : public virtual ident<_CharT, _Traits> {
53 public:
54   inline _TMS2_INLINE_VISIBILITY
55   initialize_(
56       ::std::basic_ostream<_CharT, _Traits> &out,
57       ::std::size_t &level) :
58       ident<_CharT, _Traits>::ident(out, level),
59       stmts_(0) {
60     this->level_++;
61   }
62   inline _TMS2_INLINE_VISIBILITY
63   ~initialize_() { this->level_--; }
```

```
64 |   inline _TMS2_INLINE_VISIBILITY
65 |   ::std::basic_ostream<_CharT, _Traits> &operator()(bool nl) {
66 |     if (this->stmts_++) {
67 |       if (nl) { this->out_ << ',' << *this; }
68 |       else { this->out_ << ',' << ' '; }
69 |     } else if (nl) { this->out_ << *this; }
70 |     return this->out_;
71 |   }
72 | protected:
73 |   mutable ::std::size_t stmts_;
74 | private:
75 |   initialize_() = delete;
76 |   initialize_(const initialize_ &) = delete;
77 |   initialize_ &operator=(const initialize_ &) = delete;
78 | };
79 | template<typename _CharT = char, typename _Traits=::std::char_traits<_CharT> >
80 | class block_ : public virtual ident<_CharT, _Traits> {
81 | public:
82 |   typedef typename ::std::basic_ostream<_CharT, _Traits> &(*action_t)(
83 |       ::std::basic_ostream<_CharT, _Traits> &);
84 |   inline _TMS2_INLINE_VISIBILITY
85 |   block_(
86 |       ::std::basic_ostream<_CharT, _Traits> &out,
87 |       ::std::size_t &level,
88 |       action_t open,
89 |       action_t close) :
90 |       ident<_CharT>::ident(out, level),
91 |       stmts_(0),
92 |       open_(open),
93 |       close_(close) {
94 |     this->out_ << open_;
95 |   }
96 |   inline _TMS2_INLINE_VISIBILITY
97 |   ~block_() { this->out_ << close_; }
98 |   //virtual ::std::basic_ostream<_CharT, _Traits> &operator()() {
99 |   //  this->out_ << *this;
100 |  //  return this->out_;
101 |  //}
102 | protected:
103 |   mutable ::size_t stmts_;
104 | private:
105 |   action_t close_;
106 |   action_t open_;
107 |   block_() = delete;
108 |   block_(const block_ &) = delete;
109 |   block_ &operator=(const block_ &) = delete;
110 | };
111 | template<typename _CharT, typename _Traits=::std::char_traits<char>>
112 | class parenthesis_ : public virtual block_<_CharT, _Traits> {
113 | public:
114 |   inline _TMS2_INLINE_VISIBILITY
115 |   parenthesis_(
116 |       ::std::basic_ostream<_CharT, _Traits> &out,
117 |       ::std::size_t &level) :
118 |       ident<_CharT>::ident(out, level),
119 |       block_<_CharT, _Traits>::block_(out, level, open_, close_) {}
120 |   inline _TMS2_INLINE_VISIBILITY
121 |   ~parenthesis_() {}
122 |   inline _TMS2_INLINE_VISIBILITY
123 |   ::std::basic_ostream<_CharT, _Traits> &operator()(bool nl) {
124 |     if (this->stmts_++) {
125 |       if (nl) { this->out_ << ',' << *this; }
126 |       else { this->out_ << ',' << ' '; }
127 |     } else if (nl) { this->out_ << *this; }
128 |     return this->out_;
129 |   }
130 |   friend inline _TMS2_INLINE_VISIBILITY
131 |   ::std::basic_ostream<_CharT, _Traits> &operator<<(
132 |       ::std::basic_ostream<_CharT, _Traits> &out,
133 |       const parenthesis_ &c) {
```

```
134 |     out << '\n';
135 |     for (int i = 1; i < c.level_; i++) out << ' ' << ' ';
136 |     out.flush();
137 |     return out;
138 |   }
139 | private:
140 |   static ::std::basic_ostream<_CharT, _Traits> &open_(
141 |       ::std::basic_ostream<_CharT, _Traits> &out) {
142 |     return out << '(';
143 |   }
144 |   static ::std::basic_ostream<_CharT, _Traits> &close_(
145 |       ::std::basic_ostream<_CharT, _Traits> &out) {
146 |     return out << ')';
147 |   }
148 |   parenthesis_() = delete;
149 |   parenthesis_(const parenthesis_ &) = delete;
150 |   parenthesis_ &operator=(const parenthesis_ &) = delete;
151 | };
152 | template<typename _CharT, typename _Traits=::std::char_traits<char>>
153 | class braces_ : public virtual block_<_CharT, _Traits> {
154 | public:
155 |   inline _TMS2_INLINE_VISIBILITY
156 |   braces_(
157 |       ::std::basic_ostream<_CharT, _Traits> &out,
158 |       ::std::size_t &level) :
159 |       ident<_CharT>::ident(out, level),
160 |       block_<_CharT, _Traits>::block_(out, level, open_, close_) {}
161 |   inline _TMS2_INLINE_VISIBILITY
162 |   ~braces_() {
163 |     if (this->stmts_) {
164 |       this->out_ << '\n';
165 |       for (int i = 2; i < this->level_; i++)
166 |         this->out_ << ' ' << ' ';
167 |     }
168 |   }
169 |   friend inline _TMS2_INLINE_VISIBILITY
170 |   ::std::basic_ostream<_CharT, _Traits> &operator<<(
171 |       ::std::basic_ostream<_CharT, _Traits> &out,
172 |       const braces_ &c) {
173 |     c.stmts_++;
174 |     out << '\n';
175 |     for (int i = 1; i < c.level_; i++) out << ' ' << ' ';
176 |     out.flush();
177 |     return out;
178 |   }
179 | private:
180 |   static ::std::basic_ostream<_CharT, _Traits> &open_(
181 |       ::std::basic_ostream<_CharT, _Traits> &out) {
182 |     return out << ' ' << '{';
183 |   }
184 |   static ::std::basic_ostream<_CharT, _Traits> &close_(
185 |       ::std::basic_ostream<_CharT, _Traits> &out) {
186 |     return out << '}';
187 |   }
188 |   braces_() = delete;
189 |   braces_(const braces_ &) = delete;
190 |   braces_ &operator=(const braces_ &) = delete;
191 | };
192 | template<typename _CharT, typename _Traits=::std::char_traits<char>>
193 | class brackets_ : public virtual block_<_CharT, _Traits> {
194 | public:
195 |   inline _TMS2_INLINE_VISIBILITY
196 |   brackets_(
197 |       ::std::basic_ostream<_CharT, _Traits> &out,
198 |       ::std::size_t &level) :
199 |       ident<_CharT>::ident(out, level),
200 |       block_<_CharT, _Traits>::block_(out, level, open_, close_) {}
201 |   inline _TMS2_INLINE_VISIBILITY
202 |   ~brackets_() {}
203 |   inline _TMS2_INLINE_VISIBILITY
```

```
204 |   ::std::basic_ostream<_CharT, _Traits> &operator()(bool nl) {
205 |     if (this->stmts_++) {
206 |       if (nl) { this->out_ << ',' << '\n' << *this; }
207 |       else { this->out_ << ',' << ' '; }
208 |     }
209 |     return this->out_;
210 |   }
211 |   friend inline _TMS2_INLINE_VISIBILITY
212 |   ::std::basic_ostream<_CharT, _Traits> &operator<<(
213 |       ::std::basic_ostream<_CharT, _Traits> &out,
214 |       const brackets_ &c) {
215 |     for (int i = 0; i < c.level_; i++) out << ' ' << ' ';
216 |     out.flush();
217 |     return out;
218 |   }
219 | private:
220 |   static ::std::basic_ostream<_CharT, _Traits> &open_(
221 |       ::std::basic_ostream<_CharT, _Traits> &out) {
222 |     return out << '[';
223 |   }
224 |   static ::std::basic_ostream<_CharT, _Traits> &close_(
225 |       ::std::basic_ostream<_CharT, _Traits> &out) {
226 |     return out << ']';
227 |   }
228 |   brackets_() = delete;
229 |   brackets_(const brackets_ &) = delete;
230 |   brackets_ &operator=(const brackets_ &) = delete;
231 | };
232 | template<typename _CharT, typename _Traits=::std::char_traits<char>>
233 | class sbrackets_ : public virtual block_<_CharT, _Traits> {
234 | public:
235 |   inline _TMS2_INLINE_VISIBILITY
236 |   sbrackets_(
237 |       ::std::basic_ostream<_CharT, _Traits> &out,
238 |       ::std::size_t &level) :
239 |       ident<_CharT>::ident(out, level),
240 |       block_<_CharT, _Traits>::block_(out, level, open_, close_) {}
241 |   inline _TMS2_INLINE_VISIBILITY
242 |   ~sbrackets_() {}
243 |   inline _TMS2_INLINE_VISIBILITY
244 |   ::std::basic_ostream<_CharT, _Traits> &operator()(bool nl) {
245 |     if (this->stmts_++) {
246 |       if (nl) { this->out_ << ',' << ' '; }
247 |       else { this->out_ << ',' << '\n' << *this; }
248 |     }
249 |     return this->out_;
250 |   }
251 |   friend inline _TMS2_INLINE_VISIBILITY
252 |   ::std::basic_ostream<_CharT, _Traits> &operator<<(
253 |       ::std::basic_ostream<_CharT, _Traits> &out,
254 |       const sbrackets_ &c) {
255 |     for (int i = 0; i < c.level_; i++) out << ' ' << ' ';
256 |     out.flush();
257 |     return out;
258 |   }
259 | private:
260 |   static ::std::basic_ostream<_CharT, _Traits> &open_(
261 |       ::std::basic_ostream<_CharT, _Traits> &out) {
262 |     return out << '<';
263 |   }
264 |   static ::std::basic_ostream<_CharT, _Traits> &close_(
265 |       ::std::basic_ostream<_CharT, _Traits> &out) {
266 |     return out << '>';
267 |   }
268 |   sbrackets_() = delete;
269 |   sbrackets_(const sbrackets_ &) = delete;
270 |   sbrackets_ &operator=(const sbrackets_ &) = delete;
271 | };
272 | class __tms2_da {
273 | public:
```

```
274 |   _TMS2_INLINE_VISIBILITY
275 |   __tms2_da(::std::size_t __n) : __n_(__n) {}
276 |   template<class _CharT, class _Traits>
277 |   friend inline _TMS2_INLINE_VISIBILITY
278 |   ::std::basic_istream<_CharT, _Traits> &operator>>(
279 |       ::std::basic_istream<_CharT, _Traits> &__is,
280 |       const __tms2_da &__x) {
281 |     __is.width(__x.__n_);
282 |     return __is;
283 |   }
284 |   template<class _CharT, class _Traits>
285 |   friend inline _TMS2_INLINE_VISIBILITY
286 |   ::std::basic_ostream<_CharT, _Traits> &operator<<(
287 |       ::std::basic_ostream<_CharT, _Traits> &__os,
288 |       const __tms2_da &__x) {
289 |     __os << '_' << '_' << 'd' << 'a';
290 |     __os.width(9);
291 |     __os.fill('0');
292 |     __os << __x.__n_;
293 |     return __os;
294 |   }
295 | private:
296 |   ::std::size_t __n_;
297 | };
298 | inline _TMS2_INLINE_VISIBILITY
299 | __tms2_da _da(::std::size_t n) { return __tms2_da(n); }
300 | template<class _CharT, class _Traits>
301 | class __tms2_da2 {
302 | public:
303 |   inline _TMS2_INLINE_VISIBILITY
304 |   explicit __tms2_da2(
305 |       ::std::size_t __n,
306 |       ::std::basic_string<_CharT, _Traits> __suffix) :
307 |       __n_(__n),
308 |       __suffix_(__suffix) {}
309 |   friend inline _TMS2_INLINE_VISIBILITY
310 |   ::std::basic_istream<_CharT, _Traits> &operator>>(
311 |       ::std::basic_istream<_CharT, _Traits> &__is,
312 |       const __tms2_da2 &__x) {
313 |     __is.width(__x.__n_);
314 |     return __is;
315 |   }
316 |   friend inline _TMS2_INLINE_VISIBILITY
317 |   ::std::basic_ostream<_CharT, _Traits> &operator<<(
318 |       ::std::basic_ostream<_CharT, _Traits> &__os,
319 |       const __tms2_da2 &__x) {
320 |     __os << '_' << '_' << 'd' << 'a';
321 |     __os.width(9);
322 |     __os.fill('0');
323 |     __os << __x.__n_;
324 |     __os << __x.__suffix_;
325 |     return __os;
326 |   }
327 | private:
328 |   ::std::size_t __n_;
329 |   ::std::basic_string<_CharT, _Traits> __suffix_;
330 | };
331 | template<class _CharT, class _Traits>
332 | inline _TMS2_INLINE_VISIBILITY
333 | __tms2_da2<_CharT, _Traits>
334 | _da(::std::size_t n, ::std::basic_string<_CharT, _Traits> suffix) {
335 |   return __tms2_da2<_CharT, _Traits>(n, suffix);
336 | }
337 | template<class _CharT, class _Traits>
338 | class __tms2_var {
339 | public:
340 |   _TMS2_INLINE_VISIBILITY inline
341 |   __tms2_var(
342 |       ::std::basic_string<_CharT, _Traits> __prefix,
343 |       ::std::basic_string<_CharT, _Traits> __name,
344 |       ::std::basic_string<_CharT, _Traits> __suffix) :
```

```
345 |       __prefix_(__prefix),
346 |       __name_(__name),
347 |       __suffix_(__suffix) {}
348 |   _TMS2_INLINE_VISIBILITY
349 |   __tms2_var(const __tms2_var &) = default;
350 |   _TMS2_INLINE_VISIBILITY
351 |   __tms2_var &operator=(const __tms2_var &other) {
352 |     return *new(*this) __tms2_var(other);
353 |   }
354 |   friend inline _TMS2_INLINE_VISIBILITY
355 |   ::std::basic_ostream<_CharT, _Traits> &
356 |   operator<<(::std::basic_ostream<_CharT, _Traits> &__os,
357 |              const __tms2_var &__x) {
358 |     __os << __x.__prefix_ << __x.__name_ << __x.__suffix_;
359 |     return __os;
360 |   }
361 |   inline _TMS2_INLINE_VISIBILITY
362 |   __tms2_var to_local(void) const noexcept {
363 |     return __tms2_var("", __name_, "_");
364 |   }
365 |   ::std::basic_string<_CharT, _Traits> __prefix_;
366 |   ::std::basic_string<_CharT, _Traits> __name_;
367 |   ::std::basic_string<_CharT, _Traits> __suffix_;
368 | private:
369 |   __tms2_var() = delete;
370 | };
371 | template<class... Args>
372 | struct __tms2_par {
373 | public:
374 |   inline _TMS2_INLINE_VISIBILITY
375 |   __tms2_par() {}
376 |   inline _TMS2_INLINE_VISIBILITY
377 |   __tms2_par(const __tms2_par &) = default;
378 |   template<class _CharT, class _Traits>
379 |   inline _TMS2_INLINE_VISIBILITY
380 |   void print(::std::basic_ostream<_CharT, _Traits> &__out,
381 |              ::std::size_t count) const {}
382 |   template<class _CharT, class _Traits>
383 |   friend inline _TMS2_INLINE_VISIBILITY
384 |   ::std::basic_ostream<_CharT, _Traits> &
385 |   operator<<(
386 |       ::std::basic_ostream<_CharT, _Traits> &__os,
387 |       const __tms2_par &__x) {
388 |     __os << '(';
389 |     __x.print(__os, 0);
390 |     __os << ')';
391 |     return __os;
392 |   }
393 | };
394 | template<class T, class... Args>
395 | struct __tms2_par<T, Args...> : __tms2_par<Args...> {
396 |   const T &__arg_;
397 | public:
398 |   inline _TMS2_INLINE_VISIBILITY
399 |   __tms2_par(const T &arg, const Args &...args) :
400 |       __tms2_par<Args...>::__tms2_par(args...),
401 |       __arg_(arg) {}
402 |   template<class _CharT, class _Traits>
403 |   friend inline _TMS2_INLINE_VISIBILITY
404 |   ::std::basic_ostream<_CharT, _Traits> &
405 |   operator<<(::std::basic_ostream<_CharT, _Traits> &__os,
406 |              const __tms2_par &__x) {
407 |     __os << '(';
408 |     __x.print(__os, 0);
409 |     __os << ')';
410 |     return __os;
411 |   }
412 |   inline _TMS2_INLINE_VISIBILITY
413 |   __tms2_par(const __tms2_par &) = default;
414 |   template<class _CharT, class _Traits>
415 |   inline _TMS2_INLINE_VISIBILITY
```

```
416 |   void print(
417 |       ::std::basic_ostream<_CharT, _Traits> &__out,
418 |       ::std::size_t count) const {
419 |     if (count) __out << ',' << ' ';
420 |     __out << __arg_;
421 |     __tms2_par<Args...>::print(__out, count + 1);
422 |   }
423 | };
424 | template<class... Args>
425 | inline _TMS2_INLINE_VISIBILITY
426 | __tms2_par<Args...> par(const Args &...args) {
427 |   return __tms2_par<Args...>(args...);
428 | }
429 | template<class _CharT, class _Traits>
430 | inline _TMS2_INLINE_VISIBILITY
431 | __tms2_var<_CharT, _Traits>
432 | _private_member(::std::basic_string<_CharT, _Traits> name) {
433 |   return __tms2_var<_CharT, _Traits>(
434 |       ::std::string(""),
435 |       name,
436 |       ::std::string("_"));
437 | }
438 | template<class _CharT, class _Traits>
439 | inline _TMS2_INLINE_VISIBILITY
440 | __tms2_var<_CharT, _Traits> _member(
441 |     ::std::basic_string<_CharT, _Traits> name) {
442 |   return __tms2_var<_CharT, _Traits>(
443 |       ::std::string("__"),
444 |       ::std::string(name),
445 |       ::std::string("_"));
446 | }
447 | template<class _CharT, class _Traits>
448 | inline _TMS2_INLINE_VISIBILITY
449 | __tms2_var<_CharT, _Traits> _public_member(
450 |     ::std::basic_string<_CharT, _Traits> name) {
451 |   return __tms2_var<_CharT, _Traits>(
452 |       ::std::string(""),
453 |       ::std::string(name),
454 |       ::std::string(""));
455 | }
456 | template<class _CharT, class _Traits>
457 | inline _TMS2_INLINE_VISIBILITY
458 | __tms2_var<_CharT, _Traits>
459 | _protected_member(::std::basic_string<_CharT, _Traits> name) {
460 |   return __tms2_var<_CharT, _Traits>(
461 |       ::std::string("__"),
462 |       ::std::string(name),
463 |       ::std::string("_"));
464 | }
465 | template<class _CharT, class _Traits>
466 | inline _TMS2_INLINE_VISIBILITY
467 | __tms2_var<_CharT, _Traits>
468 | _local(::std::basic_string<_CharT, _Traits> name) {
469 |   return __tms2_var<_CharT, _Traits>(
470 |       ::std::string("__"),
471 |       ::std::string(name),
472 |       ::std::string(""));
473 | }
474 | template<class _CharT, class _Traits>
475 | inline _TMS2_INLINE_VISIBILITY
476 | __tms2_var<_CharT, _Traits>
477 | _type(::std::basic_string<_CharT, _Traits> name) {
478 |   return __tms2_var<_CharT, _Traits>(
479 |       ::std::string(""),
480 |       ::std::string(name),
481 |       ::std::string(" "));
482 | }
483 | __tms2_var<char, ::std::char_traits<char>> operator "" _member(
484 |     const char *name, ::std::size_t) {
485 |   return __tms2_var<char, ::std::char_traits<char>>(
```

```
486 |       ::std::string("__"),
487 |       ::std::string(name),
488 |       ::std::string("_"));
489 |}
490 |__tms2_var<char, ::std::char_traits<char>> operator "" _member_public(
491 |    const char *name, ::std::size_t) {
492 |  return __tms2_var<char, ::std::char_traits<char>>(
493 |       ::std::string(""),
494 |       ::std::string(name),
495 |       ::std::string(""));
496 |}
497 |__tms2_var<char, ::std::char_traits<char>> operator "" _local(
498 |    const char *name, ::std::size_t) {
499 |  return __tms2_var<char, ::std::char_traits<char>>(
500 |       ::std::string("__"),
501 |       ::std::string(name),
502 |       ::std::string(""));
503 |}
504 |__tms2_var<char, ::std::char_traits<char>> operator "" _type(
505 |    const char *name, ::std::size_t) {
506 |  return __tms2_var<char, ::std::char_traits<char>>(
507 |       ::std::string(""),
508 |       ::std::string(name),
509 |       ::std::string(" "));
510 |}
511 |__tms2_var<char, ::std::char_traits<char>> operator "" _type_struct(
512 |    const char *name, ::std::size_t) {
513 |  return __tms2_var<char, ::std::char_traits<char>>(
514 |       ::std::string("struct "),
515 |       ::std::string(name),
516 |       ::std::string(" "));
517 |}
518 |template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
519 |inline _TMS2_INLINE_VISIBILITY
520 |::std::basic_ostream<_CharT, _Traits> &
521 |_alignas(::std::basic_ostream<_CharT, _Traits> &out) {
522 |  out << 'a' << 'l' << 'i' << 'g' << 'n' << 'a' << 's' << ' ';
523 |  return out;
524 |}
525 |template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
526 |inline _TMS2_INLINE_VISIBILITY
527 |::std::basic_ostream<_CharT, _Traits> &
528 |_alignof(::std::basic_ostream<_CharT, _Traits> &out) {
529 |  out << 'a' << 'l' << 'i' << 'g' << 'n' << 'o' << 'f' << ' ';
530 |  return out;
531 |}
532 |template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
533 |inline _TMS2_INLINE_VISIBILITY
534 |::std::basic_ostream<_CharT, _Traits> &
535 |_and(::std::basic_ostream<_CharT, _Traits> &out) {
536 |  out << ' ' << '&' << '&' << ' ';
537 |  return out;
538 |}
539 |template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
540 |inline _TMS2_INLINE_VISIBILITY
541 |::std::basic_ostream<_CharT, _Traits> &
542 |_and_eq(::std::basic_ostream<_CharT, _Traits> &out) {
543 |  out << ' ' << 'a' << 'n' << 'd' << '_' << 'e' << 'q' << ' ';
544 |  return out;
545 |}
546 |template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
547 |inline _TMS2_INLINE_VISIBILITY
548 |::std::basic_ostream<_CharT, _Traits> &
549 |_asm(::std::basic_ostream<_CharT, _Traits> &out) {
550 |  out << 'a' << 's' << 'm' << ' ';
551 |  return out;
552 |}
553 |template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
554 |inline _TMS2_INLINE_VISIBILITY
555 |::std::basic_ostream<_CharT, _Traits> &
```

```
556 _atomic_cancel(::std::basic_ostream<_CharT, _Traits> &out) {
557   out << 'a' << 't' << 'o' << 'm' << 'i' << 'c' << '_' << 'c' << 'a'
558       << 'n' << 'c' << 'e' << 'l' << ' ';
559   return out;
560 }
561 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
562 inline _TMS2_INLINE_VISIBILITY
563 ::std::basic_ostream<_CharT, _Traits> &
564 _atomic_commit(::std::basic_ostream<_CharT, _Traits> &out) {
565   out << 'a' << 't' << 'o' << 'm' << 'i' << 'c' << '_' << 'c' << 'o'
566       << 'm' << 'm' << 'i' << 't' << ' ';
567   return out;
568 }
569 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
570 inline _TMS2_INLINE_VISIBILITY
571 ::std::basic_ostream<_CharT, _Traits> &
572 _atomic_noexcept(::std::basic_ostream<_CharT, _Traits> &out) {
573   out << 'a' << 't' << 'o' << 'm' << 'i' << 'c' << '_' << 'n' << 'o'
574       << 'e' << 'x' << 'c' << 'e' << 'p' << 't' << ' ';
575   return out;
576 }
577 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
578 inline _TMS2_INLINE_VISIBILITY
579 ::std::basic_ostream<_CharT, _Traits> &
580 _auto(::std::basic_ostream<_CharT, _Traits> &out) {
581   out << 'a' << 'u' << 't' << 'o' << ' ';
582   return out;
583 }
584 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
585 inline _TMS2_INLINE_VISIBILITY
586 ::std::basic_ostream<_CharT, _Traits> &
587 _bitand(::std::basic_ostream<_CharT, _Traits> &out) {
588   out << '&';
589   return out;
590 }
591 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
592 inline _TMS2_INLINE_VISIBILITY
593 ::std::basic_ostream<_CharT, _Traits> &
594 _bitor(::std::basic_ostream<_CharT, _Traits> &out) {
595   out << '|';
596   return out;
597 }
598 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
599 inline _TMS2_INLINE_VISIBILITY
600 ::std::basic_ostream<_CharT, _Traits> &
601 _bool(::std::basic_ostream<_CharT, _Traits> &out) {
602   out << 'b' << 'o' << 'o' << 'l' << ' ';
603   return out;
604 }
605 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
606 inline _TMS2_INLINE_VISIBILITY
607 ::std::basic_ostream<_CharT, _Traits> &
608 _break(::std::basic_ostream<_CharT, _Traits> &out) {
609   out << 'b' << 'r' << 'e' << 'a' << 'k' << ' ';
610   return out;
611 }
612 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
613 inline _TMS2_INLINE_VISIBILITY
614 ::std::basic_ostream<_CharT, _Traits> &
615 _case(::std::basic_ostream<_CharT, _Traits> &out) {
616   out << 'c' << 'a' << 's' << 'e' << ' ';
617   return out;
618 }
619 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
620 inline _TMS2_INLINE_VISIBILITY
621 ::std::basic_ostream<_CharT, _Traits> &
622 _catch(::std::basic_ostream<_CharT, _Traits> &out) {
623   out << 'c' << 'a' << 't' << 'c' << 'h' << ' ';
624   return out;
625 }
626 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
627 inline _TMS2_INLINE_VISIBILITY
628 ::std::basic_ostream<_CharT, _Traits> &
```

```
629 _char(::std::basic_ostream<_CharT, _Traits> &out) {
630   out << 'c' << 'h' << 'a' << 'r' << ' ';
631   return out;
632 }
633 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
634 inline _TMS2_INLINE_VISIBILITY
635 ::std::basic_ostream<_CharT, _Traits> &
636 _char16_t(::std::basic_ostream<_CharT, _Traits> &out) {
637   out << 'c' << 'h' << 'a' << 'r' << '1' << '6' << '_' << 't' << ' ';
638   return out;
639 }
640 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
641 inline _TMS2_INLINE_VISIBILITY
642 ::std::basic_ostream<_CharT, _Traits> &
643 _char32_t(::std::basic_ostream<_CharT, _Traits> &out) {
644   out << 'c' << 'h' << 'a' << 'r' << '3' << '2' << '_' << 't' << ' ';
645   return out;
646 }
647 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
648 inline _TMS2_INLINE_VISIBILITY
649 ::std::basic_ostream<_CharT, _Traits> &
650 _class(::std::basic_ostream<_CharT, _Traits> &out) {
651   out << 'c' << 'l' << 'a' << 's' << 's' << ' ';
652   return out;
653 }
654 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
655 inline _TMS2_INLINE_VISIBILITY
656 ::std::basic_ostream<_CharT, _Traits> &
657 _compl(::std::basic_ostream<_CharT, _Traits> &out) {
658   out << 'c' << 'o' << 'm' << 'p' << 'l' << ' ';
659   return out;
660 }
661 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
662 inline _TMS2_INLINE_VISIBILITY
663 ::std::basic_ostream<_CharT, _Traits> &
664 _concept(::std::basic_ostream<_CharT, _Traits> &out) {
665   out << 'c' << 'o' << 'n' << 'c' << 'e' << 'p' << 't' << ' ';
666   return out;
667 }
668 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
669 inline _TMS2_INLINE_VISIBILITY
670 ::std::basic_ostream<_CharT, _Traits> &
671 _const(::std::basic_ostream<_CharT, _Traits> &out) {
672   out << 'c' << 'o' << 'n' << 's' << 't' << ' ';
673   return out;
674 }
675 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
676 inline _TMS2_INLINE_VISIBILITY
677 ::std::basic_ostream<_CharT, _Traits> &
678 _constexpr(::std::basic_ostream<_CharT, _Traits> &out) {
679   out << 'c' << 'o' << 'n' << 's' << 't' << 'e' << 'x' << 'p' << 'r' << ' ';
680   return out;
681 }
682 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
683 inline _TMS2_INLINE_VISIBILITY
684 ::std::basic_ostream<_CharT, _Traits> &
685 _const_cast(::std::basic_ostream<_CharT, _Traits> &out) {
686   out << 'c' << 'o' << 'n' << 's' << 't' << '_' << 'c' << 'a' << 's' << 't'
687       << ' ';
688   return out;
689 }
690 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
691 inline _TMS2_INLINE_VISIBILITY
692 ::std::basic_ostream<_CharT, _Traits> &
693 _continue(::std::basic_ostream<_CharT, _Traits> &out) {
694   out << 'c' << 'o' << 'n' << 't' << 'i' << 'n' << 'u' << 'e' << ' ';
695   return out;
696 }
697 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
698 inline _TMS2_INLINE_VISIBILITY
699 ::std::basic_ostream<_CharT, _Traits> &
700 _co_await(::std::basic_ostream<_CharT, _Traits> &out) {
701   out << 'c' << 'o' << '_' << 'a' << 'w' << 'a' << 'i' << 't' << ' ';
```

```
702 |   return out;
703 | }
704 | template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
705 | inline _TMS2_INLINE_VISIBILITY
706 | ::std::basic_ostream<_CharT, _Traits> &
707 | _co_return(::std::basic_ostream<_CharT, _Traits> &out) {
708 |   out << 'c' << 'o' << '_' << 'r' << 'e' << 't' << 'u' << 'r' << 'n' << ' ';
709 |   return out;
710 | }
711 | template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
712 | inline _TMS2_INLINE_VISIBILITY
713 | ::std::basic_ostream<_CharT, _Traits> &
714 | _co_yield(::std::basic_ostream<_CharT, _Traits> &out) {
715 |   out << 'c' << 'o' << '_' << 'y' << 'i' << 'e' << 'l' << 'd' << ' ';
716 |   return out;
717 | }
718 | template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
719 | inline _TMS2_INLINE_VISIBILITY
720 | ::std::basic_ostream<_CharT, _Traits> &
721 | _decltype(::std::basic_ostream<_CharT, _Traits> &out) {
722 |   out << 'd' << 'e' << 'c' << 'l' << 't' << 'y' << 'p' << 'e' << ' ';
723 |   return out;
724 | }
725 | template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
726 | inline _TMS2_INLINE_VISIBILITY
727 | ::std::basic_ostream<_CharT, _Traits> &
728 | _default(::std::basic_ostream<_CharT, _Traits> &out) {
729 |   out << 'd' << 'e' << 'f' << 'a' << 'u' << 'l' << 't' << ' ';
730 |   return out;
731 | }
732 | template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
733 | inline _TMS2_INLINE_VISIBILITY
734 | ::std::basic_ostream<_CharT, _Traits> &
735 | _delete(::std::basic_ostream<_CharT, _Traits> &out) {
736 |   out << 'd' << 'e' << 'l' << 'e' << 't' << 'e' << ' ';
737 |   return out;
738 | }
739 | template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
740 | inline _TMS2_INLINE_VISIBILITY
741 | ::std::basic_ostream<_CharT, _Traits> &
742 | _do(::std::basic_ostream<_CharT, _Traits> &out) {
743 |   out << 'd' << 'o' << ' ';
744 |   return out;
745 | }
746 | template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
747 | inline _TMS2_INLINE_VISIBILITY
748 | ::std::basic_ostream<_CharT, _Traits> &
749 | _double(::std::basic_ostream<_CharT, _Traits> &out) {
750 |   out << 'd' << 'o' << 'u' << 'b' << 'l' << 'e' << ' ';
751 |   return out;
752 | }
753 | template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
754 | inline _TMS2_INLINE_VISIBILITY
755 | ::std::basic_ostream<_CharT, _Traits> &
756 | _dynamic_cast(::std::basic_ostream<_CharT, _Traits> &out) {
757 |   out << 'd' << 'y' << 'n' << 'a' << 'm' << 'i' << 'c' << '_' << 'c'
758 |       << 'a' << 's' << 't' << ' ';
759 |   return out;
760 | }
761 | template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
762 | inline _TMS2_INLINE_VISIBILITY
763 | ::std::basic_ostream<_CharT, _Traits> &
764 | _else(::std::basic_ostream<_CharT, _Traits> &out) {
765 |   out << 'e' << 'l' << 's' << 'e' << ' ';
766 |   return out;
767 | }
768 | template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
769 | inline _TMS2_INLINE_VISIBILITY
770 | ::std::basic_ostream<_CharT, _Traits> &
771 | _enum(::std::basic_ostream<_CharT, _Traits> &out) {
772 |   out << 'e' << 'n' << 'u' << 'm' << ' ';
773 |   return out;
774 | }
```

```
775 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
776 inline _TMS2_INLINE_VISIBILITY
777 ::std::basic_ostream<_CharT, _Traits> &
778 _explicit(::std::basic_ostream<_CharT, _Traits> &out) {
779   out << 'e' << 'x' << 'p' << 'l' << 'i' << 'c' << 'i' << 't' << ' ';
780   return out;
781 }
782 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
783 inline _TMS2_INLINE_VISIBILITY
784 ::std::basic_ostream<_CharT, _Traits> &
785 _export(::std::basic_ostream<_CharT, _Traits> &out) {
786   out << 'e' << 'x' << 'p' << 'o' << 'r' << 't' << ' ';
787   return out;
788 }
789 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
790 inline _TMS2_INLINE_VISIBILITY
791 ::std::basic_ostream<_CharT, _Traits> &
792 _extern(::std::basic_ostream<_CharT, _Traits> &out) {
793   out << 'e' << 'x' << 't' << 'e' << 'r' << 'n' << ' ';
794   return out;
795 }
796 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
797 inline _TMS2_INLINE_VISIBILITY
798 ::std::basic_ostream<_CharT, _Traits> &
799 _false(::std::basic_ostream<_CharT, _Traits> &out) {
800   out << 'f' << 'a' << 'l' << 's' << 'e' << ' ';
801   return out;
802 }
803 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
804 inline _TMS2_INLINE_VISIBILITY
805 ::std::basic_ostream<_CharT, _Traits> &
806 _float(::std::basic_ostream<_CharT, _Traits> &out) {
807   out << 'f' << 'l' << 'o' << 'a' << 't' << ' ';
808   return out;
809 }
810 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
811 inline _TMS2_INLINE_VISIBILITY
812 ::std::basic_ostream<_CharT, _Traits> &
813 _for(::std::basic_ostream<_CharT, _Traits> &out) {
814   out << 'f' << 'o' << 'r' << ' ';
815   return out;
816 }
817 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
818 inline _TMS2_INLINE_VISIBILITY
819 ::std::basic_ostream<_CharT, _Traits> &
820 _friend(::std::basic_ostream<_CharT, _Traits> &out) {
821   out << 'f' << 'r' << 'i' << 'e' << 'n' << 'd' << ' ';
822   return out;
823 }
824 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
825 inline _TMS2_INLINE_VISIBILITY
826 ::std::basic_ostream<_CharT, _Traits> &
827 _goto(::std::basic_ostream<_CharT, _Traits> &out) {
828   out << 'g' << 'o' << 't' << 'o' << ' ';
829   return out;
830 }
831 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
832 inline _TMS2_INLINE_VISIBILITY
833 ::std::basic_ostream<_CharT, _Traits> &
834 _if(::std::basic_ostream<_CharT, _Traits> &out) {
835   out << 'i' << 'f' << ' ';
836   return out;
837 }
838 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
839 inline _TMS2_INLINE_VISIBILITY
840 ::std::basic_ostream<_CharT, _Traits> &
841 _import(::std::basic_ostream<_CharT, _Traits> &out) {
842   out << 'i' << 'm' << 'p' << 'o' << 'r' << 't' << ' ';
843   return out;
844 }
845 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
846 inline _TMS2_INLINE_VISIBILITY
847 ::std::basic_ostream<_CharT, _Traits> &
```

```
848 _inline(::std::basic_ostream<_CharT, _Traits> &out) {
849   out << 'i' << 'n' << 'l' << 'i' << 'n' << 'e' << ' ';
850   return out;
851 }
852 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
853 inline _TMS2_INLINE_VISIBILITY
854 ::std::basic_ostream<_CharT, _Traits> &
855 _int(::std::basic_ostream<_CharT, _Traits> &out) {
856   out << 'i' << 'n' << 't' << ' ';
857   return out;
858 }
859 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
860 inline _TMS2_INLINE_VISIBILITY
861 ::std::basic_ostream<_CharT, _Traits> &
862 _long(::std::basic_ostream<_CharT, _Traits> &out) {
863   out << 'l' << 'o' << 'n' << 'g' << ' ';
864   return out;
865 }
866 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
867 inline _TMS2_INLINE_VISIBILITY
868 ::std::basic_ostream<_CharT, _Traits> &
869 _module(::std::basic_ostream<_CharT, _Traits> &out) {
870   out << 'm' << 'o' << 'd' << 'u' << 'l' << 'e' << ' ';
871   return out;
872 }
873 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
874 inline _TMS2_INLINE_VISIBILITY
875 ::std::basic_ostream<_CharT, _Traits> &
876 _mutable(::std::basic_ostream<_CharT, _Traits> &out) {
877   out << 'm' << 'u' << 't' << 'a' << 'b' << 'l' << 'e' << ' ';
878   return out;
879 }
880 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
881 inline _TMS2_INLINE_VISIBILITY
882 ::std::basic_ostream<_CharT, _Traits> &
883 _namespace(::std::basic_ostream<_CharT, _Traits> &out) {
884   out << 'n' << 'a' << 'm' << 'e' << 's' << 'p' << 'a' << 'c' << 'e' << ' ';
885   return out;
886 }
887 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
888 inline _TMS2_INLINE_VISIBILITY
889 ::std::basic_ostream<_CharT, _Traits> &
890 _new(::std::basic_ostream<_CharT, _Traits> &out) {
891   out << 'n' << 'e' << 'w' << ' ';
892   return out;
893 }
894 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
895 inline _TMS2_INLINE_VISIBILITY
896 ::std::basic_ostream<_CharT, _Traits> &
897 _noexcept(::std::basic_ostream<_CharT, _Traits> &out) {
898   out << 'n' << 'o' << 'e' << 'x' << 'c' << 'e' << 'p' << 't' << ' ';
899   return out;
900 }
901 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
902 inline _TMS2_INLINE_VISIBILITY
903 ::std::basic_ostream<_CharT, _Traits> &
904 _not(::std::basic_ostream<_CharT, _Traits> &out) {
905   out << 'n' << 'o' << 't' << ' ';
906   return out;
907 }
908 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
909 inline _TMS2_INLINE_VISIBILITY
910 ::std::basic_ostream<_CharT, _Traits> &
911 _not_eq(::std::basic_ostream<_CharT, _Traits> &out) {
912   out << 'n' << 'o' << 't' << '_' << 'e' << 'q' << ' ';
913   return out;
914 }
915 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
916 inline _TMS2_INLINE_VISIBILITY
917 ::std::basic_ostream<_CharT, _Traits> &
918 _nullptr(::std::basic_ostream<_CharT, _Traits> &out) {
919   out << 'n' << 'u' << 'l' << 'l' << 'p' << 't' << 'r' << ' ';
920   return out;
```

```
921 }
922 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
923 inline _TMS2_INLINE_VISIBILITY
924 ::std::basic_ostream<_CharT, _Traits> &
925 _operator(::std::basic_ostream<_CharT, _Traits> &out) {
926   out << 'o' << 'p' << 'e' << 'r' << 'a' << 't' << 'o' << 'r' << ' ';
927   return out;
928 }
929 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
930 inline _TMS2_INLINE_VISIBILITY
931 ::std::basic_ostream<_CharT, _Traits> &
932 _or(::std::basic_ostream<_CharT, _Traits> &out) {
933   out << ' ' << '|' << '|' << ' ';
934   return out;
935 }
936 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
937 inline _TMS2_INLINE_VISIBILITY
938 ::std::basic_ostream<_CharT, _Traits> &
939 _or_eq(::std::basic_ostream<_CharT, _Traits> &out) {
940   out << 'o' << 'r' << '_' << 'e' << 'q' << ' ';
941   return out;
942 }
943 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
944 inline _TMS2_INLINE_VISIBILITY
945 ::std::basic_ostream<_CharT, _Traits> &
946 _private(::std::basic_ostream<_CharT, _Traits> &out) {
947   out << 'p' << 'r' << 'i' << 'v' << 'a' << 't' << 'e' << ' ';
948   return out;
949 }
950 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
951 inline _TMS2_INLINE_VISIBILITY
952 ::std::basic_ostream<_CharT, _Traits> &
953 _protected(::std::basic_ostream<_CharT, _Traits> &out) {
954   out << 'p' << 'r' << 'o' << 't' << 'e' << 'c' << 't' << 'e' << 'd' << ' ';
955   return out;
956 }
957 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
958 inline _TMS2_INLINE_VISIBILITY
959 ::std::basic_ostream<_CharT, _Traits> &
960 _public(::std::basic_ostream<_CharT, _Traits> &out) {
961   out << 'p' << 'u' << 'b' << 'l' << 'i' << 'c' << ' ';
962   return out;
963 }
964 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
965 inline _TMS2_INLINE_VISIBILITY
966 ::std::basic_ostream<_CharT, _Traits> &
967 _register(::std::basic_ostream<_CharT, _Traits> &out) {
968   out << 'r' << 'e' << 'g' << 'i' << 's' << 't' << 'e' << 'r' << ' ';
969   return out;
970 }
971 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
972 inline _TMS2_INLINE_VISIBILITY
973 ::std::basic_ostream<_CharT, _Traits> &
974 _reinterpret_cast(::std::basic_ostream<_CharT, _Traits> &out) {
975   out << 'r' << 'e' << 'i' << 'n' << 't' << 'e' << 'r' << 'p' << 'r'
976       << 'e' << 't' << '_' << 'c' << 'a' << 's' << 't' << ' ';
977   return out;
978 }
979 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
980 inline _TMS2_INLINE_VISIBILITY
981 ::std::basic_ostream<_CharT, _Traits> &
982 _requires(::std::basic_ostream<_CharT, _Traits> &out) {
983   out << 'r' << 'e' << 'q' << 'u' << 'i' << 'r' << 'e' << 's' << ' ';
984   return out;
985 }
986 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
987 inline _TMS2_INLINE_VISIBILITY
988 ::std::basic_ostream<_CharT, _Traits> &
989 _return(::std::basic_ostream<_CharT, _Traits> &out) {
990   out << 'r' << 'e' << 't' << 'u' << 'r' << 'n' << ' ';
991   return out;
992 }
993 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
```

```
994  inline _TMS2_INLINE_VISIBILITY
995  ::std::basic_ostream<_CharT, _Traits> &
996  _short(::std::basic_ostream<_CharT, _Traits> &out) {
997    out << 's' << 'h' << 'o' << 'r' << 't' << ' ';
998    return out;
999  }
1000 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1001 inline _TMS2_INLINE_VISIBILITY
1002 ::std::basic_ostream<_CharT, _Traits> &
1003 _signed(::std::basic_ostream<_CharT, _Traits> &out) {
1004   out << 's' << 'i' << 'g' << 'n' << 'e' << 'd' << ' ';
1005   return out;
1006 }
1007 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1008 inline _TMS2_INLINE_VISIBILITY
1009 ::std::basic_ostream<_CharT, _Traits> &
1010 _sizeof(::std::basic_ostream<_CharT, _Traits> &out) {
1011   out << 's' << 'i' << 'z' << 'e' << 'o' << 'f' << ' ';
1012   return out;
1013 }
1014 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1015 inline _TMS2_INLINE_VISIBILITY
1016 ::std::basic_ostream<_CharT, _Traits> &
1017 _static(::std::basic_ostream<_CharT, _Traits> &out) {
1018   out << 's' << 't' << 'a' << 't' << 'i' << 'c' << ' ';
1019   return out;
1020 }
1021 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1022 inline _TMS2_INLINE_VISIBILITY
1023 ::std::basic_ostream<_CharT, _Traits> &
1024 _static_assert(::std::basic_ostream<_CharT, _Traits> &out) {
1025   out << 's' << 't' << 'a' << 't' << 'i' << 'c' << '_' << 'a' << 's'
1026       << 's' << 'e' << 'r' << 't' << ' ';
1027   return out;
1028 }
1029 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1030 inline _TMS2_INLINE_VISIBILITY
1031 ::std::basic_ostream<_CharT, _Traits> &
1032 _static_cast(::std::basic_ostream<_CharT, _Traits> &out) {
1033   out << 's' << 't' << 'a' << 't' << 'i' << 'c' << '_' << 'c' << 'a'
1034       << 's' << 't' << ' ';
1035   return out;
1036 }
1037 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1038 inline _TMS2_INLINE_VISIBILITY
1039 ::std::basic_ostream<_CharT, _Traits> &
1040 _struct(::std::basic_ostream<_CharT, _Traits> &out) {
1041   out << 's' << 't' << 'r' << 'u' << 'c' << 't' << ' ';
1042   return out;
1043 }
1044 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1045 inline _TMS2_INLINE_VISIBILITY
1046 ::std::basic_ostream<_CharT, _Traits> &
1047 _switch(::std::basic_ostream<_CharT, _Traits> &out) {
1048   out << 's' << 'w' << 'i' << 't' << 'c' << 'h' << ' ';
1049   return out;
1050 }
1051 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1052 inline _TMS2_INLINE_VISIBILITY
1053 ::std::basic_ostream<_CharT, _Traits> &
1054 _synchronized(::std::basic_ostream<_CharT, _Traits> &out) {
1055   out << 's' << 'y' << 'n' << 'c' << 'h' << 'r' << 'o' << 'n' << 'i'
1056       << 'z' << 'e' << 'd' << ' ';
1057   return out;
1058 }
1059 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1060 inline _TMS2_INLINE_VISIBILITY
1061 ::std::basic_ostream<_CharT, _Traits> &
1062 _template(::std::basic_ostream<_CharT, _Traits> &out) {
1063   out << 't' << 'e' << 'm' << 'p' << 'l' << 'a' << 't' << 'e' << ' ';
1064   return out;
1065 }
1066 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
```

```
1067 inline _TMS2_INLINE_VISIBILITY
1068 ::std::basic_ostream<_CharT, _Traits> &
1069 _this(::std::basic_ostream<_CharT, _Traits> &out) {
1070   out << 't' << 'h' << 'i' << 's' << ' ';
1071   return out;
1072 }
1073 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1074 inline _TMS2_INLINE_VISIBILITY
1075 ::std::basic_ostream<_CharT, _Traits> &
1076 _thread_local(::std::basic_ostream<_CharT, _Traits> &out) {
1077   out << 't' << 'h' << 'r' << 'e' << 'a' << 'd' << '_' << 'l' << 'o'
1078       << 'c' << 'a' << 'l' << ' ';
1079   return out;
1080 }
1081 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1082 inline _TMS2_INLINE_VISIBILITY
1083 ::std::basic_ostream<_CharT, _Traits> &
1084 _throw(::std::basic_ostream<_CharT, _Traits> &out) {
1085   out << 't' << 'h' << 'r' << 'o' << 'w' << ' ';
1086   return out;
1087 }
1088 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1089 inline _TMS2_INLINE_VISIBILITY
1090 ::std::basic_ostream<_CharT, _Traits> &
1091 _true(::std::basic_ostream<_CharT, _Traits> &out) {
1092   out << 't' << 'r' << 'u' << 'e' << ' ';
1093   return out;
1094 }
1095 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1096 inline _TMS2_INLINE_VISIBILITY
1097 ::std::basic_ostream<_CharT, _Traits> &
1098 _try(::std::basic_ostream<_CharT, _Traits> &out) {
1099   out << 't' << 'r' << 'y' << ' ';
1100   return out;
1101 }
1102 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1103 inline _TMS2_INLINE_VISIBILITY
1104 ::std::basic_ostream<_CharT, _Traits> &
1105 _typedef(::std::basic_ostream<_CharT, _Traits> &out) {
1106   out << 't' << 'y' << 'p' << 'e' << 'd' << 'e' << 'f' << ' ';
1107   return out;
1108 }
1109 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1110 inline _TMS2_INLINE_VISIBILITY
1111 ::std::basic_ostream<_CharT, _Traits> &
1112 _typeid(::std::basic_ostream<_CharT, _Traits> &out) {
1113   out << 't' << 'y' << 'p' << 'e' << 'i' << 'd' << ' ';
1114   return out;
1115 }
1116 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1117 inline _TMS2_INLINE_VISIBILITY
1118 ::std::basic_ostream<_CharT, _Traits> &
1119 _typename(::std::basic_ostream<_CharT, _Traits> &out) {
1120   out << 't' << 'y' << 'p' << 'e' << 'n' << 'a' << 'm' << 'e' << ' ';
1121   return out;
1122 }
1123 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1124 inline _TMS2_INLINE_VISIBILITY
1125 ::std::basic_ostream<_CharT, _Traits> &
1126 _union(::std::basic_ostream<_CharT, _Traits> &out) {
1127   out << 'u' << 'n' << 'i' << 'o' << 'n' << ' ';
1128   return out;
1129 }
1130 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1131 inline _TMS2_INLINE_VISIBILITY
1132 ::std::basic_ostream<_CharT, _Traits> &
1133 _unsigned(::std::basic_ostream<_CharT, _Traits> &out) {
1134   out << 'u' << 'n' << 's' << 'i' << 'g' << 'n' << 'e' << 'd' << ' ';
1135   return out;
1136 }
1137 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1138 inline _TMS2_INLINE_VISIBILITY
1139 ::std::basic_ostream<_CharT, _Traits> &
```

```
1140 _using(::std::basic_ostream<_CharT, _Traits> &out) {
1141   out << 'u' << 's' << 'i' << 'n' << 'g' << ' ';
1142   return out;
1143 }
1144 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1145 inline _TMS2_INLINE_VISIBILITY
1146 ::std::basic_ostream<_CharT, _Traits> &
1147 _virtual(::std::basic_ostream<_CharT, _Traits> &out) {
1148   out << 'v' << 'i' << 'r' << 't' << 'u' << 'a' << 'l' << ' ';
1149   return out;
1150 }
1151 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1152 inline _TMS2_INLINE_VISIBILITY
1153 ::std::basic_ostream<_CharT, _Traits> &
1154 _void(::std::basic_ostream<_CharT, _Traits> &out) {
1155   out << 'v' << 'o' << 'i' << 'd' << ' ';
1156   return out;
1157 }
1158 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1159 inline _TMS2_INLINE_VISIBILITY
1160 ::std::basic_ostream<_CharT, _Traits> &
1161 _volatile(::std::basic_ostream<_CharT, _Traits> &out) {
1162   out << 'v' << 'o' << 'l' << 'a' << 't' << 'i' << 'l' << 'e' << ' ';
1163   return out;
1164 }
1165 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1166 inline _TMS2_INLINE_VISIBILITY
1167 ::std::basic_ostream<_CharT, _Traits> &
1168 _wchar_t(::std::basic_ostream<_CharT, _Traits> &out) {
1169   out << 'w' << 'c' << 'h' << 'a' << 'r' << '_' << 't' << ' ';
1170   return out;
1171 }
1172 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1173 inline _TMS2_INLINE_VISIBILITY
1174 ::std::basic_ostream<_CharT, _Traits> &
1175 _while(::std::basic_ostream<_CharT, _Traits> &out) {
1176   out << 'w' << 'h' << 'i' << 'l' << 'e' << ' ';
1177   return out;
1178 }
1179 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1180 inline _TMS2_INLINE_VISIBILITY
1181 ::std::basic_ostream<_CharT, _Traits> &
1182 _xor(::std::basic_ostream<_CharT, _Traits> &out) {
1183   out << 'x' << 'o' << 'r' << ' ';
1184   return out;
1185 }
1186 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1187 inline _TMS2_INLINE_VISIBILITY
1188 ::std::basic_ostream<_CharT, _Traits> &
1189 _xor_eq(::std::basic_ostream<_CharT, _Traits> &out) {
1190   out << 'x' << 'o' << 'r' << '_' << 'e' << 'q' << ' ';
1191   return out;
1192 }
1193 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1194 inline _TMS2_INLINE_VISIBILITY
1195 ::std::basic_ostream<_CharT, _Traits> &
1196 _dot(::std::basic_ostream<_CharT, _Traits> &out) {
1197   out << '.';
1198   return out;
1199 }
1200 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1201 inline _TMS2_INLINE_VISIBILITY
1202 ::std::basic_ostream<_CharT, _Traits> &
1203 _access(::std::basic_ostream<_CharT, _Traits> &out) {
1204   out << '-' << '>';
1205   return out;
1206 }
1207 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1208 inline _TMS2_INLINE_VISIBILITY
1209 ::std::basic_ostream<_CharT, _Traits> &
1210 _IF(::std::basic_ostream<_CharT, _Traits> &out) {
1211   out << '#' << 'i' << 'f' << ' ';
1212   return out;
```

```
1213 }
1214 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1215 inline _TMS2_INLINE_VISIBILITY
1216 ::std::basic_ostream<_CharT, _Traits> &
1217 _ELIF(::std::basic_ostream<_CharT, _Traits> &out) {
1218   out << '#' << 'e' << 'l' << 'i' << 'f' << ' ';
1219   return out;
1220 }
1221 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1222 inline _TMS2_INLINE_VISIBILITY
1223 ::std::basic_ostream<_CharT, _Traits> &
1224 _ELSE(::std::basic_ostream<_CharT, _Traits> &out) {
1225   out << '#' << 'e' << 'l' << 's' << 'e' << ' ';
1226   return out;
1227 }
1228 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1229 inline _TMS2_INLINE_VISIBILITY
1230 ::std::basic_ostream<_CharT, _Traits> &
1231 _ENDIF(::std::basic_ostream<_CharT, _Traits> &out) {
1232   out << '#' << 'e' << 'n' << 'd' << 'i' << 'f' << ' ';
1233   return out;
1234 }
1235 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1236 inline _TMS2_INLINE_VISIBILITY
1237 ::std::basic_ostream<_CharT, _Traits> &
1238 _DEFINED(::std::basic_ostream<_CharT, _Traits> &out) {
1239   out << '#' << 'd' << 'e' << 'f' << 'i' << 'n' << 'e' << 'd' << ' ';
1240   return out;
1241 }
1242 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1243 inline _TMS2_INLINE_VISIBILITY
1244 ::std::basic_ostream<_CharT, _Traits> &
1245 _IFDEF(::std::basic_ostream<_CharT, _Traits> &out) {
1246   out << '#' << 'i' << 'f' << 'd' << 'e' << 'f' << ' ';
1247   return out;
1248 }
1249 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1250 inline _TMS2_INLINE_VISIBILITY
1251 ::std::basic_ostream<_CharT, _Traits> &
1252 _IFNDEF(::std::basic_ostream<_CharT, _Traits> &out) {
1253   out << '#' << 'i' << 'f' << 'n' << 'd' << 'e' << 'f' << ' ';
1254   return out;
1255 }
1256 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1257 inline _TMS2_INLINE_VISIBILITY
1258 ::std::basic_ostream<_CharT, _Traits> &
1259 _DEFINE(::std::basic_ostream<_CharT, _Traits> &out) {
1260   out << '#' << 'd' << 'e' << 'f' << 'i' << 'n' << 'e' << ' ';
1261   return out;
1262 }
1263 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1264 inline _TMS2_INLINE_VISIBILITY
1265 ::std::basic_ostream<_CharT, _Traits> &
1266 _UNDEF(::std::basic_ostream<_CharT, _Traits> &out) {
1267   out << '#' << 'u' << 'n' << 'd' << 'e' << 'f' << ' ';
1268   return out;
1269 }
1270 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1271 inline _TMS2_INLINE_VISIBILITY
1272 ::std::basic_ostream<_CharT, _Traits> &
1273 _INCLUDE(::std::basic_ostream<_CharT, _Traits> &out) {
1274   out << '#' << 'i' << 'n' << 'c' << 'l' << 'u' << 'd' << 'e' << ' ';
1275   return out;
1276 }
1277 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1278 inline _TMS2_INLINE_VISIBILITY
1279 ::std::basic_ostream<_CharT, _Traits> &
1280 _LINE(::std::basic_ostream<_CharT, _Traits> &out) {
1281   out << '#' << 'l' << 'i' << 'n' << 'e' << ' ';
1282   return out;
1283 }
1284 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1285 inline _TMS2_INLINE_VISIBILITY
```

```
1286 ::std::basic_ostream<_CharT, _Traits> &
1287 _ERROR(::std::basic_ostream<_CharT, _Traits> &out) {
1288   out << '#' << 'e' << 'r' << 'r' << 'o' << 'r' << ' ';
1289   return out;
1290 }
1291 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1292 inline _TMS2_INLINE_VISIBILITY
1293 ::std::basic_ostream<_CharT, _Traits> &
1294 _PRAGMA(::std::basic_ostream<_CharT, _Traits> &out) {
1295   out << '#' << 'p' << 'r' << 'a' << 'g' << 'm' << 'a' << ' ';
1296   return out;
1297 }
1298 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1299 inline _TMS2_INLINE_VISIBILITY
1300 ::std::basic_ostream<_CharT, _Traits> &
1301 _colon(::std::basic_ostream<_CharT, _Traits> &out) {
1302   out << ':' << ' ';
1303   return out;
1304 }
1305 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1306 inline _TMS2_INLINE_VISIBILITY
1307 ::std::basic_ostream<_CharT, _Traits> &
1308 _semicolon(::std::basic_ostream<_CharT, _Traits> &out) {
1309   out << ';' << ' ';
1310   return out;
1311 }
1312 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1313 inline _TMS2_INLINE_VISIBILITY
1314 ::std::basic_ostream<_CharT, _Traits> &
1315 _comma(::std::basic_ostream<_CharT, _Traits> &out) {
1316   out << ',' << ' ';
1317   return out;
1318 }
1319 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1320 inline _TMS2_INLINE_VISIBILITY
1321 ::std::basic_ostream<_CharT, _Traits> &
1322 _space(::std::basic_ostream<_CharT, _Traits> &out) {
1323   out << ' ';
1324   return out;
1325 }
1326 template<typename _CharT, typename _Traits=::std::char_traits<_CharT> >
1327 inline _TMS2_INLINE_VISIBILITY
1328 ::std::basic_ostream<_CharT, _Traits> &
1329 _scope(::std::basic_ostream<_CharT, _Traits> &out) {
1330   out << ':' << ':';
1331   return out;
1332 }
1333 template<typename _CharT, typename _Traits=::std::char_traits<char>>
1334 inline _TMS2_INLINE_VISIBILITY
1335 ::std::basic_ostream<_CharT, _Traits> &
1336 _lt(::std::basic_ostream<_CharT, _Traits> &out) {
1337   out << ' ' << '<' << ' ';
1338   return out;
1339 }
1340 template<typename _CharT, typename _Traits=::std::char_traits<char>>
1341 inline _TMS2_INLINE_VISIBILITY
1342 ::std::basic_ostream<_CharT, _Traits> &
1343 _lte(::std::basic_ostream<_CharT, _Traits> &out) {
1344   out << ' ' << '<' << '=' << ' ';
1345   return out;
1346 }
1347 template<typename _CharT, typename _Traits=::std::char_traits<char>>
1348 inline _TMS2_INLINE_VISIBILITY
1349 ::std::basic_ostream<_CharT, _Traits> &
1350 _ltlt(::std::basic_ostream<_CharT, _Traits> &out) {
1351   out << ' ' << '<' << '<' << ' ';
1352   return out;
1353 }
1354 template<typename _CharT, typename _Traits=::std::char_traits<char>>
1355 inline _TMS2_INLINE_VISIBILITY
1356 ::std::basic_ostream<_CharT, _Traits> &
1357 _gt(::std::basic_ostream<_CharT, _Traits> &out) {
1358   out << ' ' << '>' << ' ';
```

```
1359 |   return out;
1360 | }
1361 | template<typename _CharT, typename _Traits=::std::char_traits<char>>
1362 | inline _TMS2_INLINE_VISIBILITY
1363 | ::std::basic_ostream<_CharT, _Traits> &
1364 | _gte(::std::basic_ostream<_CharT, _Traits> &out) {
1365 |   out << ' ' << '>' << '=' << ' ';
1366 |   return out;
1367 | }
1368 | template<typename _CharT, typename _Traits=::std::char_traits<char>>
1369 | inline _TMS2_INLINE_VISIBILITY
1370 | ::std::basic_ostream<_CharT, _Traits> &
1371 | _gtgt(::std::basic_ostream<_CharT, _Traits> &out) {
1372 |   out << ' ' << '>' << '>' << ' ';
1373 |   return out;
1374 | }
1375 | template<typename _CharT, typename _Traits=::std::char_traits<char>>
1376 | inline _TMS2_INLINE_VISIBILITY
1377 | ::std::basic_ostream<_CharT, _Traits> &
1378 | _inc(::std::basic_ostream<_CharT, _Traits> &out) {
1379 |   out << '+' << '+';
1380 |   return out;
1381 | }
1382 | template<typename _CharT, typename _Traits=::std::char_traits<char>>
1383 | inline _TMS2_INLINE_VISIBILITY
1384 | ::std::basic_ostream<_CharT, _Traits> &
1385 | _dec(::std::basic_ostream<_CharT, _Traits> &out) {
1386 |   out << '-' << '-';
1387 |   return out;
1388 | }
1389 | template<typename _CharT, typename _Traits=::std::char_traits<char>>
1390 | inline _TMS2_INLINE_VISIBILITY
1391 | ::std::basic_ostream<_CharT, _Traits> &
1392 | _set(::std::basic_ostream<_CharT, _Traits> &out) {
1393 |   out << ' ' << '=' << ' ';
1394 |   return out;
1395 | }
1396 | template<typename _CharT, typename _Traits=::std::char_traits<char>>
1397 | inline _TMS2_INLINE_VISIBILITY
1398 | ::std::basic_ostream<_CharT, _Traits> &
1399 | _eq(::std::basic_ostream<_CharT, _Traits> &out) {
1400 |   out << ' ' << '=' << '=' << ' ';
1401 |   return out;
1402 | }
1403 | template<typename _CharT, typename _Traits=::std::char_traits<char>>
1404 | inline _TMS2_INLINE_VISIBILITY
1405 | ::std::basic_ostream<_CharT, _Traits> &
1406 | _neq(::std::basic_ostream<_CharT, _Traits> &out) {
1407 |   out << ' ' << '!' << '=' << ' ';
1408 |   return out;
1409 | }
1410 | template<typename _CharT, typename _Traits=::std::char_traits<char>>
1411 | inline _TMS2_INLINE_VISIBILITY
1412 | ::std::basic_ostream<_CharT, _Traits> &
1413 | _formatf(::std::basic_ostream<_CharT, _Traits> &out) {
1414 |   out.fill(' ');
1415 |   out.width(9);
1416 |   out.precision(5);
1417 |   return out;
1418 | }
1419 | template<typename _CharT, typename _Traits=::std::char_traits<char>>
1420 | inline _TMS2_INLINE_VISIBILITY
1421 | ::std::basic_ostream<_CharT, _Traits> &
1422 | _formati(::std::basic_ostream<_CharT, _Traits> &out) {
1423 |   out.fill(' ');
1424 |   out.width(9);
1425 |   return out;
1426 | }
1427 | #if __cplusplus >= 201703L
1428 | #define TabOnce true
1429 | #else
1430 | #define TabOnce *IvarRef = &Ivar; IvarRef; IvarRef = nullptr
1431 | #endif
```

```
#if __cplusplus >= 201703L
#define Ident \
if (ident<char> Ivar(Out, Level); true)
#else
#define Ident \
for (ident<char> Ivar(Out, Level), TabOnce)
#endif
#if __cplusplus >= 201703L
#define Init \
if (initialize_<char> Ivar(Out, Level); true)
#else
#define Init \
for (initialize_<char> Ivar(Out, Level), TabOnce)
#endif
#if __cplusplus >= 201703L
#define Pars \
if (parenthesis_<char> Ivar(Out, Level); true)
#else
#define Pars \
for (parenthesis_<char> Ivar(Out, Level), TabOnce)
#endif
#if __cplusplus >= 201703L
#define Brcs \
if (braces_<char> Ivar(Out, Level); true)
#else
#define Brcs \
for (braces_<char> Ivar(Out, Level), TabOnce)
#endif
#if __cplusplus >= 201703L
#define Brks \
if (brackets_<char> Ivar(Out, Level); true)
#else
#define Brks \
for (brackets_<char> Ivar(Out, Level), TabOnce)
#endif
#if __cplusplus >= 201703L
#define SBrks \
if (sbrackets_<char> Ivar(Out, Level); true)
#else
#define SBrks \
for (sbrackets_<char> Ivar(Out, Level), TabOnce)
#endif
```

#### 6.1.5.1. Line control on generated source code

```
#ifdef NDEBUG
#define LineInfo \
Out << ::std::endl << ::std::setw(2 * Level - 2) << ::std::setfill(' ') << \
_LINE << __LINE__ << ' ' << '\"' << __FILE__<< '"' \
<< ::std::endl << ::std::setw(2 * Level - 2) << ::std::setfill(' ') << ' '
#else
#define LineInfo \
Out << ::std::endl << ::std::setw(2 * Level - 1) << ::std::setfill(' ') << \
_LINE << __LINE__ << ' ' << '\"' << __FILE__<< '"' \
<< ::std::endl << ::std::setw(2 * Level - 1) << ::std::setfill(' ') << ' '
#endif
#if __cplusplus >= 201703L
#define If(cond) \
if (braces_<char> Ivar(LineInfo << _if << ' ' << '(' << cond << ')', Level); true)
#else
#define If(cond) \
for (braces_<char> Ivar(LineInfo << _if << ' ' << '(' << cond << ')', Level),
    TabOnce)
#endif
#define Stmt(stmt) \
Tab << stmt << _semicolon;
#ifdef NDEBUG
#define Tab \
Out << Ivar
#else
#define Tab \
```

```
1500 |Out << Ivar << _LINE << __LINE__ << ' ' << '\"' << __FILE__<< '\"' << Ivar
1501 |#endif
1502 |#ifdef NDEBUG
1503 |#define ArgNL \
1504 |Ivar(true)
1505 |#else
1506 |#define ArgNL \
1507 |Ivar(true) << _LINE << __LINE__ << ' ' << '\"' << __FILE__<< '\"' << Ivar
1508 |#endif
1509 |#define ArgN \
1510 |Ivar(false)
1511 |#define PASTE_TEXT_(X, Y) X##Y
1512 |#define PASTE_TEXT(X, Y) PASTE_TEXT_(X,Y)
1513 |#define Out __out
1514 |#define Level __level
1515 |#define Ivar __tab
1516 |#define IvarRef __tabRef
1517 |#define Setup(out, level) \
1518 |::std::basic_ostream<char> &Out = out; \
1519 |::std::size_t &Level = level; \
1520 |::tms2::cpp::ident<char> Ivar(Out, Level); \
1521 |}
1522 |}
1523 |#endif //TMS2_PRINTER_H
```

**Listing 6.6:** `embedresource.cpp`: Resource source

```
1 //
```

## 2 6.1.6. Embedded resources support source

```
3  #include <fstream>
4  #include <boost/filesystem.hpp>
5  #include <tms2/printer.h>
6  namespace fs = boost::filesystem;
7  using namespace tms2::cpp;
8  int main(int argc, char **argv) {
9    if (argc < 3) {
10     fprintf(stderr,
11             "USAGE: %s {sym} {rsrc}\n\n"
12                 "  Creates {sym}.c from the contents of {rsrc}\n",
13             argv[0]);
14     return EXIT_FAILURE;
15   }
16   fs::path dst(argv[1]);
17   fs::path src(argv[2]);
18   ::std::string sym = src.filename().string();
19   replace(sym.begin(), sym.end(), '.', '_');
20   replace(sym.begin(), sym.end(), '-', '_');
21   create_directories(dst.parent_path());
22   ::std::fstream out(dst.c_str(), ::std::fstream::out);
23   ::std::fstream ifs(src.c_str(), ::std::fstream::in);
24   ::std::size_t level = 0;
25   Setup(out, level);
26   Out << _INCLUDE;
27   SBrks { Out << "stdlib.h"; };
28   Out << ::std::endl;
29   const auto resource_start = _member("resource_" + sym + "_start");
30   const auto resource_len = _member("resource_" + sym + "_len");
31   Tab << _const << _char << resource_start << '[' << ']' << _set;
32   Brcs {
33     Tab;
34     size_t lineCount = 0;
35     while (!ifs.eof()) {
36       char c;
37       ifs.get(c);
38       Out << "0x" << ::std::hex << (c & 0xff) << ", " << ::std::dec;
39       if (++lineCount == 10) {
40         Tab;
41         lineCount = 0;
42       }
43     }
44   }
45   Out << _semicolon;
46   Out << ::std::dec;
47   Stmt(_const << "size_t"_type << resource_len << _set << _sizeof
48               << par(resource_start));
49   return EXIT_SUCCESS;
50 }
```

**Listing 6.7:** `CMakeLists.txt`: Index header

```
cmake_minimum_required(VERSION 3.5)
project(tms2)
# ---------------------------------------------------------------------------
# PARAMETERS
# ---------------------------------------------------------------------------
# Param BUILD Type
if (NOT CMAKE_BUILD_TYPE)
    set(CMAKE_BUILD_TYPE RelWithDebInfo)
endif (NOT CMAKE_BUILD_TYPE)
# Param index maximum value
if (NOT tms2index_maximum)
    set(tms2index_maximum 700)
endif (NOT tms2index_maximum)
# Param index maximum error window value
if (NOT tms2index_window)
    set(tms2index_window 10)
endif (NOT tms2index_window)
# Param resolution value
if (NOT tms2index_resolution)
    set(tms2index_resolution 100)
endif (NOT tms2index_resolution)
# ---------------------------------------------------------------------------
# ADDITIONAL SUPPORT
# ---------------------------------------------------------------------------
# Add support for Uninstall
list(APPEND CMAKE_MODULE_PATH "${CMAKE_CURRENT_LIST_DIR}/CMake")
# Support for C++17
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -pedantic-errors")
set(CMAKE_CXX_STANDARD 14)
set(CMAKE_POSITION_INDEPENDENT_CODE ON)
#include(ExternalProject)
#add_subdirectory(lib/googletest)
#include_directories(lib/googletest)
#include_directories(lib/googletest/googletest/include)
# Include directories
include_directories(AFTER /usr/local/include)
# Define include library
include_directories(BEFORE include)
include_directories(AFTER src/tms2R/src)
if (APPLE)
    add_definitions(-DGTEST_USE_OWN_TR1_TUPLE)
    add_definitions(-D__GLIBCXX__)
endif (APPLE)
set_property(DIRECTORY PROPERTY ADDITIONAL_MAKE_CLEAN_FILES
        "${CMAKE_CURRENT_BINARY_DIR}/include"
        "${CMAKE_CURRENT_BINARY_DIR}/src"
        "${CMAKE_CURRENT_BINARY_DIR}/tms2R_0.1.0.tar.gz"
        "${CMAKE_SOURCE_DIR}/src/tms2R/src/index.h"
        "${CMAKE_SOURCE_DIR}/src/tms2R/src/spectrum.cpp"
        "${CMAKE_SOURCE_DIR}/src/tms2R/src/tms2/bitree.h"
        "${CMAKE_SOURCE_DIR}/src/tms2R/src/tms2/da.h"
        "${CMAKE_SOURCE_DIR}/src/tms2R/src/tms2/spectrum.h")
# ---------------------------------------------------------------------------
# LIBRARIES
# ---------------------------------------------------------------------------
# Search for Boost filesystem system program_options
find_package(Boost REQUIRED filesystem system program_options)
if (Boost_FOUND)
    include_directories(${Boost_INCLUDE_DIRS})
else ()
    message(FATAL_ERROR "Failed to find the Boost library.")
endif ()
# Search LibR
find_package(LibR REQUIRED)
if (${LIBR_FOUND})
    execute_process(
```

```
67 |             COMMAND ${LIBR_EXECUTABLE} "--slave" "-e" "stopifnot(require('Rcpp')); cat(Rcpp:::Rcpp.system.file('include'))"
68 |             OUTPUT_VARIABLE LIBRCPP_INCLUDE_DIRS
69 |     )
70 |     message("Rcpp: ${LIBRCPP_INCLUDE_DIRS}")
71 |     include_directories(BEFORE ${LIBR_INCLUDE_DIRS})
72 |     include_directories(BEFORE ${LIBRCPP_INCLUDE_DIRS})
73 |else ()
74 |     message(FATAL_ERROR "No R/Rcpp Library found...")
75 |endif ()
76 |# Embedded resources support
77 |set(EMBED_RES_INCLUDE_DIRS ${CMAKE_CURRENT_SOURCE_DIR})
78 |function(embed_resources out_var)
79 |     set(result)
80 |     foreach (in_f ${ARGN})
81 |         file(RELATIVE_PATH src_f ${CMAKE_SOURCE_DIR} ${CMAKE_CURRENT_SOURCE_DIR}/${in_f})
82 |         set(out_f "${PROJECT_BINARY_DIR}/${in_f}.c")
83 |         add_custom_command(OUTPUT ${out_f}
84 |                 COMMAND embed-resource ${out_f} ${src_f}
85 |                 DEPENDS ${in_f}
86 |                 WORKING_DIRECTORY ${CMAKE_SOURCE_DIR}
87 |                 COMMENT "Building binary file for embedding ${out_f}"
88 |                 VERBATIM)
89 |         list(APPEND result "${out_f}")
90 |         LIST(APPEND OUTPUT "${out_f}")
91 |     endforeach ()
92 |     set(${out_var} "${result}" PARENT_SCOPE)
93 |endfunction()
94 |add_executable(embed-resource src/cpp/embedresource.cpp)
95 |target_link_libraries(embed-resource
96 |         ${Boost_FILESYSTEM_LIBRARY}
97 |         ${Boost_SYSTEM_LIBRARY})
98 |set(interpret_SOURCE_FILES src/cpp/tms2search.cpp)
99 |list(APPEND interpret_SOURCE_FILES src/cpp/spectrum.cpp)
100|embed_resources(tms2indexgen_resources
101|         include/tms2/bitree.h
102|         include/tms2/spectrum.h
103|         src/cpp/spectrum.cpp
104|         CMake/cmake_uninstall.cmake.in)
105|add_executable(tms2indexgen ${tms2indexgen_resources} src/cpp/tms2index.cpp)
106|target_link_libraries(tms2indexgen LINK_PUBLIC ${Boost_LIBRARIES} pthread)
107|add_library(tms2 STATIC ${interpret_SOURCE_FILES})
108|install(TARGETS tms2
109|         RUNTIME DESTINATION bin
110|         LIBRARY DESTINATION lib
111|         ARCHIVE DESTINATION lib)
112|install(DIRECTORY include/ DESTINATION include)
113|add_custom_target(tms2R
114|         COMMAND ${LIBR_EXECUTABLE} CMD build ${CMAKE_SOURCE_DIR}/src/tms2R)
115|add_custom_target(tms2R-install
116|         COMMAND ${LIBR_EXECUTABLE} CMD INSTALL --preclean --no-multiarch --with-keep.source --debug --latex --html ${CMAKE_SOURCE_DIR}/src/tms2R)
117|add_dependencies(tms2R-install tms2R)
118|add_custom_target(tms2R-deploy
119|         COMMAND curl -k -u 'r@julatec.name:bvB}KyJaIF_6' -T ${CMAKE_CURRENT_BINARY_DIR}/tms2R_0.1.0.tar.gz ftp://julatec.name/src/contrib/)
120|add_dependencies(tms2R-deploy tms2R)
121|find_library(TMS2_INDEX NAMES tms2index PATHS ${CMAKE_INSTALL_PREFIX})
122|if (TMS2_INDEX)
123|     message("Found libtms2index previously installed on ${TMS2_INDEX}")
124|     target_link_libraries(tms2 tms2index)
125|else ()
126|     message("Execute: make -C ${CMAKE_CURRENT_BINARY_DIR} tms2indexgen && ${CMAKE_CURRENT_BINARY_DIR}/tms2indexgen | /bin/bash")
127|     add_custom_target(tms2index-local
128|             COMMAND tms2indexgen --tms2R ${CMAKE_SOURCE_DIR}/src/tms2R --maximum ${
```

```
    |                tms2index_maximum} --window ${tms2index_window} --resolution ${
    |                tms2index_resolution}
129 |            DEPENDS tms2indexgen)
130 |    # TODO: Enable only when for release.
131 |    add_dependencies(tms2R tms2index-local)
132 |    add_dependencies(tms2 tms2index-local)
133 |    target_link_libraries(tms2 tms2index)
134 |    include_directories("${CMAKE_CURRENT_BINARY_DIR}/include")
135 |    link_directories("${CMAKE_CURRENT_BINARY_DIR}/lib")
136 |endif ()
137 |# ------------------------------------------------------------------------------
138 |# ADDITIONAL TARGERS
139 |# ------------------------------------------------------------------------------
140 |# Uninstall target
141 |if (NOT TARGET uninstall)
142 |    configure_file(
143 |            "${CMAKE_CURRENT_SOURCE_DIR}/CMake/cmake_uninstall.cmake.in"
144 |            "${CMAKE_CURRENT_BINARY_DIR}/CMake/cmake_uninstall.cmake"
145 |            IMMEDIATE @ONLY)
146 |    add_custom_target(uninstall
147 |            COMMAND ${CMAKE_COMMAND} -P ${CMAKE_CURRENT_BINARY_DIR}/CMake/
    |                cmake_uninstall.cmake)
148 |endif ()
```

# 6.2. Search Prototype Source Code

**Listing 6.8:** `bitree.h`: Tensor implementation as a fixed height binary tree

```
1 |//
```

## 6.2.1. Tensor implementation

```
2 |#ifndef TMS2_BITREE_H
3 |#define TMS2_BITREE_H
4 |namespace tms2 {
5 |inline namespace search {
6 |// This boolean binary tree allows to keep track of the visited frontier.
7 |struct bitree {
8 |public:
9 |  // Default constructor with the height of the tree.
10|  // Do not stores unnecessary memory.
11|  inline bitree(::std::size_t height) noexcept :
12|      height_(height),
13|      b1_(false),
14|      b2_(false),
15|      child1_(nullptr),
16|      child2_(nullptr) {}
17|  // Frees the memory only if it was allocated before.
18|  inline ~bitree() {
19|    if (child1_) delete child1_;
20|    if (child2_) delete child2_;
21|  }
22|  // Constant time (height) access for the value encoded in the given key.
23|  inline bool &operator[](const bool *const k) {
24|    if (height_ == 1) return *k ? b1_ : b2_;
25|    if (*k) { if (!child1_) child1_ = new bitree(height_ - 1); }
26|    else { if (!child2_) child2_ = new bitree(height_ - 1); }
27|    return (*k ? child1_ : child2_)->operator[](k + 1);
28|  }
29|private:
30|  // Avoid bad behavior.
31|  bitree() = delete;
32|  // Avoid bad behavior.
33|  bitree(const bitree &) = delete;
34|  // Avoid bad behavior.
35|  bitree &operator=(const bitree &) = delete;
36|  // Height of the tree
37|  const ::std::size_t height_;
38|  // Values stored.
39|  bool b1_;
40|  bool b2_;
41|  // Left child.
42|  bitree *child1_;
43|  // Right child.
44|  bitree *child2_;
45|};
46|}
47|}
48|#endif //TMS2_BITREE_H
```

**Listing 6.9:** `spectrum.h`: Spectrum header

```
1 //
```

## 2 6.2.2. Spectrum header

```
3  #ifndef __TMS2_SPECTRUM_H__
4  #define __TMS2_SPECTRUM_H__
5  #include <cmath>
6  #include <ostream>
7  #include <map>
8  #include <queue>
9  #include <sstream>
10 #include <vector>
11 #include <tms2/da.h>
12 #include <tms2/bitree.h>
13 namespace tms2 {
14 inline namespace search {
15 constexpr ida H = 001.0079_da;
16 constexpr ida C = 012.0107_da;
17 constexpr ida N = 014.0067_da;
18 constexpr ida O = 015.9994_da;
19 constexpr ida H2O = H * 2 + O;
20 constexpr ida NH2 = N + H * 2;
21 constexpr ida NH3 = N + H * 3;
22 typedef enum {
23   tU,
24   tN,
25   tC
26 } terminal_t;
27 constexpr terminal_t operator~(terminal_t terminal) {
28   return
29       terminal == tN ? tC :
30       terminal == tC ? tN :
31       tU;
32 }
33 constexpr ida terminal_masses[][2] = {
34     {0.0_da, 0.0_da},
35     {+(H),   +(H2O)},
36     {0.0_da, +(H2O + H)}
37 };
38 // Ion types
39 typedef enum : unsigned char {
40   u,
41   a,
42   as,
43   ao,
44   b,
45   bs,
46   bo,
47   c,
48   d,
49   v,
50   w,
51   x,
52   y,
53   ys,
54   yo,
55   z,
56   _last,
57   _nbegin = a,
58   _nend = d,
59   _cbegin = x,
60   _cend = _last
61 } ion_t;
62 typedef struct {
63   const terminal_t t;
64   const ion_t i;
65   const ida mass;
66   const char c;
```

```
67 |   const char c2;
68 | } ion;
69 | constexpr ion u_ion = {tU, u, 0, '?', 0};
70 | constexpr ion a_ion = {tN, a, -(C + O + H), 'a', 0};
71 | constexpr ion as_ion = {tN, as, a_ion.mass - (NH3), 'a', '*'};
72 | constexpr ion ao_ion = {tN, ao, a_ion.mass - (H2O), 'a', 'o'};
73 | constexpr ion b_ion = {tN, b, -(H), 'b', 0};
74 | constexpr ion bs_ion = {tN, bs, b_ion.mass - (NH3), 'b', '*'};
75 | constexpr ion bo_ion = {tN, bo, b_ion.mass - (H2O), 'b', 'o'};
76 | constexpr ion c_ion = {tN, c, +(NH2), 'c', 0};
77 | constexpr ion d_ion = {tN, d, 0, 'd', 0};
78 | constexpr ion v_ion = {tC, v, 0, 'v', 0};
79 | constexpr ion w_ion = {tC, w, 0, 'w', 0};
80 | constexpr ion x_ion = {tC, x, +(C + O - H), 'x', 0};
81 | constexpr ion y_ion = {tC, y, +(H), 'y', 0};
82 | constexpr ion ys_ion = {tC, ys, y_ion.mass - (NH3), 'y', '*'};
83 | constexpr ion yo_ion = {tC, yo, y_ion.mass - (H2O), 'y', 'o'};
84 | constexpr ion z_ion = {tC, z, +(NH2), 'z', 0};
85 | inline ion_t &operator++(ion_t &i) {
86 |   unsigned char c = i;
87 |   return i = static_cast<ion_t>(++c);
88 | }
89 | constexpr ion ions_list[] = {
90 |     u_ion,
91 |     a_ion,
92 |     as_ion,
93 |     ao_ion,
94 |     b_ion,
95 |     bs_ion,
96 |     bo_ion,
97 |     c_ion,
98 |     d_ion,
99 |     v_ion,
100|     w_ion,
101|     x_ion,
102|     y_ion,
103|     ys_ion,
104|     yo_ion,
105|     z_ion
106| };
107| inline const ion &operator*(ion_t i) {
108|   return ions_list[i];
109| }
110| constexpr struct {
111|   inline ion_t begin() const noexcept { return ion_t::_nbegin; }
112|   inline ion_t end() const noexcept { return ion_t::_nend; }
113| } n_ions = {};
114| constexpr struct {
115|   inline ion_t begin() const noexcept { return ion_t::_cbegin; }
116|   inline ion_t end() const noexcept { return ion_t::_cend; }
117| } c_ions = {};
118| constexpr ida cord(ida W, ion_t i, ::std::size_t n) {
119|   return
120|       (n == 0 && ions_list[i].t == tN) ? 0 :
121|       (n == 0 && ions_list[i].t == tC) ? W - (H * 2 + O) :
122|       (ions_list[i].t == tN) ? W + ions_list[i].mass :
123|       (ions_list[i].t == tC) ? W + ions_list[i].mass :
124|       W;
125| }
126| typedef fda ferr;
127| typedef ida zerr;
128| typedef float fit;
129| typedef int zit;
130| // Print function for ion types
131| template<typename charT>
132| inline ::std::basic_ostream<charT> &operator<<(
133|     ::std::basic_ostream<charT> &out, ion_t ion) {
134|   const search::ion &i = *ion;
135|   out << i.c;
136|   if (i.c2) out << '^' << i.c2;
```

```
137 |   return out;
138 | }
139 | // Spectrum class definition.
140 | class spectrum {
141 | public:
142 |   // Peaks are implemented as a sorted set of peaks.
143 |   // See \fullref{}.
144 |   typedef ::std::map<ida, fit> peaks_t;
145 |   typedef typename peaks_t::value_type peak_t;
146 |   // The keys of spectrum. See \fullref{}.
147 |   typedef bool *key_t;
148 |   bool contains(const ida peak, tms2::ida &observed) const;
149 | private:
150 |   // Peaks are implemented as a sorted set of peaks
151 |   const peaks_t peaks_;
152 |   const ida precursorMZ_;
153 |   const zerr __err__;
154 |   // Sum of the square of errors.
155 |   const zerr __serr__;
156 |   // Sum of the square of errors.
157 |   const zerr __sserr__;
158 |   // Estimation of the ion type.
159 |   const terminal_t terminal_;
160 |   // Sum of the intensities
161 |   const fit __sum_it__;
162 |   // Max intensity
163 |   const fit __max_it__;
164 |   // Possible permutations
165 |   const ::std::size_t __permutations__;
166 |   const bool __admissible__;
167 |   // Estimation of the current peak. Stored to avoid recalculations.
168 |   const double __estimation__;
169 | private:
170 |   inline spectrum(
171 |       const peaks_t &_peaks_,
172 |       const ida _precursorMZ_,
173 |       const zerr _err_,
174 |       const zerr _serr_,
175 |       const zerr _sserr_,
176 |       const terminal_t terminal_,
177 |       const fit _sum_it_,
178 |       const fit _max_it_,
179 |       const size_t _permutations_,
180 |       const bool _admissible_,
181 |       const double _estimation_) :
182 |       peaks_(_peaks_),
183 |       precursorMZ_(_precursorMZ_),
184 |       __err__(_err_),
185 |       __serr__(_serr_),
186 |       __sserr__(_sserr_),
187 |       terminal_(terminal_),
188 |       __sum_it__(_sum_it_),
189 |       __max_it__(_max_it_),
190 |       __permutations__(_permutations_),
191 |       __admissible__(_admissible_),
192 |       __estimation__(_estimation_) {}
193 | public:
194 |   // Constructor that takes initialization from a set of peaks.
195 |   spectrum(const peaks_t &peaks, ida precursorMZ, terminal_t terminal, ida err);
196 |   // Empty constructor for a spectrum.
197 |   inline spectrum() noexcept :
198 |       peaks_(),
199 |       precursorMZ_(H2O + H),
200 |       __err__(0),
201 |       __serr__(0.0),
202 |       __sserr__(0.0),
203 |       terminal_(tU),
204 |       __sum_it__(0.0),
205 |       __max_it__(0.0),
206 |       __permutations__(1),
```

```
207 |       __admissible__(true),
208 |       __estimation__(::std::numeric_limits<double>::infinity()) {
209 |     // Nothing to do here.
210 |   }
211 |   // Copy constructor from another peak.
212 |   inline spectrum(const spectrum &other) = default;
213 |   // Assignment operator from another peak.
214 |   inline spectrum &operator=(const spectrum &other) noexcept {
215 |     this->~spectrum();
216 |     return *new(this) spectrum(other);
217 |   }
218 |   // Assignment operator from another peak.
219 |   inline spectrum &operator|=(const spectrum &other) noexcept {
220 |     peaks_t peaks = peaks_;
221 |     for (const auto &p : other) peaks[p.first] = p.second;
222 |     return *new(this) spectrum(peaks, precursorMZ_, terminal_, err());
223 |   }
224 |   // Tests the admissibility of this spectrum for the default admissibleerror.
225 |   inline operator bool() const noexcept { return __admissible__; }
226 |   // Peaks are implemented as a sorted set of peaks
227 |   inline double estimation() const noexcept { return __estimation__; }
228 |   // Gets the pre-calculated sum of squares of errors.
229 |   inline zerr sserr() const noexcept { return __sserr__; }
230 |   // Gets the pre-calculated sum of squares of errors.
231 |   inline zerr serr() const noexcept { return __serr__; }
232 |   // Gets the pre-calculated ion type.
233 |   inline terminal_t terminal() const noexcept { return terminal_; }
234 |   // Gets the pre-calculated sum of intensities.
235 |   inline fit intensity() const noexcept { return __sum_it__; }
236 |   // Gets the pre-calculated max of intensities.
237 |   inline fit max_it() const { return __max_it__; }
238 |   // Gets the minimum peak of this spectrum.
239 |   inline ida min() const { return peaks_.begin()->first; }
240 |   // Gets the maximum peak of this spectrum.
241 |   inline ida max() const { return peaks_.rbegin()->first; }
242 |   // Points to the beginning of the spectrum
243 |   inline peaks_t::const_iterator
244 |   begin() const noexcept { return peaks_.cbegin(); }
245 |   // Points to the end of the spectrum.
246 |   inline peaks_t::const_iterator
247 |   end() const noexcept { return peaks_.cend(); }
248 |   // Points to the beginning of the spectrum
249 |   inline peaks_t::const_reverse_iterator
250 |   rbegin() const noexcept { return peaks_.crbegin(); }
251 |   // Points to the end of the spectrum.
252 |   inline peaks_t::const_reverse_iterator
253 |   rend() const noexcept { return peaks_.crend(); }
254 |   // Gets the size of the current spectrum.
255 |   inline ::std::size_t size() const { return peaks_.size(); }
256 |   // Searches for the best fit for a given distance searching at the maximumadmissible error given.
257 |   static inline ::std::size_t search_best(
258 |       const ida p,
259 |       const ida err,
260 |       da &best,
261 |       ida &error) noexcept {
262 |     const da center = p;
263 |     ::std::size_t result;
264 |     if (p >= da::imax()) {
265 |       error = p;
266 |       best = center;
267 |       result = ::std::numeric_limits<::std::size_t>::max() / 10000 + p;
268 |     } else {
269 |       best = center[err];
270 |       error = best.ivalue - p;
271 |       result = best.tpermutations;
272 |     }
273 |     //::std::cout << "best(" << center << "," << error << ") = " << best
274 |     //           << "/" << result << std::endl;
275 |     return result;
```

```
276 |   }
277 |   // Gets an spectrum that is encoded with the given key.
278 |   inline spectrum operator()(key_t k, terminal_t terminal) const noexcept {
279 |     peaks_t peaks;
280 |     for (const peak_t &pair : peaks_)
281 |       if (*k++) peaks[pair.first] = pair.second;
282 |     return spectrum(peaks, precursorMZ_, terminal, __err__);
283 |   }
284 |   // Stores the result of the key of a given to the given target key.
285 |   inline void
286 |   operator()(const spectrum &other, key_t const target) const {
287 |     auto o_it = other.peaks_.cbegin();
288 |     auto s_it = peaks_.cbegin();
289 |     const auto o_end = other.peaks_.cend();
290 |     const ::std::size_t size = peaks_.size();
291 |     for (int i = 0; i < size; i++) {
292 |       if (o_it != o_end) {
293 |         if (*s_it == *o_it) {
294 |           target[i] = true;
295 |           o_it++;
296 |         } else {
297 |           target[i] = false;
298 |         }
299 |         s_it++;
300 |       } else {
301 |         target[i] = false;
302 |       }
303 |     }
304 |   }
305 |   // Stores the result of combining this spectrum with the given
306 |   // key to the given target.
307 |   inline void operator()(
308 |       const spectrum &other,
309 |       const key_t merge,
310 |       key_t const target) const {
311 |     auto o_it = other.peaks_.cbegin();
312 |     auto s_it = peaks_.cbegin();
313 |     const auto o_end = other.peaks_.cend();
314 |     const ::std::size_t size = peaks_.size();
315 |     for (int i = 0; i < size; i++) {
316 |       if (o_it != o_end) {
317 |         if (*s_it == *o_it) {
318 |           target[i] = true;
319 |           o_it++;
320 |         } else {
321 |           target[i] = merge[i];
322 |         }
323 |         s_it++;
324 |       } else {
325 |         target[i] = merge[i];
326 |       }
327 |     }
328 |   }
329 |   // Prints the current spectrum to R list format.
330 |   ::std::pair<ida, ::std::string> explanation_c(
331 |       ida mz,
332 |       ida &last,
333 |       const int column) const {
334 |     ida center = mz - last;
335 |     last = mz;
336 |     da best;
337 |     ida error;
338 |     ::std::stringstream out;
339 |     if (center > da::imax()) {
340 |       out << "expression({";
341 |       out << "group('[', '";
342 |       out << da::scale(center);
343 |       out << "', ']')";
344 |       out << "})";
345 |       return {center, out.str()};
```

```
346     }
347     out << "expression({";
348     switch (search_best(center, __err__, best, error)) {
349       case 0:
350         out << "group('X', '";
351         out << da::scale(center);
352         out << "', 'X')";
353         break;
354       case 1:
355         out << best.lcombinations[0].vector;
356         break;
357       default:
358         switch (best.tcombinations) {
359           case 1:
360             //out << "group('[', list('" << da::scale(center) << '+' << da::scale(
                    error) << "', ";
361             out << "group('[', list(";
362             out << best.lcombinations[0].vector;
363             out << "), ']')";
364             break;
365           case 2:
366             //out << "group('[', list('" << da::scale(center) << '+' << da::scale(
                    error) << "', ";
367             out << "group('[', list(";
368             out << best.lcombinations[0].vector << ", ";
369             out << best.lcombinations[1].vector
370                 << "), ']')";
371             break;
372           default:
373             //out << "group('[', list('" << da::scale(center) << '+' << da::scale(
                    error) << "', ";
374             out << "group('[', list(";
375             out << best.lcombinations[0].vector << ", ";
376             out << '.' << '.' << '.' << ',';
377             out << best.lcombinations[1].vector;
378             out << "), ']')";
379             break;
380         }
381         break;
382     }
383     out << "})";
384     return {center, out.str()};
385   }
386   // Prints the current spectrum to R list format.
387   void
388   explanation_c(
389       ::std::vector<::std::pair<ida, ::std::string>> &out) const {
390     ida last = terminal_masses[tN][0];
391     int column = 0;
392     for (const auto &peak : peaks_) {
393       out.push_back(explanation_c(peak.first, last, column++));
394     }
395   }
396   // Prints the current spectrum in R list format.
397   template<typename charT>
398   friend ::std::basic_ostream<charT> &
399   operator<<(::std::basic_ostream<charT> &out, const spectrum &set) {
400     int index = 0;
401     out << 'c' << '(';
402     for (const auto &pair : set.peaks_) {
403       if (index++) out << ',' << ' ';
404       out << ::std::fixed << ::std::setprecision(2)
405           << (da::scale(pair.first));
406     }
407     out << ')';
408     return out;
409   }
410   // Defines a total order relation for spectra.
411   friend inline bool operator<(const spectrum &a, const spectrum &b) {
412     const zerr err = ::std::max(a.__err__, b.__err__);
413     auto a_beg = a.peaks_.rbegin();
```

```
414 |     const auto a_end = a.peaks_.rend();
415 |     auto b_beg = b.peaks_.rbegin();
416 |     const auto b_end = b.peaks_.rend();
417 |     while (a_beg != a_end || b_beg != b_end) {
418 |       if (!(a_beg != a_end)) return true;
419 |       if (!(b_beg != b_end)) return false;
420 |       ida av = a_beg->first;
421 |       ida bv = b_beg->first;
422 |       if (::std::abs(av - bv) < err) {
423 |         ++a_beg;
424 |         ++b_beg;
425 |       } else {
426 |         return av < bv;
427 |       }
428 |     }
429 |     return false;
430 |   }
431 |   // Tests if two spectra are the same.
432 |   friend inline bool
433 |   operator==(const spectrum &a, const spectrum &b) {
434 |     return !(a < b || b < a);
435 |   }
436 |   // Tests if two spectra are the different.
437 |   friend inline bool
438 |   operator!=(const spectrum &a, const spectrum &b) {
439 |     return !(a == b);
440 |   }
441 |   // Calculates the total number of permutations of this peaks.
442 |   inline ::std::size_t
443 |   tpermutations() const noexcept { return __permutations__; }
444 |   inline ferr err() const noexcept { return da::scale(__err__); }
445 |   spectrum get_other_ion(const spectrum &best) const;
446 | public:
447 |   ida toNTerminal(ida peak) const {
448 |     return precursorMZ_ - (peak + terminal_masses[tC][1]) +
449 |            terminal_masses[tN][0];
450 |   }
451 |   ida toCTerminal(ida peak) const {
452 |     return precursorMZ_ - (peak - terminal_masses[tN][0]) -
453 |            terminal_masses[tC][1];
454 |   }
455 |   static spectrum merge(
456 |       const spectrum &tNSpectrum,
457 |       const spectrum &tCSpectrum) {
458 |     peaks_t peaks = tNSpectrum.peaks_;
459 |     for (const auto &p : tCSpectrum.peaks_) {
460 |       const ida peak = tNSpectrum.toNTerminal(p.first);
461 |       if (peak && p.second &&
462 |           peak < tCSpectrum.precursorMZ_ - terminal_masses[tC][1]) {
463 |         peaks[tNSpectrum.toNTerminal(p.first)] = p.second;
464 |       }
465 |     }
466 |     return spectrum(peaks, tNSpectrum.precursorMZ_, tN, tNSpectrum.__err__);
467 |   }
468 | private:
469 |   static void __process__(
470 |       const peaks_t &p,
471 |       const ida precursorMZ,
472 |       const zerr &e,
473 |       double &serr,
474 |       double &sserr,
475 |       terminal_t &terminal,
476 |       fit &sit,
477 |       fit &mit,
478 |       ::std::size_t &per,
479 |       bool &adm,
480 |       double &est) noexcept;
481 | };
482 | // Solutions are stored using different order.
483 | struct by_error {
```

```
484 |   // The order takes in consideration size, estimation, sserrors.
485 |   inline bool operator()(const spectrum &a, const spectrum &b) {
486 |     if (a.intensity() < b.intensity()) return true;
487 |     if (a.intensity() > b.intensity()) return false;
488 |     if (a.size() < b.size()) return true;
489 |     if (a.size() > b.size()) return false;
490 |     if (a.estimation() < b.estimation()) return true;
491 |     if (a.estimation() > b.estimation()) return false;
492 |     if (a.sserr() < b.sserr()) return true;
493 |     if (a.sserr() > b.sserr()) return false;
494 |     return false;
495 |   }
496 | };
497 | class interpretation {
498 | public:
499 |   const spectrum full;
500 |   const spectrum tNSpectrum;
501 |   const spectrum tCSpectrum;
502 |   const ::std::size_t size;
503 |   const ::std::size_t ssize;
504 |   const ferr serr;
505 |   const ferr sserr;
506 |   const fit sum_it;
507 |   const double estimation;
508 |   interpretation(
509 |       const spectrum &tNSpectrum_,
510 |       const spectrum &tCSpectrum_) noexcept :
511 |       full(spectrum::merge(tNSpectrum_, tCSpectrum_)),
512 |       tNSpectrum(tNSpectrum_),
513 |       tCSpectrum(tCSpectrum_),
514 |       size(full.size()),
515 |       ssize(tNSpectrum.size() * tCSpectrum.size()),
516 |       serr(tNSpectrum.serr() + tCSpectrum.serr()),
517 |       sserr(tNSpectrum.sserr() + tCSpectrum.sserr()),
518 |       sum_it(full.intensity()),
519 |       estimation(tNSpectrum.estimation() * tCSpectrum.estimation()) {
520 |   }
521 |   interpretation(
522 |       const spectrum &full_,
523 |       const spectrum &tNSpectrum_,
524 |       const spectrum &tCSpectrum_,
525 |       const size_t size_,
526 |       const size_t ssize_,
527 |       const double serr_,
528 |       const double sserr_,
529 |       const double sum_it_,
530 |       const double estimation_) noexcept :
531 |       full(full_),
532 |       tNSpectrum(tNSpectrum_),
533 |       tCSpectrum(tCSpectrum_),
534 |       size(size_),
535 |       ssize(ssize_),
536 |       serr(serr_),
537 |       sserr(sserr_),
538 |       sum_it(sum_it_),
539 |       estimation(estimation_) {
540 |   }
541 |   interpretation(const interpretation &other) = default;
542 |   interpretation &operator=(const interpretation &other) noexcept {
543 |     this->~interpretation();
544 |     return *new(this) interpretation(other);
545 |   }
546 |   operator bool() const { return ::std::isinf(full.estimation()) == false; }
547 |   template<typename _CharT, typename _Traits = ::std::char_traits<_CharT> >
548 |   friend ::std::basic_ostream<_CharT, _Traits> &operator<<(
549 |       ::std::basic_ostream<_CharT, _Traits> &out,
550 |       const interpretation &interpretation) {
551 |     out << "F: " << interpretation.full << std::endl;
552 |     out << "Valid:" << ((bool) interpretation) << "Est: "
553 |         << interpretation.full.estimation() << std::endl;
```

```
554 |     out << "N: " << interpretation.tNSpectrum << std::endl;
555 |     out << "C: " << interpretation.tCSpectrum << std::endl;
556 |     out << std::endl;
557 |     return out;
558 |   }
559 | };
560 | // Solutions are stored using different order.
561 | struct by_estimation {
562 |   // The order takes in consideration size, estimation, sserrors.
563 |   inline bool
564 |   operator()(const interpretation &a, const interpretation &b) {
565 |     if (a.ssize < b.ssize) return true;
566 |     if (a.ssize > b.ssize) return false;
567 |     if (a.size < b.size) return true;
568 |     if (a.size > b.size) return false;
569 |     if (a.estimation < b.estimation) return true;
570 |     if (a.estimation > b.estimation) return false;
571 |     if (a.sum_it < b.sum_it) return true;
572 |     if (a.sum_it > b.sum_it) return false;
573 |     if (a.sserr < b.sserr) return true;
574 |     if (a.sserr > b.sserr) return false;
575 |     return false;
576 |   }
577 | };
578 | class solution {
579 | public:
580 |   typedef std::priority_queue<
581 |       interpretation,
582 |       std::vector<interpretation>,
583 |       by_estimation> queue_by_estimation;
584 |   typedef std::priority_queue<
585 |       interpretation,
586 |       std::vector<interpretation>,
587 |       by_error> queue_by_error;
588 |   typedef bitree visited_t;
589 |   const ida precursorMZ;
590 |   const ida fragmentation_error;
591 |   const spectrum spectra;
592 |   const ::std::size_t size;
593 |   mutable queue_by_estimation solutions;
594 |   queue_by_estimation pending;
595 |   visited_t visited;
596 |   inline solution(
597 |       const spectrum::peaks_t &peaks,
598 |       ferr err,
599 |       fda _precursorMZ) :
600 |       precursorMZ(da::scale(_precursorMZ)),
601 |       fragmentation_error(da::scale(err)),
602 |       spectra(guards(peaks, precursorMZ), precursorMZ, tU, fragmentation_error),
603 |       size(spectra.size()),
604 |       visited(size) {}
605 |   inline ~solution() {}
606 |   ::std::size_t extend(spectrum::key_t tNKey, spectrum::key_t tCKey) {
607 |     ::std::size_t newSolutions = 0;
608 |     if (visited[tNKey] == false || visited[tCKey] == false) {
609 |       visited[tNKey] = true;
610 |       visited[tCKey] = true;
611 |       for (::std::size_t i = 2; i < size; i++) {
612 |         if (tNKey[i] == false) {
613 |           tNKey[i] = true;
614 |           if (visited[tNKey] == false) {
615 |             spectrum tNSpectrum = spectra(tNKey, tN);
616 |             spectrum tCSpectrum = spectra(tCKey, tC);
617 |             interpretation interpretation(tNSpectrum, tCSpectrum);
618 |             //std::cout << interpretation << std::endl;
619 |             if (interpretation) {
620 |               pending.push(interpretation);
621 |               newSolutions++;
622 |             }
623 |           }
```

```
624 |           tNKey[i] = false;
625 |         }
626 |         if (tCKey[i] == false) {
627 |           tCKey[i] = true;
628 |           if (visited[tCKey] == false) {
629 |             spectrum tNSpectrum = spectra(tNKey, tN);
630 |             spectrum tCSpectrum = spectra(tCKey, tC);
631 |             interpretation interpretation(tNSpectrum, tCSpectrum);
632 |             //::std::cout << interpretation << ::std::endl;
633 |             if (interpretation) {
634 |               pending.push(interpretation);
635 |               newSolutions++;
636 |             }
637 |           }
638 |           tCKey[i] = false;
639 |         }
640 |       }
641 |     }
642 |     return newSolutions;
643 |   }
644 |   static spectrum::peaks_t guards(
645 |       const spectrum::peaks_t &peaks,
646 |       ida precursorMZ) {
647 |     spectrum::peaks_t result(peaks);
648 |     result[terminal_masses[tN][0]] = 0;
649 |     result[precursorMZ - terminal_masses[tN][1]] = 0;
650 |     result[terminal_masses[tC][0]] = 0;
651 |     result[precursorMZ - terminal_masses[tC][1]] = 0;
652 |     return result;
653 |   }
654 |   template<typename P>
655 |   ::std::size_t find(
656 |       ::std::size_t iterations,
657 |       P &pb,
658 |       bool (*check_abort)()) {
659 |     {
660 |       spectrum::peaks_t tNPeaks;
661 |       tNPeaks[terminal_masses[tN][0]] = 0;
662 |       tNPeaks[precursorMZ - terminal_masses[tN][1]] = 0;
663 |       spectrum tNSpectrum(tNPeaks, precursorMZ, tN, fragmentation_error);
664 |       spectrum::peaks_t tCPeaks;
665 |       tCPeaks[terminal_masses[tC][0]] = 0;
666 |       tCPeaks[precursorMZ - terminal_masses[tC][1]] = 0;
667 |       spectrum tCSpectrum(tCPeaks, precursorMZ, tC, fragmentation_error);
668 |       interpretation empty(tNSpectrum, tCSpectrum);
669 |       pending.push(empty);
670 |     }
671 |     ::std::allocator<bool> allocator;
672 |     bool *tNKey = allocator.allocate(size);
673 |     bool *tCKey = allocator.allocate(size);
674 |     while (!pending.empty() && !check_abort()) {
675 |       if (iterations-- <= 0) break;
676 |       pb.increment(1);
677 |       const interpretation currentSolution = pending.top();
678 |       pending.pop();
679 |       spectra(currentSolution.tNSpectrum, tNKey);
680 |       spectra(currentSolution.tCSpectrum, tCKey);
681 |       // std::cout << "Current:" << currentSolution << std::endl;
682 |       if (visited[tNKey] && visited[tCKey]) continue;
683 |       if (extend(tNKey, tCKey) == 0) {
684 |         auto it = currentSolution.full.rbegin();
685 |         const ida bLast1 = it->first;
686 |         ++it;
687 |         const ida bLast2 = it->first;
688 |         da last;
689 |         ida err;
690 |         switch (currentSolution.full.search_best(
691 |             bLast1 - bLast2,
692 |             fragmentation_error,
```

```
693 |             last,
694 |             err)) {
695 |           case 0:
696 |             break;
697 |           case 1:
698 |             if (last.lcombinations[0].vector.R ||
699 |                 last.lcombinations[0].vector.K) {
700 |               solutions.push(currentSolution);
701 |             }
702 |           default:
703 |             for (::std::size_t i = 0; i < last.ccombinations; i++) {
704 |               if (last.lcombinations[i].vector.R ||
705 |                   last.lcombinations[i].vector.K) {
706 |                 solutions.push(currentSolution);
707 |                 break;
708 |               }
709 |             }
710 |             break;
711 |         }
712 |       }
713 |     }
714 |     ::std::size_t found = solutions.size();
715 |     for (int to_copy = 1000 - solutions.size();
716 |          to_copy--> 0 && !pending.empty();) {
717 |       solutions.push(pending.top());
718 |       pending.pop();
719 |     }
720 |     return found;
721 |   }
722 |   template<typename charT>
723 |   friend inline ::std::basic_ostream<charT> &
724 |   operator<<(::std::basic_ostream<charT> &out, const solution &s) {
725 |     out << 'n' << '=' << '\'' << '\'' << ::std::endl;
726 |     out << 'i' << '=' << 'c' << '(' << '0';
727 |     for (const spectrum::peak_t &peak : s.spectra)
728 |       out << ',' << ' ' << peak.second;
729 |     out << ')' << ::std::endl;
730 |     out << 'm' << '=' << 'c' << '(' << '0';
731 |     for (const spectrum::peak_t &peak : s.spectra)
732 |       out << ',' << ' ' << da::scale(peak.first);
733 |     out << ')' << ::std::endl;
734 |     const std::size_t limit = std::min((std::size_t) 10,
735 |                                        s.solutions.size());
736 |     out
737 |         << "plot(m, i, type='h', main=n, xlab='m/z', ylab='intensity')"
738 |         << ::std::endl;
739 |     out << "c=rainbow(" << limit << ")" << ::std::endl;
740 |     const float step = s.spectra.max_it() / (limit + 2);
741 |     for (int i = 0; i < limit; i += 2) {
742 |       s.print_explanation(out, s.solutions.top(), i + 1, step);
743 |       s.solutions.pop();
744 |     }
745 |     return out;
746 |   }
747 |   template<typename charT>
748 |   void print_explanation(
749 |       ::std::basic_ostream<charT> &out,
750 |       const interpretation &explanation,
751 |       const int index,
752 |       const float step) const {
753 |     out << "points(" << explanation.full << ", rep(";
754 |     out << spectra.max_it() - (index + 0) * step << ", ";
755 |     out << explanation.full.size() << "), ";
756 |     out << "type = 'p', col='gray', cex=2.0, pch='|')"
757 |         << ::std::endl;
758 |     out << "points(" << explanation.tNSpectrum << ", rep(";
759 |     out << spectra.max_it() - (index - 1.0) * step << ", ";
760 |     out << explanation.tNSpectrum.size() << "), ";
761 |     out << "type = 'p', col='blue', cex=.7, pch='b')"
762 |         << ::std::endl;
763 |     out << "points(" << explanation.tCSpectrum << ", rep(";
```

```
764 |     out << spectra.max_it() - (index + 1.0) * step << ", ";
765 |     out << explanation.tCSpectrum.size() << "), ";
766 |     out << "type = 'p', col='red', cex=.7, pch='y')"
767 |         << ::std::endl;
768 |     ::std::vector<::std::pair<ida, ::std::string>> texts;
769 |     explanation.full.explanation_c(texts);
770 |     int column = 0;
771 |     {
772 |       const ida first = explanation.full.min();
773 |       ida curr = 0;
774 |       ida last = 0;
775 |       out << "text(c(";
776 |       for (auto it : explanation.full) {
777 |         const ida mz = it.first;
778 |         if (column++) {
779 |           ida dist = ::std::abs(last - mz);
780 |           ida middle = curr + dist / 2;
781 |           curr += dist;
782 |           out << ", ";
783 |           out << da::scale(middle);
784 |         } else {
785 |           ida middle = first / 2;
786 |           curr += first;
787 |           out << da::scale(middle);
788 |         }
789 |         last = mz;
790 |         if (curr > spectra.max()) {
791 |           break;
792 |         }
793 |       }
794 |       out << "), rep(" << spectra.max_it() - index * step << ", ";
795 |       out << column << "), c(";
796 |       for (int i = 0; i < column; i++) {
797 |         if (i) out << ", ";
798 |         out << texts[i].second;
799 |       }
800 |       out << "), cex=1.0)" << ::std::endl;
801 |     }
802 |     {
803 |       out << "text(" << explanation.full << ", ";
804 |       out << "rep(" << spectra.max_it() - (index - 0.60) * step << ", ";
805 |       out << explanation.full.size() << "), c(";
806 |       int i = 0;
807 |       for (auto it : explanation.full) {
808 |         if (i++) {
809 |           out << ", ";
810 |         }
811 |         ida peak;
812 |         if (explanation.tNSpectrum.contains(it.first, peak)) {
813 |           out << "'" << ::std::setprecision(1) << da::scale(peak) << "'";
814 |         } else {
815 |           out << "'(" << ::std::setprecision(1) << da::scale(it.first) << ")'";
816 |         }
817 |       }
818 |       out << "), cex=0.6)" << ::std::endl;
819 |     }
820 |     {
821 |       out << "text(" << explanation.full << ", ";
822 |       out << "rep(" << spectra.max_it() - (index + 0.60) * step << ", ";
823 |       out << explanation.full.size() << "), c(";
824 |       int i = 0;
825 |       for (auto it : explanation.full) {
826 |         if (i++) {
827 |           out << ", ";
828 |         }
829 |         ida peak;
830 |         if (explanation.tCSpectrum.contains(
831 |             explanation.tCSpectrum.toCTerminal(it.first), peak)) {
832 |           out << "'" << ::std::setprecision(1) << da::scale(peak) << "'";
833 |         } else {
```

```
834 |           out << "'(" << ::std::setprecision(1) << da::scale(explanation.tCSpectrum
    |               .toCTerminal(it.first)) << ")'";
835 |         }
836 |       }
837 |       out << "), cex=0.6)" << ::std::endl;
838 |     }
839 |   }
840 |};
841 |}
842 |}
843 |#endif //__TMS2_SPECTRUM_H__
```

**Listing 6.10:** `spectrum.cpp`: Spectrum source

```
1 //
```

## 2 6.2.3. Spectrum source

```
3  #include <tms2/da.h>
4  #include <tms2/spectrum.h>
5  // Calculates the estimation of this spectrum.
6  void tms2::search::spectrum::__process__(
7      const peaks_t &p,
8      const ida precursorMZ,
9      const zerr &e,
10     double &serr,
11     double &sserr,
12     terminal_t &terminal,
13     fit &sit,
14     fit &mit,
15     ::std::size_t &per,
16     bool &adm,
17     double &est) noexcept {
18   // Default values if there are no peaks.
19   serr = 0;
20   sserr = 0;
21   sit = 0.0;
22   mit = 0.0;
23   per = 0;
24   adm = false;
25   est = ::std::numeric_limits<double>::infinity();
26   if (p.empty()) return;
27   est = 0;
28   adm = true;
29   //zerr err = spectrum::estimation_ion(e, p.begin()->first, terminal);
30   zerr err = 0.0;
31   sit += ::std::log(p.begin()->second);
32   mit = ::std::max(mit, (fit) p.begin()->second);
33   if (p.size() > 1) {
34     da best;
35     ida last = 0;
36     int index = 0;
37     for (const peak_t &pk : p) {
38       if (index++) {
39         sit += ::std::log(pk.second);
40         mit = ::std::max(mit, (fit) pk.second);
41         if (adm) {
42           ::std::size_t ambiguity = spectrum::search_best(pk.first - last, e, best,
                 err);
43           serr += err;
44           sserr += ::std::pow(err, 2);
45           switch (ambiguity) {
46             case 0:
47               adm = false;
48               serr = ::std::numeric_limits<double>::infinity();
49               sserr = ::std::numeric_limits<double>::infinity();
50               break;
51             case 1:
52               est += ::std::log(ambiguity);
53               break;
54             default:
55               est += ::std::log(ambiguity);
56               break;
57           }
58         }
59       }
60       last = pk.first;
61     }
62   }
63   if (adm) est /= ::std::pow(2, p.size());
```

```
64 |   else est = ::std::numeric_limits<double>::infinity();
65 | }
66 | tms2::search::spectrum::spectrum(
67 |     const peaks_t &p,
68 |     ida precursorMZ,
69 |     terminal_t terminal,
70 |     ida e) :
71 |     peaks_(),
72 |     precursorMZ_(precursorMZ),
73 |     __err__(0),
74 |     __serr__(0.0),
75 |     __sserr__(0.0),
76 |     terminal_(tU),
77 |     __sum_it__(0.0),
78 |     __max_it__(0.0),
79 |     __permutations__(1),
80 |     __admissible__(true),
81 |     __estimation__(::std::numeric_limits<double>::infinity()) {
82 |   const zerr err = e;
83 |   double serr, sserr, est;
84 |   bool adm;
85 |   fit sit, mit;
86 |   ::std::size_t perm;
87 |   __process__(p, precursorMZ, err, serr, sserr, terminal, sit, mit, perm, adm, est)
   |       ;
88 |   new(this) spectrum(p, precursorMZ, err, serr, sserr, terminal, sit, mit, perm,
   |       adm, est);
89 | }
90 | bool tms2::search::spectrum::contains(const ida peak, tms2::ida &observed) const {
91 |   if (peaks_.find(peak) != peaks_.end()) {
92 |     observed = peak;
93 |     return true;
94 |   }
95 |   {
96 |     auto it = peaks_.lower_bound(peak);
97 |     if (it != peaks_.begin() && ::std::abs(it->first - peak) <= __err__) {
98 |       observed = it->first;
99 |       return true;
100|     }
101|   }
102|   {
103|     auto it = peaks_.upper_bound(peak);
104|     if (it != peaks_.end() && ::std::abs(it->first - peak) <= __err__) {
105|       observed = it->first;
106|       return true;
107|     }
108|   }
109|   return false;
110| }
```

## 6.3. R extension package: tms2R

The current implementation does not use a Shared library, all the codes are placed in a single Static library. This decision was made to facilitate the deployment on Windows machines. But, for better usage of memory; it will be recommendable to use a Shared library. Alternatives to use parallel programming can be followed as Matloff suggests [91].

**Listing 6.11:** tms2R.cpp: R extension package

```
1 |//
```

### 6.3.1. R extension linkage

```
3 |#ifdef _OPENMP
4 |#include <omp.h>
5 |// [[Rcpp::plugins(openmp)]]
6 |#endif
7 |// [[Rcpp::depends(RcppProgress)]]
8 |// [[Rcpp::plugins("cpp11")]]
9 |#include <Rcpp.h>
10 |#include <R.h>
11 |#include <Rinternals.h>
12 |#include <progress.hpp>
13 |#include <tms2/da.h>
14 |#include <tms2/spectrum.h>
15 |#include "tms2R.h"
16 |using namespace Rcpp;
17 |using namespace tms2;
18 |bool get_peaks(
19 |        const NumericVector peaks,
20 |        const NumericVector its,
21 |        const ::std::size_t max_peaks,
22 |        spectrum::peaks_t &result,
23 |        fda &maxIt) {
24 |    maxIt = 0.0f;
25 |    const double res = 100.0;
26 |    ::std::map< ::std::tuple < zit, ida >, ida > it2mz;
27 |    ::std::map<ida, zit> mz2it;
28 |    ::std::map<ida, bool> mz2vv;
29 |    ::std::size_t size = std::min(peaks.size(), its.size());
30 |    for (::std::size_t row = 0; row < size; row++) {
31 |        const ida mz = da::scale(peaks[row]);
32 |        zit it = -res * its[row];
33 |        if (mz2it.find(mz) != mz2it.end()) {
34 |            it = ::std::min(it, mz2it[mz]);
35 |        }
36 |        const ::std::tuple <ida, ida> key = std::make_tuple(it, -mz);
37 |        it2mz[key] = mz;
38 |        mz2it[mz] = it;
39 |        mz2vv[mz] = false;
40 |        maxIt = ::std::max((fda) its[row], maxIt);
41 |    }
42 |    for (int i = 0; i < max_peaks && it2mz.empty() == false; i++) {
43 |        const auto first = *it2mz.begin();
44 |        it2mz.erase(first.first);
45 |        const ida mz = first.second;
46 |        if (mz2vv[mz]) {
```

```
47 |             i--;
48 |             continue;
49 |         } else {
50 |             const ::std::map<ida, zit>::const_iterator lower =
51 |                     mz2it.lower_bound(first.second - da::scale(5.0f));
52 |             const ::std::map<ida, zit>::const_iterator upper =
53 |                     mz2it.upper_bound(first.second + da::scale(5.0f));
54 |             for (::std::map<ida, zit>::const_iterator i = lower;
55 |                  i != upper;
56 |                  ++i) {
57 |                 const ida mz = i->first;
58 |                 const zit it = i->second;
59 |                 const ::std::tuple <zit, ida> key = std::make_tuple(-it, mz);
60 |                 it2mz.erase(key);
61 |                 mz2vv[mz] = true;
62 |             }
63 |             mz2vv[mz] = true;
64 |             result[mz] = mz2it[mz] / -res;
65 |         }
66 |     }
67 |     ::std::cerr << "Taken highest: " << result.size() << ::std::endl;
68 |     return true;
69 | }
70 | bool get_peaks(
71 |         NumericVector &peaks,
72 |         NumericVector &its,
73 |         const spectrum::peaks_t &from) {
74 |     int i = 0;
75 |     for (const auto &pair : from) {
76 |         peaks[i] = da::scale((ida) pair.first);
77 |         its[i] = pair.second;
78 |         i++;
79 |     }
80 |     return true;
81 | }
82 | // [[Rcpp::export]]
83 | List rcpp_interpret(
84 |         NumericVector pIn,
85 |         NumericVector iIn,
86 |         float fragmentationError,
87 |         ::std::size_t max_peaks,
88 |         ::std::size_t max_iterations,
89 |         ::std::size_t max_results,
90 |         float precursorMZ) {
91 |     spectrum::peaks_t peaks;
92 |     fda maxIt;
93 |     get_peaks(pIn, iIn, max_peaks, peaks, maxIt);
94 |     ::std::cerr << "Searching on " << max_iterations << " iterations:"
95 |                 << std::endl;
96 |     solution sol(peaks, fragmentationError, precursorMZ);
97 |     {
98 |         Progress progress(max_iterations, true);
99 |         sol.find(max_iterations, progress, &Progress::check_abort);
100 |     }
101 |     ::std::cerr << "Interpretations: " << sol.solutions.size() << ::std::endl;
102 |     ::std::stringstream ss;
103 |     const ::std::size_t limit = 3 * max_results;
104 |     const fda step = maxIt / (limit + 2);
105 |     for (int interpretation = 0;
106 |          sol.solutions.size() > 0 && interpretation < limit;
107 |          sol.solutions.pop(), interpretation += 3) {
108 |         sol.print_explanation(ss, sol.solutions.top(), interpretation + 1,
109 |                               step);
110 |     }
111 | #ifdef TRACE
112 |     ::std::cerr << ss.str() << ::std::endl;
113 | #endif
114 |     CharacterVector commands = CharacterVector::create(ss.str());
115 |     NumericVector y = NumericVector::create(0.0, 1.0);
116 |     NumericVector pOut(peaks.size());
```

```
117 |     NumericVector iOut(peaks.size());
118 |     get_peaks(pOut, iOut, peaks);
119 |     List z = List::create(pOut, iOut, commands);
120 |     return z;
121 |}
122 |// [[Rcpp::export]]
123 |NumericMatrix rcpp_get_index() {
124 |     NumericMatrix matrix(da::imax() + 1, 5);
125 |     Progress progress(da::imax(), true);
126 |     matrix(0, 1) = 1;
127 |     matrix(0, 2) = 1;
128 |     matrix(0, 3) = 1;
129 |     matrix(0, 4) = 1;
130 |#ifdef _OPENMP
131 |     if ( threads > 0 )
132 |         omp_set_num_threads( threads );
133 |     REprintf("Number of threads=%i\\n", omp_get_max_threads());
134 |#endif
135 |#pragma omp parallel
136 |     {
137 |#pragma omp parallel for schedule(dynamic)
138 |         for (::std::size_t i = 1; i <= da::imax(); i++) {
139 |         progress.increment(1);
140 |           da mw((ida) i);
141 |           matrix(i, 0) = mw.fvalue;
142 |           matrix(i, 1) = (float) mw.tcombinations;
143 |           matrix(i, 2) = (float) mw.tpermutations;
144 |           matrix(i, 3) = (float) mw.ccombinations;
145 |           matrix(i, 4) = (float) mw.cpermutations;
146 |       }
147 |     }
148 |     return matrix;
149 |}
```

**Listing 6.12:** `tms2.R`: R extension package

```
interpret <- function(
 pks,
 max_error = 0.25,
 max_peaks = 30,
 max_iterations = 50000,
 max_results = 5,
 precursorMZ = 0) {
   pks2 <- rbind(pks, 0:0)
   plot(pks2[,1], pks2[,2], type='h',col='gray',lwd=1, xlab = "m/z", ylab = "
       Intensity")
   #abline(v=precursorMZ - 19, col="blue")
   #abline(v=precursorMZ - 18, col="red")
   abline(v=precursorMZ, col="green")
   #abline(v=1, col="yellow")
   i = tms2R::rcpp_interpret(pks[,1], pks[,2], max_error, max_peaks, max_iterations
       , max_results, precursorMZ)
   par(new=TRUE)
   points(as.numeric(i[[1]]), as.numeric(i[[2]]),col='green',lwd=1)
   eval(parse(text=i[[3]]))
 }
show_index <- function() {
  i = tms2R::rcpp_get_index()
  i = i[i[,2] >= 1,]
  qplot(i[,1],i[,3], xlab = "Molecular weight", ylab = "Ambiguity", log="y", geom='
      bin2d', bins =200) +
    scale_fill_gradientn(colours=topo.colors(100), trans="log", breaks = 10^(0:2))
}
```

Chapter 7

# Acronyms

**ANOVA** Analysis of Variance . . . . . 66

Statistical model used to analyse the differences among group means and their associated procedures developed by statistician and evolutionary biologist Ronald Fisher [103; 104].

The design of experiments is the design of any task that aims to describe or explain the variation of information under conditions that are hypothesised to reflect the variation [103; 104].

**DNA** Deoxyribonucleic Acid . . . . . 2

Deoxyribonucleic acid is a thread-like chain of nucleotides carrying the genetic instructions used in the growth, development, functioning and reproduction of all known living organisms and many viruses [63].

**RNA** Ribonucleic Acid

Any of various nucleic acids that contain ribose and uracil as structural components and are associated with the control of cellular chemical activities

[64].

**fdr** False Discovery Rate . . . . . 56

The proportion of the results that are false positive.

**FFT** Fast Fourier Transforms . . . . . 59

It is an algorithm that computes the discrete Fourier transform of a sequence, or its inverse. Fourier analysis converts a signal from its original domain (often time or space) to a representation in the frequency domain and vice-versa.

**GPU** Graphics Processor Unit . . . . . 60

It is a specialised electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display.

**HG38** Human Genome Version 38 . . . . . 3

Human Genome version 38

**HiRIEF** High-resolution Isoelectric Focusing . . . . . XIX

High-resolution isoelectric focusing technique that fragments the sample by peptide isoelectric point prediction [15].

**l-system** L-system [86]

Lindenmayer system [86] An **L-system** or **Lindenmayer System** is a formal grammar (a set of rules and symbols) most famously used to model the growth processes of plant development, though able to shape the morphology of a variety of organisms. L-systems can also be used to generate self-similar fractals such as iterated function systems. L-systems were introduced and developed in 1968 by the Lindenmayer [86].

**LC-MS** Liquid Chromatography Mass Spectrometry . . . . . 22

It is an analytical chemistry technique that combines the physical separation

capabilities of liquid chromatography with the mass analysis capabilities of mass spectrometry [15].

**LSH** Locality Sensitive Hashing . . . . . 54

Technique that reduces the dimensional complexity of the data calculating small signatures [118].

**MS** Mass Spectrometry . . . . . 2

Mass spectrum measures the masses within a sample.

**MS/MS** Tandem Mass Spectrometry

Mass Spectrometry that involves multiple steps of mass spectrometry selection, with some form of fragmentation occurring in between the stages.

**mw** Molecular Weight . . . . . 8

Weight of a molecule measured in Daltons.

**NNS** Nearest Neighbour Search

It is an optimisation problem for finding closest or most similar points.

**PSM** Peptide Spectrum Match . . . . . 56

The number of peptide-spectrum matches. The number of PSM's is the total number of identified peptide spectra matched for the protein [116].

**pH** Hydrogenic Potential . . . . . 22

It is a numeric scale used to specify the acidity or basicity of an aqueous solution. It is approximately the negative of the base 10 logarithm of the molar concentration, measured in units of moles per litre, of hydrogen ions

**pI** Isolectric/Isoionic point . . . . . 22

It is the pH at which the amino acid does not migrate in an electric field [116].

**PTM** Post-Translational Modification . . . . . 15

A post-translational modification is a biochemical modification that occurs to one or more amino acids in protein after the ribosome has translated the protein. [17].

**HER2** Human Epidermal Growth Factor Receptor 2 . . . . . 2

HER2 is one such gene that can play a role in the development of breast cancer [120].

**XCorr** Cross Correlation . . . . . 58

It is a measure of the goodness of fit of experimental peptide fragments to theoretical spectra created from the sequence $b$ and $y$ ions. [116].

# References


[1] Memoization. URL https://en.wikipedia.org/wiki/Memoization.

[2] Overview of Post-Translational Modification. URL https://www.thermofisher.com/cr/en/home/life-science/protein-biology/protein-biology-learning-center/protein-biology-resource-library/pierce-protein-methods/overview-post-translational-modification.html.

[3] Standardization of heparins — salts of heparin. URL https://www.mlo-online.com/articles/200710/1007cover_story.pdf.

[4] l-CYSTINE. *Organic Syntheses*, 5:39, 1925. ISSN 00786209. doi: 10.15227/orgsyn.005.0039. URL http://orgsyn.org/demo.aspx?prep=CV1P0194.

[5] Bruce Alberts. *Molecular biology of the cell*. ISBN 0815345240.

[6] J Andersen and Matthias Mann. Functional genomics by mass spectrometry. *FEBS Letters*, 480(1):25–31, 2000. ISSN 0014-5793. URL http://reviews.bmn.com/journals/atoz/latest?uid=FEBS.bmn0003a_00145793_v0480i01_00017737.

[7] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *Proceedings of the 5th ACM-SIAM Sympos. Discrete Algorithms*, 1(212):573–582, 1994. ISSN 00045411. doi: 10.1145/293347.293348.

[8] Nuno Bandeira, Jesper V Olsen, Matthias Mann, and Pavel A Pevzner. Multi-spectra peptide sequencing and its applications to multistage mass spectrometry. 24:416–423, 2008. doi: 10.1093/bioinformatics/btn184. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2718660/pdf/btn184.pdf.

[9] Christian Bartels. Fast algorithm for peptide sequencing by mass spectroscopy. *Biological Mass Spectrometry*, 19(6):363–368, 6 1990. ISSN 1052-9306.

doi: 10.1002/bms.1200190607. URL http://doi.wiley.com/10.1002/bms.1200190607.

[10] Beman Dawes. Boost Filesystem Library Version 3, 2005. URL http://www.boost.org/doc/libs/1_66_0/libs/filesystem/doc/index.htm.

[11] Jeremy M Berg, John L Tymoczko, and Lubert Stryer. Vitamins Are Often Precursors to Coenzymes. 2002. URL https://www.ncbi.nlm.nih.gov/books/NBK22549/.

[12] K. Biemann, C. Cone, B. R. Webster, and G. P. Arsenault. Determination of the Amino Acid Sequence in Oligopeptides by Computer Interpretation of Their High-Resolution Mass Spectra. *Journal of the American Chemical Society*, 88 (23):5598–5606, 12 1966. doi: 10.1021/ja00975a045. URL http://pubs.acs.org/doi/abs/10.1021/ja00975a045.

[13] Ken Bill, Martin Hoffman, Andy Cedilnik, David Cole, Marcus Hanwell, Julien Jomier, Brad King, and Alex Neundorf. Mastering CMake Fifth Edition. URL www.cmake.org.

[14] Guy E Blelloch. Programming Parallel Algorithms. URL http://web.scandal.cs.cmu.edu/www/cacm.html.

[15] Rui M M Branca, Lukas M Orre, Henrik J Johansson, Viktor Granholm, Mikael Huss, Åsa Pérez-Bercoff, Jenny Forshed, Lukas Käll, and Janne Lehtiö. HiRIEF LC-MS enables deep proteome coverage and unbiased proteogenomics. *Nature Methods*, 11(1):59–62, 2014. ISSN 1548-7091. doi: 10.1038/nmeth.2732. URL http://www.nature.com/doifinder/10.1038/nmeth.2732.

[16] Breastcancer.org. HER2 Status | Breastcancer.org. URL http://www.breastcancer.org/symptoms/diagnosis/her2.

[17] Sydney. Brenner, Jeffrey H. Miller, and William (William J.) Broughton. *Encyclopedia of genetics*. Academic Press, 2002. ISBN 9780122270802. URL https://www.sciencedirect.com/science/referenceworks/9780122270802.

[18] Veit R. Buchholz and Michael Flossdorf. Chapter One – Single-Cell Resolution of T Cell Immune Responses. In *Advances in Immunology*, volume 137, pages 1–41. 2018. ISBN 9780128151891. doi: 10.1016/bs.ai.2017.12.001.

[19] Cambridge. quality of life Definition in the Cambridge English Dictionary, 2016. URL https://dictionary.cambridge.org/us/dictionary/english/amino-acid?a=britishhttp://dictionary.cambridge.org/us/dictionary/english/quality-of-life.

[20] Arthur Cayley. The Theory of Groups: Graphical Representation. *American Journal of Mathematics*, 1(2):174–176, 1878. URL http://www.jstor.org/stable/2369306http://www.jstor.org/stable/2369306.

[21] Professor Cayley. Desiderata and Suggestions: No. 2. The Theory of Groups: Graphical Representation. *American Journal of Mathematics*, 1(2):174, 1878. ISSN 00029327. doi: 10.2307/2369306. URL http://www.jstor.org/stable/2369306?origin=crossref.

[22] Matthew C Chambers, Brendan Maclean, Robert Burke, Dario Amodei, Daniel L Ruderman, Steffen Neumann, Laurent Gatto, Bernd Fischer, Brian Pratt, Jarrett Egertson, Katherine Hoff, Darren Kessner, Natalie Tasman, Nicholas Shulman, Barbara Frewen, Tahmina A Baker, Mi-Youn Brusniak, Christopher Paulse, David Creasy, Lisa Flashner, Kian Kani, Chris Moulding, Sean L Seymour, Lydia M Nuwaysir, Brent Lefebvre, Frank Kuhlmann, Joe Roark, Paape Rainer, Suckau Detlev, Tina Hemenway, Andreas Huhmer, James Langridge, Brian Connolly, Trey Chadick, Krisztina Holly, Josh Eckels, Eric W Deutsch, Robert L Moritz, Jonathan E Katz, David B Agus, Michael MacCoss, David L Tabb, and Parag Mallick. A cross-platform toolkit for mass spectrometry and proteomics. *Nature Biotechnology*, 30(10):918–920, 10 2012. ISSN 1087-0156. doi: 10.1038/nbt.2377. URL http://www.nature.com/articles/nbt.2377.

[23] Ting Chen, Ming-Yang Kao, Matthew Tepel, John Rush, and George M. Church. A Dynamic Programming Approach to De Novo Peptide Sequencing via Tandem Mass Spectrometry. *Journal of Computational Biology*, 8(3):325–337, 6 2001. ISSN 1066-5277. doi: 10.1089/10665270152530872. URL http://www.ncbi.nlm.nih.gov/pubmed/11535179http://www.liebertonline.com/doi/abs/10.1089/10665270152530872.

[24] David W Christianson##, Stefan0 Manganiv, Gil Shohamll, and William N Lipscomb#. THE JOURNAL OF BIOLOGICAL CHEMISTRY Binding of D-Phenylalanine and D-Tyrosine to Carboxypeptidase A*. 264(22):12849–12853, 1989. URL http://www.jbc.org/content/264/22/12849.full.pdf.

[25] Michael M. Cox, Jennifer A. Doudna, and Michael (Biochemist) O'Donnell. *Molecular biology : principles and practice*. ISBN 1464126143.

[26] Vlado Danick, Theresa A Addona, Karl R Clauser, James E Vath, and Pavel A Pevzner. De Novo Peptide Sequencing via Tandem Mass Spectrometry. *JOURNAL OF COMPUTATIONAL BIOLOGY*, 64(3):327–342, 1999. URL https://pdfs.semanticscholar.org/70ee/1c7b5bab9626db7e57451b5fbf176d33773a.pdf.

[27] Eugene E. Dekker. Data for Biochemical Research. *Journal of the American Chemical Society*, 82(19):5258–5258, 10 1960. ISSN 0002-7863. doi: 10.1021/ja01504a071. URL http://pubs.acs.org/doi/abs/10.1021/ja01504a071.

[28] Arun Devabhaktuni and Joshua E. Elias. Application of de Novo Sequencing to Large-Scale Complex Proteomics Data Sets. *Journal of Proteome Research*, 15(3):732–742, 3 2016. ISSN 1535-3893. doi: 10.1021/acs.jproteome.5b00861. URL http://pubs.acs.org/doi/10.1021/acs.jproteome.5b00861.

[29] J. Dieudonné and M. Landsberg. *Foundations of Modern Analysis.*, volume 41. WILEYfiVCH Verlag, 1961. doi: 10.1002/zamm.19610410413. URL http://doi.wiley.com/10.1002/zamm.19610410413.

[30] Jian Ding, Chengjiang Ruan, Ying Guan, and Priti Krishna. Identification of microRNAs involved in lipid biosynthesis and seed size in developing sea buckthorn seeds using high-throughput sequencing. *Scientific Reports*, 8(1): 4022, 12 2018. ISSN 2045-2322. doi: 10.1038/s41598-018-22464-w. URL http://www.nature.com/articles/s41598-018-22464-w.

[31] Gary Van Domselaar. Laboratory 2.4: Mascot. *Proteomics*, pages 1–17, 2004.

[32] Kevin M. Downard. Francis William Aston: The Man Behind the Mass Spectrograph. *European Journal of Mass Spectrometry*, 13(3):177–190, 6 2007. ISSN 1469-0667. doi: 10.1255/ejms.878. URL http://journals.sagepub.com/doi/10.1255/ejms.878.

[33] Roderic G. Eckenhoff and Ivan J. Dmochowski. *Chemical and Biochemical Approaches for the Study of Anesthetic Function. Part A*. ISBN 9780128127407.

[34] Encyclopaedia Britannica Inc. Imidazole. URL https://www.britannica.com/science/imidazole-organic-compound-class.

[35] Encyclopaedia Britannica Inc. Algebra, 2018. URL https://www.britannica.com/science/algebra.

[36] Encyclopaedia Britannica Inc. Covalent bond, 2018. URL https://www.britannica.com/science/covalent-bond.

[37] Encyclopaedia Britannica Inc. Lysine | chemical compound | Britannica.com, 2018. URL https://www.britannica.com/science/lysine.

[38] Encyclopaedia Britannica Inc. Methionine, 2018. URL https://www.britannica.com/science/methionine.

[39] Encyclopaedia Britannica Inc. Peptide, 2018. URL https://www.britannica.com/science/peptide.

[40] Encyclopaedia Britannica Inc. Proline, 2018. URL https://www.britannica.com/science/proline.

[41] Encyclopaedia Britannica Inc. Topology, 2018. URL https://www.britannica.com/science/topology.

[42] Encyclopaedia Britannica Inc. Tryptophan, 2018. URL https://www.britannica.com/science/tryptophan.

[43] Encyclopaedia Britannica Inc. Tyrosine, 2018. URL https://www.britannica.com/science/tyrosine.

[44] Encyclopaedia Britannica Inc. Valine, 2018. URL https://www.britannica.com/science/valine.

[45] Herbert B. Enderton. *Elements of set theory*. ISBN 9780122384400.

[46] Jimmy K Eng, Bernd Fischer, Jonas Grossmann, and Michael J MacCoss. SEQUEST. *Journal of Proteome Research*, 7:4598–4602, 2008. doi: 10.1021/pr800420s. URL http://dx.doi.org/10.1021/pr800420s.

[47] Marcel Erné. Lattice Topologies with Interval Bases. URL http://www2.iazd.uni-hannover.de/~erne/preprints/Lattice_Top_Intervals.pdf.

[48] A. M. Falick, W. M. Hines, K. F. Medzihradszky, M. A. Baldwin, and B. W. Gibson. Low-mass ions produced from peptides by high-energy collision-induced dissociation in tandem mass spectrometry. *Journal of the American Society for Mass Spectrometry*, 4(11):882–893, 11 1993. ISSN 1044-0305. doi: 10.1016/1044-0305(93)87006-X. URL http://link.springer.com/10.1016/1044-0305(93)87006-X.

[49] Jorge Fernández-de Cossio, Javier Gonzalez, and Vladimir Besada. A computer program to aid the sequencing of peptides in collision-activated decomposition experiments. *Bioinformatics*, 11(4):427–434, 8 1995. ISSN 1367-4803. doi: 10.1093/bioinformatics/11.4.427. URL https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/11.4.427.

[50] Bernd Fischer, Volker Roth, Franz Roos, Jonas Grossmann, Sacha Baginsky, Peter Widmayer, Wilhelm Gruissem, and Joachim M. Buhmann. NovoHMM: A Hidden Markov Model for de Novo Peptide Sequencing. *Analytical Chemistry*, 77(22):7265–7273, 11 2005. ISSN 0003-2700. doi: 10.1021/ac0508853. URL http://www.ncbi.nlm.nih.gov/pubmed/16285674http://pubs.acs.org/doi/abs/10.1021/ac0508853.

[51] Scott C. Forbes and Gordon J. Bell. The acute effects of a low and high dose of oral l -arginine supplementation in young active males at rest. *Applied Physiology, Nutrition, and Metabolism*, 36(3):405–411, 6 2011. ISSN 1715-5312. doi: 10.1139/h11-035. URL http://www.nrcresearchpress.com/doi/10.1139/h11-035.

[52] Ari Frank and Pavel Pevzner. PepNovo: De Novo Peptide Sequencing via Probabilistic Network Modeling. 2005. doi: 10.1021/AC048788H. URL https://pubs.acs.org/doi/abs/10.1021/ac048788h.

[53] Michael L. Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34(3):596–615, 7 1987. ISSN 00045411. doi: 10.1145/28869.28874. URL http://portal.acm.org/citation.cfm?doid=28869.28874.

[54] Orrin Frink. TOPOLOGY IN LATTICES. URL http://www.ams.org/journals/tran/1942-051-00/S0002-9947-1942-0006496-X/S0002-9947-1942-0006496-X.pdf.

[55] Jonas Grossmann, Franz F. Roos, Mark Cieliebak, Zsuzsanna Lipták, Lucas K. Mathis, Matthias Müller, Wilhelm Gruissem, and Sacha Baginsky. AUDENS: A Tool for Automated Peptide de Novo Sequencing. *Journal of Proteome Research*, 4(5):1768–1774, 10 2005. ISSN 1535-3893. doi: 10.1021/pr050070a. URL http://www.ncbi.nlm.nih.gov/pubmed/16212431http://pubs.acs.org/doi/abs/10.1021/pr050070a.

[56] Nitin Gupta, Stephen Tanner, Navdeep Jaitly, Joshua N. Adkins, Mary Lipton, Robert Edwards, Margaret Romine, Andrei Osterman, Vineet Bafna, Richard D. Smith, and Pavel A. Pevzner. Whole proteome analysis of post-translational modifications: Applications of mass-spectrometry for proteogenomic annotation. *Genome Research*, 17(9):1362–1377, 2007. ISSN 10889051. doi: 10.1101/gr.6427907.

[57] Adrian Guthals, Christina Boucher, and Nuno Bandeira. The Generating Function Approach for Peptide Identification in Spectral Networks. *Journal of Computational Biology*, 22(5):353–366, 5 2015. ISSN 1066-5277. doi: 10.1089/cmb.2014.0165. URL http://online.liebertpub.com/doi/10.1089/cmb.2014.0165.

[58] Antonin Guttman. R-trees. *Proceedings of the 1984 ACM SIGMOD international conference on Management of data - SIGMOD '84*, page 47, 1984. ISSN 01635808. doi: 10.1145/602259.602266. URL http://portal.acm.org/citation.cfm?doid=602259.602266.

[59] C. W. Hamm, W. E. Wilson, and D. J. Harvan. Peptide sequencing program. *Bioinformatics*, 2(2):115–118, 6 1986. ISSN 1367-4803. doi: 10.1093/bioinformatics/2.2.115. URL https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/2.2.115.

[60] Qiye He, Jeff Johnston, Julia Zeitlinger, Kansas City, and Kansas City. Proteogenomics: concepts, applications, and computational strategies, 2015. ISSN 1527-5418.

[61] Edmond De Hoffmann and Vincent Stroobant. *Mass Spectrometry - Priniples and Applications.* Number 6. Wiley Subscription Services, Inc., A Wiley Company, 2007. ISBN 9780470033104. doi: 10.1002/mas.20296. URL http://www.ncbi.nlm.nih.gov/pubmed/21057935.

[62] Ian Hunt. Introductory Organic Nomenclature, 2014. URL http://www.chem.ucalgary.ca/courses/351/WebContent/ion/orgnom/index.html.

[63] Merriam-Webster Incorporated. DNA, . URL https://www.merriam-webster.com/dictionary/DNA.

[64] Merriam-Webster Incorporated. RNA, . URL https://www.merriam-webster.com/dictionary/RNA.

[65] Robert A. Ingle. Histidine Biosynthesis. *The Arabidopsis Book*, 9:e0141, 1 2011. ISSN 1543-8120. doi: 10.1199/tab.0141. URL http://www.bioone.org/doi/abs/10.1199/tab.0141.

[66] IonSource. De Novo Peptide Sequencing. URL http://www.ionsource.com/tutorial/DeNovo/b_and_y.htm.

[67] K. Ishikawa and Y. Niwa. Computer-aided peptide sequencing by fast atom bombardment mass spectrometry. *Biological Mass Spectrometry*, 13(7):373–380, 7 1986. ISSN 1052-9306. doi: 10.1002/bms.1200130709. URL http://doi.wiley.com/10.1002/bms.1200130709.

[68] Iupac. monoisotopic mass spectrum. doi: 10.1351/goldbook.M04014. URL https://www.iupac.org/goldbook/M04014.pdf.

[69] Jonathan. Jacky. *The way of Z : practical programming with formal methods*. Cambridge University Press, 1997. ISBN 0521559766.

[70] I. M. (Ioan Mackenzie) James. *History of topology*. Elsevier Science B.V, 1999. ISBN 9780080534077.

[71] Irene Jarchum. Immunology: Transcriptomics in the NICHE. *Nature Methods*, 15(3):165–165, 2 2018. ISSN 1548-7091. doi: 10.1038/nmeth.4625. URL http://www.nature.com/doifinder/10.1038/nmeth.4625.

[72] Johns Hopkins University. What Are Tumors?, 2016. URL http://pathology.jhu.edu/pc/BasicTypes1.php?area=ba.

[73] J David Johnson. {UCINET}: A software tool for network analysis. *Communication Education*, 36(1):92–94, 1987. doi: 10.1080/03634528709378647. URL http://dx.doi.org/10.1080/03634528709378647.

[74] Richard S. Johnson and Klaus Biemann. Computer program (SEQPEP) to aid in the interpretation of high-energy collision tandem mass spectra of peptides. *Biological Mass Spectrometry*, 18(11):945–957, 11 1989. ISSN 0887-6134. doi: 10.1002/bms.1200181102. URL http://doi.wiley.com/10.1002/bms.1200181102.

[75] Richard S Johnson and J Alex Taylor. Searching sequence databases via de novo peptide sequencing by tandem mass spectrometry. *Molecular biotechnology*, 22(3):301–15, 11 2002. ISSN 1073-6085. doi: 10.1385/MB:22:3:301. URL http://link.springer.com/10.1385/MB:22:3:301http://www.ncbi.nlm.nih.gov/pubmed/12448884.

[76] Thomas W. Judson. *Abstract algebra : theory and applications*. PWS Pub. Co, 2017. ISBN 9781944325053. URL https://www.barnesandnoble.com/w/abstract-algebra-thomas-w-judson/1101838049.

[77] Gerald. Karp. *Cell and molecular biology : concepts and experiments*. John Wiley, 2013. ISBN 111830179X.

[78] George A. Khoury, Richard C. Baliban, and Christodoulos A. Floudas. Proteome-wide post-translational modification statistics: frequency analysis and curation of the swiss-prot database. *Scientific Reports*, 1(1):90, 12 2011. ISSN 2045-2322. doi: 10.1038/srep00090. URL http://www.nature.com/articles/srep00090.

[79] George A Khoury, Richard C Baliban, and Christodoulos A Floudas. Proteome-wide post-translational modification statistics: frequency analysis and curation of the swiss-prot database. *Scientific reports*, 1, 9 2011. ISSN 2045-2322. doi: 10.1038/srep00090. URL http://www.ncbi.nlm.nih.gov/pubmed/22034591http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3201773.

[80] Thilo Kielmann, Sergei Gorlatch, Utpal Banerjee, Rocco De Nicola, Jack Dongarra, Piotr Luszczek, Paul Feautrier, Srinivas Aluru, Srinivas Aluru, Robert J. van Glabbeek, Selim G. Akl, Gabriel Zachmann, Robert Geijn, Kazushige Goto, Lawrence Snyder, André Seznec, John L. Gustafson, Jesper Larsson Träff, Robert A. Geijn, Alexander Tiskin, Rajeev Balasubramonian, and Timothy M. Pinkston. Brent's Law. In *Encyclopedia of Parallel Computing*, pages 182–182. Springer US, Boston, MA, 2011. doi: 10.1007/978-0-387-09766-4{_}2194. URL http://www.springerlink.com/index/10.1007/978-0-387-09766-4_2194.

[81] Hyunwoo Kim, Hosung Jo, Heejin Park, and Eunok Paek. HiXCorr : A Portable High-speed X. *Oxford University Press*, pages 2–3, 2015.

[82] Sangtae Kim and Pavel A Pevzner. MS-GF+ makes progress towards a universal database search tool for proteomics. *Nature Communications*, 5:5277, 2014. ISSN 2041-1723. doi: 10.1038/ncomms6277. URL http://www.nature.com/doifinder/10.1038/ncomms6277.

[83] Masahiko KISUMI, Saburo KOMATSUBARA, and Ichiro CHIBATA. Pathway for Isoleucine Formation from Pyruvate by Leucine Biosynthetic Enzymes in Leucine-Accumulating Isoleucine Revertants of Serratia marcescens. *The Journal of Biochemistry*, 82(1):95–103, 7 1977. ISSN 1756-2651. doi: 10.1093/oxfordjournals.jbchem.a131698. URL https://academic.oup.com/jb/article-lookup/doi/10.1093/oxfordjournals.jbchem.a131698.

[84] Aaron A. Klammer, Christopher Y. Park, and William Stafford Noble. Statistical Calibration of the SEQUEST XCorr Function. *Journal of Proteome Research*, 8 (4):2106–2113, 4 2009. ISSN 1535-3893. doi: 10.1021/pr8011107. URL http://pubs.acs.org/doi/abs/10.1021/pr8011107.

[85] Jure Leskovec Stanford Univ Anand Rajaraman, Jeffrey D Ullman, Anand Rajaraman, Jure Leskovec, and Jeffrey D Ullman ii. Mining of Massive Datasets. 2010. URL http://infolab.stanford.edu/~ullman/mmds/book.pdf.

[86] Aristid Lindenmayer. Mathematical models for cellular interactions in development I. Filaments with one-sided inputs. *Journal of Theoretical Biology*, 18(3):280–299, 3 1968. ISSN 0022-5193. doi: 10.1016/0022-5193(68)90079-9. URL https://www.sciencedirect.com/science/article/pii/0022519368900799.

[87] Harvey Lodish, Arnold Berk, S Lawrence Zipursky, Paul Matsudaira, David Baltimore, and James Darnell. Post-Translational Modifications and Quality Control in the Rough ER. 2000. URL https://www.ncbi.nlm.nih.gov/books/NBK21741/.

[88] Bingwen Lu and Ting Chen. A Suboptimal Algorithm for De Novo Peptide Sequencing via Tandem Mass Spectrometry. *Journal of Computational Biology*, 10(1):1–12, 2 2003. ISSN 1066-5277. doi: 10.1089/106652703763255633. URL http://www.liebertonline.com/doi/abs/10.1089/106652703763255633.

[89] Erik K Malm, Vaibhav Srivastava, Gustav Sundqvist, and Vincent Bulone. APP: an Automated Proteomics Pipeline for the analysis of mass spectrometry data based on multiple open access tools. *BMC bioinformatics*, 15(1):6600, 2014. ISSN 1471-2105. doi: 10.1186/s12859-014-0441-8. URL http://www.ncbi.nlm.nih.gov/pubmed/25547515.

[90] Lewis N. Mander and Hung-wen Liu. *Comprehensive natural products II : chemistry and biology*. ISBN 9780080453828. URL https://www.sciencedirect.com/science/referenceworks/9780080453828.

[91] Norman S. Matloff. *Parallel computing for data science : with examples in R, C++ and CUDA*. ISBN 9781466587038. URL https://books.google.co.cr/books?id=SsbECQAAQBAJ&printsec=frontcover&dq=matloff+parallel&hl=en&sa=X&redir_esc=y&v=onepage&q=matloffparallel&f=false.

[92] Matrix Science. Mascot overview | Protein identification software for mass spec data, 2014. URL http://www.matrixscience.com/search_intro.html.

[93] Michael D. McCool, James. Reinders, and Arch. Robison. *Structured parallel programming : patterns for efficient computation*. Elsevier/Morgan Kaufmann, 2012. ISBN 9780124159938. URL http://dl.acm.org/citation.cfm?id=2385466.

[94] Institute of Medicine. *Dietary Reference Intakes for Thiamin, Riboflavin, Niacin, Vitamin B6, Folate, Vitamin B12, Pantothenic Acid, Biotin, and Choline*. National Academies Press, Washington, D.C., 6 1998. ISBN 978-0-309-06554-2. doi: 10.17226/6015. URL http://www.nap.edu/catalog/6015.

[95] Merriam-Webster Incorporated. Aliphatic definition, 2017. URL https://www.merriam-webster.com/dictionary/aliphatic.

[96] Merriam-Webster Incorporated. Alpha amino acid defintion, 2017. URL https://www.merriam-webster.com/medical/alphaamino.

[97] Merriam-Webster Incorporated. Carboxylic definition, 2017. URL https://www.merriam-webster.com/dictionary/carboxylichttps://www.merriam-webster.com/dictionary/logarithm.

[98] Merriam-Webster Incorporated. Methyl definition, 2017. URL https://www.merriam-webster.com/dictionary/methyl.

[99] Merriam-Webster Incorporated. Nonpolar definition, 2017. URL https://www.merriam-webster.com/dictionary/nonpolar.

[100] Merriam-Webster Incorporated. Guanidino, 2018. URL https://www.merriam-webster.com/dictionary/guanidino.

[101] Merriam-Webster Incorporated. Hydrolysis, 2018. URL https://www.merriam-webster.com/dictionary/hydrolysis.

[102] Jeffrey A. Milloy, Brendan K. Faherty, and Scott A. Gerber. Tempest: GPU-CPU computing for high-throughput database spectral matching. *Journal of Proteome Research*, 11(7):3581–3591, 2012. ISSN 15353893. doi: 10.1021/pr300338p.

[103] Douglas C Montgomery. *Design and Analysis of Experiments*. John Wiley & Sons, 2001. ISBN 0471316490.

[104] Douglas C Montgomery. *Design and Analysis of Experiments*, volume 2. 2012. ISBN 0471316490. doi: 10.1198/tech.2006.s372. URL http://cataleg.uab.cat/record=b1764873~S1*cat.

[105] J P Mothet, A T Parent, H Wolosker, R O Brady, D J Linden, C D Ferris, M A Rogawski, S H Snyder, and Solomon H. Snyder. D-serine is an endogenous ligand for the glycine site of the N-methyl-D-aspartate receptor. *Proceedings of the National Academy of Sciences of the United States of America*, 97(9):4926–31, 4 2000. ISSN 0027-8424. URL http://www.ncbi.nlm.nih.gov/pubmed/10781100http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC18334.

[106] National Cancer Institute. Immunotherapy : Using the Immune System to Treat Cancer, 2014. URL https://www.cancer.gov/research/areas/treatment/immunotherapy-using-immune-system.

[107] National Cancer Institute. Definition of polypeptide, 2018. URL https://www.cancer.gov/publications/dictionaries/cancer-terms/def/polypeptide.

[108] Alexey I Nesvizhskii. Proteogenomics: concepts, applications and computational strategies. *Nature Methods*, 11(11):1114–1125, 11 2014. ISSN 1548-7091. doi: 10.1038/nmeth.3144. URL http://www.nature.com/articles/nmeth.3144.

[109] Steffen Neumann and Laurent Gatto. msdata: Various Mass Spectrometry raw data example files, 2016.

[110] Yohei Okubo, Hiroshi Sekiya, Shigeyuki Namiki, Hirokazu Sakamoto, Sho Iinuma, Miwako Yamasaki, Masahiko Watanabe, Kenzo Hirose, and Masamitsu Iino. Imaging extrasynaptic glutamate dynamics in the brain. *Proceedings of the National Academy of Sciences of the United States of America*, 107(14):6526–31, 4 2010. ISSN 1091-6490. doi: 10.1073/pnas.0913154107. URL http://www.ncbi.nlm.nih.gov/pubmed/20308566http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC2851965.

[111] Sheldon J. Park and Jennifer R. Cochran. *Protein engineering and design*. CRC Press, 2010. ISBN 9781420076585.

[112] Rob Patro, Stephen M. Mount, and Carl Kingsford. Sailfish: Alignment-free Isoform Quantification from RNA-seq Reads using Lightweight Algorithms. *Nature Biotechnology*, 2013. doi: 10.1038/nbt.2862. URL http://arxiv.org/abs/1308.3700http://dx.doi.org/10.1038/nbt.2862.

[113] Elisha Peterson. Army Math Lattice - ElishaPeterson, 2018. URL http://elishapeterson.wikidot.com/tikz-snippets:army-math-graph.

[114] Charles C. Pinter. *A book of abstract algebra*. Dover Publications, 2010. ISBN 9780486474175.

[115] Elena A Ponomarenko, Ekaterina V Poverennaya, Ekaterina V Ilgisonis, Mikhail A Pyatnitskiy, Arthur T Kopylov, Victor G Zgoda, Andrey V Lisitsa, and Alexander I Archakov. The Size of the Human Proteome: The Width and Depth. *International journal of analytical chemistry*, 2016:7436849, 2016. ISSN 1687-8760. doi: 10.1155/2016/7436849. URL http://www.ncbi.nlm.nih.gov/pubmed/27298622http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4889822.

[116] {Princeton University}. Mass Spec Definitions, 2016. URL http://proteomics.princeton.edu/services/mass-spec-results-definitions.

[117] Badr Rai, Christophe Chassagnole, Thierry Letellier, David A Fell, and Jean-Pierre Mazat. Threonine synthesis from aspartate in Escherichia coli cell-free extracts : pathway dynamics. *Biochem. J*, 356:425–432, 2001. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1221853/pdf/11368769.pdf.

[118] Anand Rajaraman and Jeffrey D Ullman. Mining of Massive Datasets. *Lecture Notes for Stanford CS345A Web Mining*, 67:328, 2011. ISSN 01420615. doi: 10.1017/CBO9781139058452. URL http://ebooks.cambridge.org/ref/id/CBO9781139058452.

[119] J M Rogers and H Suga. Discovering functional, non-proteinogenic amino acid containing, peptides using genetic code reprogramming. *Organic & biomolecular chemistry*, 13(36):9353–63, 9 2015. ISSN 1477-0539. doi: 10.1039/c5ob01336d. URL http://www.ncbi.nlm.nih.gov/pubmed/26280393.

[120] Jeffrey S Ross, Jonathan A Fletcher, Gerald P Linette, James Stec, Edward Clark, Mark Ayers, W Fraser Symmans, Lajos Pusztai, and Kenneth J Bloom. The Her-2/neu gene and protein in breast cancer 2003: biomarker and target of therapy. *The oncologist*, 8(4):307–25, 8 2003. ISSN 1083-7159. doi: 10.1634/THEONCOLOGIST.8-4-307. URL http://www.ncbi.nlm.nih.gov/pubmed/12897328.

[121] Elizabeth K Ruzzo, José-Mario Capo-Chichi, Bruria Ben-Zeev, David Chitayat, Hanqian Mao, Andrea L Pappas, Yuki Hitomi, Yi-Fan Lu, Xiaodi Yao, Fadi F Hamdan, Kimberly Pelak, Haike Reznik-Wolf, Ifat Bar-Joseph, Danit Oz-Levi, Dorit Lev, Tally Lerman-Sagie, Esther Leshinsky-Silver, Yair Anikster, Edna Ben-Asher, Tsviya Olender, Laurence Colleaux, Jean-Claude Décarie, Susan Blaser, Brenda Banwell, Rasesh B Joshi, Xiao-Ping He, Lysanne Patry, Rachel J Silver, Sylvia Dobrzeniecka, Mohammad S Islam, Abul Hasnat, Mark E Samuels, Dipendra K Aryal, Ramona M Rodriguiz, Yong-Hui Jiang, William C Wetsel, James O McNamara, Guy A Rouleau, Debra L Silver, Doron Lancet, Elon Pras, Grant A Mitchell, Jacques L Michaud, and David B Goldstein. Deficiency of asparagine synthetase causes congenital microcephaly and a progressive form of encephalopathy. *Neuron*, 80(2):429–41, 10 2013. ISSN 1097-4199. doi: 10.1016/j.neuron.2013.08.013. URL http://www.ncbi.nlm.nih.gov/pubmed/24139043http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3820368.

[122] T. Sakurai, T. Matsuo, H. Matsuda, and I. Katakuse. PAAS 3: A computer program to determine probable sequence of peptides from mass spectrometric data. *Biological Mass Spectrometry*, 11(8):396–399, 8 1984. ISSN 0306-042X. doi: 10.1002/bms.1200110806. URL http://doi.wiley.com/10.1002/bms.1200110806.

[123] Juan Manuel Sánchez Calvo, Rosa Del, Campo Moreno, Antonio López, and San Román. MEMORIA PARA OPTAR AL GRADO DE DOCTOR PRESENTADA POR. 2013.

[124] Nachimuthu. Saraswathy and Ponnusamy. Ramalingam. *Concepts and techniques in genomics and proteomics*. Woodhead Pub, 2011. ISBN 9781907568107.

[125] Nachimuthu Saraswathy, Ponnusamy Ramalingam, Nachimuthu Saraswathy, and Ponnusamy Ramalingam. 5 – DNA sequencing methods. In *Concepts and Techniques in Genomics and Proteomics*, pages 57–76. 2011. ISBN 9781907568107. doi: 10.1533/9781908818058.57.

[126] H Sawai and L E Orgel. *IUPAC Compendium of Chemical Terminology*. Number 12. International Union of Pure and Applied Chemistry, 1975. ISBN 0-9678550-9-8. doi: 10.1351/goldbook.I03352. URL http://goldbook.iupac.org/PDF/goldbook.pdfhttp://goldbook.iupac.org/I03352.htmlhttp://www.ncbi.nlm.nih.gov/pubmed/1141584.

[127] Robert A Saxton, Kevin E Knockenhauer, Rachel L Wolfson, Lynne Chantranupong, Michael E Pacold, Tim Wang, Thomas U Schwartz, and David M Sabatini. Structural basis for leucine sensing by the Sestrin2-mTORC1 pathway. *Science (New York, N.Y.)*, 351(6268):53–8, 1 2016. ISSN 1095-9203. doi: 10.1126/science.aad2087. URL http://www.ncbi.nlm.nih.gov/pubmed/26586190http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4698039.

[128] Ole Schulz-Trieglaff, Nico Pfeifer, Clemens Gropl, Oliver Kohlbacher, and Knut Reinert. LC-MSsim - a simulation software for liquid chromatography mass spectrometry data. *BMC Bioinformatics*, 9(1):423, 2008. ISSN 1471-2105. doi: 10.1186/1471-2105-9-423. URL http://www.biomedcentral.com/1471-2105/9/423http://www.biomedcentral.com/content/pdf/1471-2105-9-423.pdf.

[129] Michaela Scigelova and Alexander Makarov. Orbitrap mass analyzer - Overview and applications in proteomics. In *Proteomics*, volume 1, pages 16–21, 2006. ISBN 1615-9853. doi: 10.1002/pmic.200600528.

[130] Scitable. Proteome. URL https://www.nature.com/scitable/definition/proteome-297.

[131] Scott Meyers. Universal References in C++11. URL https://isocpp.org/blog/2012/11/universal-references-in-c11-scott-meyers.

[132] § Sergey Pevtsov, †, § Irina Fedulova, †, ‡ Hamid Mirzaei, ‡ Charles Buck, , and ‡ Xiang Zhang*. Performance Evaluation of Existing De Novo Sequencing Algorithms. 2006. doi: 10.1021/PR060222H. URL https://pubs.acs.org/doi/abs/10.1021/pr060222h.

[133] Satish Shirali and Harkrishnan L. Vasudeva. *Metric Spaces*. Springer SciencebBusiness Media, London, 1 edition, 2006. ISBN 1852339225. doi: 10.1017/CBO9780511566141.

[134] Marshall M. Siegel and Norman Bauman. An efficient algorithm for sequencing peptides using fast atom bombardment mass spectral data. *Biological Mass Spectrometry*, 15(6):333–343, 3 1988. ISSN 0887-6134. doi: 10.1002/bms.1200150606. URL http://doi.wiley.com/10.1002/bms.1200150606.

[135] Michael Sipser. Introduction to the Theory of Computation, Third Edition. URL https://theswissbay.ch/pdf/Book/Introduction%20to%20the%20theory%20of%20computation_third%20edition%20-%20Michael%20Sipser.pdf.

[136] O David Sparkman. Book & Media Reviews Mass Spectrometry Desk Reference. *Chemical Education Today 168 Journal of Chemical Education @BULLET*, 78(2), 2001. URL https://pubs.acs.org/doi/pdf/10.1021/ed078p168.2.

[137] Gordon Squires. Francis Aston and the mass spectrograph. *Journal of the Chemical Society, Dalton Transactions*, (23):3893–3900, 1998. ISSN 03009246. doi: 10.1039/a804629h. URL http://xlink.rsc.org/?DOI=a804629h.

[138] Standard C++ Foundation. C++11 Language Extensions, 2018. URL https://isocpp.org/wiki/faq/cpp11-language-classes.

[139] Standard C++ Foundation. C++14 Language Extensions, 2018. URL https://isocpp.org/wiki/faq/cpp14-language.

[140] Marc Sturm, Andreas Bertsch, Clemens Gröpl, Andreas Hildebrandt, Rene Hussong, Eva Lange, Nico Pfeifer, Ole Schulz-Trieglaff, Alexandra Zerck, Knut Reinert, and Oliver Kohlbacher. OpenMS – An open-source software framework for mass spectrometry. *BMC Bioinformatics*, 9(1):163, 2008. ISSN 1471-2105. doi: 10.1186/1471-2105-9-163. URL http://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-9-163.

[141] Patrick Suppes. *Axiomatic set theory,*. Dover Publications, 1972. ISBN 9780486616308.

[142] David L Tabb. An algorithm for isoelectric point estimation. *Ratio*, pages 1–5, 2003. URL http://fields.scripps.edu/DTASelect/20010710-pI-Algorithm.pdf.

[143] H Tapiero, G Mathé, P Couvreur, and K.D Tew. I. Arginine. *Biomedicine & Pharmacotherapy*, 56(9):439–445, 11 2002. ISSN 0753-3322. doi: 10.1016/S0753-3322(02)00284-6. URL https://www.sciencedirect.com/science/article/abs/pii/S0753332202002846.

[144] J A Taylor and R S Johnson. Implementation and uses of automated de novo peptide sequencing by tandem mass spectrometry. *Analytical chemistry*, 73 (11):2594–604, 6 2001. ISSN 0003-2700. URL http://www.ncbi.nlm.nih.gov/pubmed/11403305.

[145] J. Alex Taylor and Richard S. Johnson. Sequence database searches viade novo peptide sequencing by tandem mass spectrometry. *Rapid Communications in Mass Spectrometry*, 11(9):1067–1075, 6 1997. ISSN 0951-4198. doi: 10.1002/(SICI)1097-0231(19970615)11:9<1067::AID-RCM953>3.0.CO;2-L. URL http://www.ncbi.nlm.nih.gov/pubmed/9204580http://doi.wiley.com/10.1002/%28SICI%291097-0231%2819970615%2911%3A9%3C1067%3A%3AAID-RCM953%3E3.0.CO%3B2-L.

[146] Amalio Telenti, Levi C T Pierce, William H Biggs, Julia Iulio, M Martin, and F Ewen. Deep Sequencing of 10 , 000 Human Genomes a a. 2016.

[147] The Open Group. POSIX: limits.h. URL http://pubs.opengroup.org/onlinepubs/009695399/basedefs/limits.h.html.

[148] Cole Trapnell, Brian a Williams, Geo Pertea, Ali Mortazavi, Gordon Kwan, Marijke J van Baren, Steven L Salzberg, Barbara J Wold, and Lior Pachter. Transcript assembly and abundance estimation from RNA-Seq reveals thousands of new transcripts and switching among isoforms. *Nature*

*Biotechnology*, 28(5):511–515, 2011. ISSN 1546-1696. doi: 10.1038/nbt.1621. Transcript. URL http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3146043&tool=pmcentrez&rendertype=abstract.

[149] U.S. National Library of Medicine. Tumor: MedlinePlus Medical Encyclopedia, 2018. URL https://medlineplus.gov/ency/article/001310.htm.

[150] Vladimir Prus. Boost Program Options, 2004. URL http://www.boost.org/doc/libs/1_66_0/doc/html/program_options.html.

[151] Donald Voet, Judith G. Voet, and Charlotte W. Pratt. *Fundamentals of biochemistry : life at the molecular level*. Fifth edition. edition, 2016. ISBN 9781118918401. URL https://www.worldcat.org/title/fundamentals-of-biochemistry-life-at-the-molecular-level/oclc/910538334.

[152] Stefan Waldmann. Construction of Topological Spaces. In *Topology*, pages 41–57. Springer International Publishing, Cham, 2014. doi: 10.1007/978-3-319-09680-3{_}3. URL http://link.springer.com/10.1007/978-3-319-09680-3_3.

[153] Stefan Waldmann. Convergence in Topological Spaces. In *Topology*, pages 59–71. Springer International Publishing, Cham, 2014. doi: 10.1007/978-3-319-09680-3{_}4. URL http://link.springer.com/10.1007/978-3-319-09680-3_4.

[154] Stefan Waldmann. Introduction to topology. In *Topology*, pages 1–3. Springer International Publishing, Cham, 2014. doi: 10.1007/978-3-319-09680-3{_}1. URL http://link.springer.com/10.1007/978-3-319-09680-3_1.

[155] Stefan Waldmann. Topological Spaces and Continuity. In *Topology*, pages 5–40. Springer International Publishing, Cham, 2014. doi: 10.1007/978-3-319-09680-3{_}2. URL http://link.springer.com/10.1007/978-3-319-09680-3_2.

[156] Stefan Waldmann. *Topology*. Springer International Publishing, Cham, 2014. ISBN 978-3-319-09679-7. doi: 10.1007/978-3-319-09680-3. URL http://link.springer.com/10.1007/978-3-319-09680-3.

[157] Ronald E Walpole, Raymond H Myers, Sharon L Myers, and Keying Ye. *Probability and Statistics for Engineers and Scientists*, volume 3rd. Prentice Hall, 2012. ISBN 0132047675.

[158] Z Wang, S Wei, S H Reed, X Wu, J Q Svejstrup, W J Feaver, R D Kornberg, and E C Friedberg. The RAD7, RAD16, and RAD23 genes of Saccharomyces cerevisiae: requirement for transcription-independent nucleotide excision repair in vitro and interactions between the gene products. *Molecular and cellular biology*, 17(2):635–43, 2 1997. ISSN

0270-7306. URL http://www.ncbi.nlm.nih.gov/pubmed/9001217http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC231789.

[159] Eric W. Weisstein. Toroid. URL http://mathworld.wolfram.com/Toroid.html.

[160] K Williams, J Chao, K Kashiwagi, T Masuko, K Igarashi, Amanda L. Jacob, James P. Snyder, Stephen F. Traynelis, and David J. A. Wyllie. Activation of N-methyl-D-aspartate receptors by glycine: role of an aspartate residue in the M3-M4 loop of the NR1 subunit. *Molecular pharmacology*, 50(4):701–8, 10 1996. ISSN 0026-895X. doi: 10.1124/mol.66.2.209. URL http://www.ncbi.nlm.nih.gov/pubmed/8863813.

[161] Guoyao Wu, Laurie A Jaeger, Fuller W Bazer, and J.Marc Rhoads. Arginine deficiency in preterm infants: Biochemical mechanisms and nutritional implications. *The Journal of Nutritional Biochemistry*, 15(8):442–451, 8 2004. ISSN 0955-2863. doi: 10.1016/J.JNUTBIO.2003.11.010. URL https://www.sciencedirect.com/science/article/pii/S0955286304000701?via%3Dihub.

[162] www.3Ders.org. 3ders.org - Virginia student uses 3D printed models to demonstrate mathematical concept of topology | 3D Printer News & 3D Printing News, 2018. URL https://www.3ders.org/articles/20170911-virginia-student-uses-3d-printed-models-to-demonstrate-mathematical-concep html.

[163] Kevin P. Wyche, Robert S. Blake, Kerry A. Willis, Paul S. Monks, and Andrew M. Ellis. Differentiation of isobaric compounds using chemical ionization reaction mass spectrometry. *Rapid Communications in Mass Spectrometry*, 19(22):3356–3362, 11 2005. ISSN 0951-4198. doi: 10.1002/rcm.2202. URL http://doi.wiley.com/10.1002/rcm.2202.

[164] Xiang-Jiao Yang and Edward Seto. Lysine acetylation: codified crosstalk with other posttranslational modifications. *Molecular cell*, 31(4):449–61, 8 2008. ISSN 1097-4164. doi: 10.1016/j.molcel.2008.07.002. URL http://www.ncbi.nlm.nih.gov/pubmed/18722172http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC2551738.

[165] Mariia Yuneva, Nicola Zamboni, Peter Oefner, Ravi Sachidanandam, and Yuri Lazebnik. Deficiency in glutamine but not glucose induces MYC-dependent apoptosis in human cells. *The Journal of cell biology*, 178(1):93–105, 7 2007. ISSN 0021-9525. doi: 10.1083/jcb.200703099. URL http://www.ncbi.nlm.nih.gov/pubmed/17606868http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC2064426.

[166] J. Zhang, L. Xin, B. Shan, W. Chen, M. Xie, D. Yuen, W. Zhang, Z. Zhang, G. A. Lajoie, and B. Ma. PEAKS DB: De Novo Sequencing Assisted Database Search for Sensitive and Accurate Peptide Identification. *Molecular & Cellular*

*Proteomics*, 11(4):M111.010587–M111.010587, 4 2012. ISSN 1535-9476. doi: 10.1074/mcp.M111.010587. URL http://www.mcponline.org/cgi/doi/10.1074/mcp.M111.010587.

# License

# 8.1. Creative Commons Legal Code

## 8.1.1. Attribution-NonCommercial-ShareAlike 4.0 International

Official translations of this license are available in other languages.

Creative Commons Corporation ("Creative Commons") is not a law firm and does not provide legal services or legal advice. Distribution of Creative Commons public licenses does not create a lawyer-client or other relationship. Creative Commons makes its licenses and related information available on an "as-is" basis. Creative Commons gives no warranties regarding its licenses, any material licensed under their terms and conditions, or any related information. Creative Commons disclaims all liability for damages resulting from their use to the fullest extent possible.

**Using Creative Commons Public Licenses**

Creative Commons public licenses provide a standard set of terms and conditions that creators and other rights holders may use to share original works of authorship and other material subject to copyright and certain other rights specified in the public license below. The following considerations are for informational purposes only, are not exhaustive, and do not form part of our licenses.

> **Considerations for licensors:** Our public licenses are intended for use by those authorized to give the public permission to use material in ways otherwise restricted by copyright and certain other rights. Our licenses are irrevocable. Licensors should read and understand the terms and conditions of the license they choose before applying it. Licensors should also secure all rights necessary before applying our licenses so that the public can reuse the material as expected. Licensors should clearly mark any material not subject to the license. This includes other CC-licensed material, or material used under an exception or limitation to copyright. More considerations for

licensors.

**Considerations for the public:** By using one of our public licenses, a licensor grants the public permission to use the licensed material under specified terms and conditions. If the licensor's permission is not necessary for any reason–for example, because of any applicable exception or limitation to copyright–then that use is not regulated by the license. Our licenses grant only permissions under copyright and certain other rights that a licensor has authority to grant. Use of the licensed material may still be restricted for other reasons, including because others have copyright or other rights in the material. A licensor may make special requests, such as asking that all changes be marked or described. Although not required by our licenses, you are encouraged to respect those requests where reasonable. More considerations for the public.

#### 8.1.1.1. Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International Public License

By exercising the Licensed Rights (defined below), You accept and agree to be bound by the terms and conditions of this Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International Public License ("Public License"). To the extent this Public License may be interpreted as a contract, You are granted the Licensed Rights in consideration of Your acceptance of these terms and conditions, and the Licensor grants You such rights in consideration of benefits the Licensor receives from making the Licensed Material available under these terms and conditions.

**Section 1 – Definitions.**

1. **Adapted Material** means material subject to Copyright and Similar Rights that is derived from or based upon the Licensed Material and in which the Licensed Material is translated, altered, arranged, transformed, or otherwise modified in

a manner requiring permission under the Copyright and Similar Rights held by the Licensor. For purposes of this Public License, where the Licensed Material is a musical work, performance, or sound recording, Adapted Material is always produced where the Licensed Material is synched in timed relation with a moving image.

2. **Adapter's License** means the license You apply to Your Copyright and Similar Rights in Your contributions to Adapted Material in accordance with the terms and conditions of this Public License.

3. **BY-NC-SA Compatible License** means a license listed at creativecommons.org/compatiblelicenses, approved by Creative Commons as essentially the equivalent of this Public License.

4. **Copyright and Similar Rights** means copyright and/or similar rights closely related to copyright including, without limitation, performance, broadcast, sound recording, and Sui Generis Database Rights, without regard to how the rights are labeled or categorized. For purposes of this Public License, the rights specified in Section 2(b)(1)-(2) are not Copyright and Similar Rights.

5. **Effective Technological Measures** means those measures that, in the absence of proper authority, may not be circumvented under laws fulfilling obligations under Article 11 of the WIPO Copyright Treaty adopted on December 20, 1996, and/or similar international agreements.

6. **Exceptions and Limitations** means fair use, fair dealing, and/or any other exception or limitation to Copyright and Similar Rights that applies to Your use of the Licensed Material.

7. **License Elements** means the license attributes listed in the name of a Creative Commons Public License. The License Elements of this Public License are

Attribution, NonCommercial, and ShareAlike.

8. **Licensed Material** means the artistic or literary work, database, or other material to which the Licensor applied this Public License.

9. **Licensed Rights** means the rights granted to You subject to the terms and conditions of this Public License, which are limited to all Copyright and Similar Rights that apply to Your use of the Licensed Material and that the Licensor has authority to license.

10. **Licensor** means the individual(s) or entity(ies) granting rights under this Public License.

11. **NonCommercial** means not primarily intended for or directed towards commercial advantage or monetary compensation. For purposes of this Public License, the exchange of the Licensed Material for other material subject to Copyright and Similar Rights by digital file-sharing or similar means is NonCommercial provided there is no payment of monetary compensation in connection with the exchange.

12. **Share** means to provide material to the public by any means or process that requires permission under the Licensed Rights, such as reproduction, public display, public performance, distribution, dissemination, communication, or importation, and to make material available to the public including in ways that members of the public may access the material from a place and at a time individually chosen by them.

13. **Sui Generis Database Rights** means rights other than copyright resulting from Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases, as amended and/or succeeded, as well as other essentially equivalent rights anywhere in the world.

14. **You** means the individual or entity exercising the Licensed Rights under this Public License. **Your** has a corresponding meaning.

**Section 2 – Scope.**

1. **License grant**.

   *a*) Subject to the terms and conditions of this Public License, the Licensor hereby grants You a worldwide, royalty-free, non-sublicensable, non-exclusive, irrevocable license to exercise the Licensed Rights in the Licensed Material to:

      1) reproduce and Share the Licensed Material, in whole or in part, for NonCommercial purposes only; and

      2) produce, reproduce, and Share Adapted Material for NonCommercial purposes only.

   *b*) Exceptions and Limitations. For the avoidance of doubt, where Exceptions and Limitations apply to Your use, this Public License does not apply, and You do not need to comply with its terms and conditions.

   *c*) Term. The term of this Public License is specified in Section 6(a).

   *d*) Media and formats; technical modifications allowed. The Licensor authorizes You to exercise the Licensed Rights in all media and formats whether now known or hereafter created, and to make technical modifications necessary to do so. The Licensor waives and/or agrees not to assert any right or authority to forbid You from making technical modifications necessary to exercise the Licensed Rights, including technical modifications necessary to circumvent Effective Technological Measures. For purposes of this Public License, simply making modifications authorized by this Section 2(a)(4) never produces Adapted Material.

*e*) Downstream recipients.

1) Offer from the Licensor – Licensed Material. Every recipient of the Licensed Material automatically receives an offer from the Licensor to exercise the Licensed Rights under the terms and conditions of this Public License.

2) Additional offer from the Licensor – Adapted Material. Every recipient of Adapted Material from You automatically receives an offer from the Licensor to exercise the Licensed Rights in the Adapted Material under the conditions of the Adapter's License You apply.

3) No downstream restrictions. You may not offer or impose any additional or different terms or conditions on, or apply any Effective Technological Measures to, the Licensed Material if doing so restricts exercise of the Licensed Rights by any recipient of the Licensed Material.

*f*) No endorsement. Nothing in this Public License constitutes or may be construed as permission to assert or imply that You are, or that Your use of the Licensed Material is, connected with, or sponsored, endorsed, or granted official status by, the Licensor or others designated to receive attribution as provided in Section 3(a)(1)(A)(i).

**Other rights**.

2. *a*) Moral rights, such as the right of integrity, are not licensed under this Public License, nor are publicity, privacy, and/or other similar personality rights; however, to the extent possible, the Licensor waives and/or agrees not to assert any such rights held by the Licensor to the limited extent necessary to allow You to exercise the Licensed Rights, but not otherwise.

*b*) Patent and trademark rights are not licensed under this Public License.

*c*) To the extent possible, the Licensor waives any right to collect royalties from

You for the exercise of the Licensed Rights, whether directly or through a collecting society under any voluntary or waivable statutory or compulsory licensing scheme. In all other cases the Licensor expressly reserves any right to collect such royalties, including when the Licensed Material is used other than for NonCommercial purposes.

**Section 3 – License Conditions.**

Your exercise of the Licensed Rights is expressly made subject to the following conditions.

1. **Attribution**.

   *a*) If You Share the Licensed Material (including in modified form), You must:

      1) retain the following if it is supplied by the Licensor with the Licensed Material:

         *a′* identification of the creator(s) of the Licensed Material and any others designated to receive attribution, in any reasonable manner requested by the Licensor (including by pseudonym if designated);

         *b′* a copyright notice;

         *c′* a notice that refers to this Public License;

         *d′* a notice that refers to the disclaimer of warranties;

         *e′* a URI or hyperlink to the Licensed Material to the extent reasonably practicable;

      2) indicate if You modified the Licensed Material and retain an indication of any previous modifications; and

      3) indicate the Licensed Material is licensed under this Public License, and include the text of, or the URI or hyperlink to, this Public License.

   *b*) You may satisfy the conditions in Section 3(a)(1) in any reasonable manner based on the medium, means, and context in which You Share the Licensed

Material. For example, it may be reasonable to satisfy the conditions by providing a URI or hyperlink to a resource that includes the required information.

*c*) If requested by the Licensor, You must remove any of the information required by Section 3(a)(1)(A) to the extent reasonably practicable.

2. **ShareAlike**.

In addition to the conditions in Section 3(a), if You Share Adapted Material You produce, the following conditions also apply.

*a*) The Adapter's License You apply must be a Creative Commons license with the same License Elements, this version or later, or a BY-NC-SA Compatible License.

*b*) You must include the text of, or the URI or hyperlink to, the Adapter's License You apply. You may satisfy this condition in any reasonable manner based on the medium, means, and context in which You Share Adapted Material.

*c*) You may not offer or impose any additional or different terms or conditions on, or apply any Effective Technological Measures to, Adapted Material that restrict exercise of the rights granted under the Adapter's License You apply.

**Section 4 – Sui Generis Database Rights.**

Where the Licensed Rights include Sui Generis Database Rights that apply to Your use of the Licensed Material:

1. for the avoidance of doubt, Section 2(a)(1) grants You the right to extract, reuse, reproduce, and Share all or a substantial portion of the contents of the database for NonCommercial purposes only;

2. if You include all or a substantial portion of the database contents in a database in which You have Sui Generis Database Rights, then the database in which You have Sui Generis Database Rights (but not its individual contents) is Adapted Material, including for purposes of Section 3(b); and

3. You must comply with the conditions in Section 3(a) if You Share all or a substantial portion of the contents of the database.

For the avoidance of doubt, this Section 4 supplements and does not replace Your obligations under this Public License where the Licensed Rights include other Copyright and Similar Rights.

**Section 5 – Disclaimer of Warranties and Limitation of Liability.**

1. **Unless otherwise separately undertaken by the Licensor, to the extent possible, the Licensor offers the Licensed Material as-is and as-available, and makes no representations or warranties of any kind concerning the Licensed Material, whether express, implied, statutory, or other. This includes, without limitation, warranties of title, merchantability, fitness for a particular purpose, non-infringement, absence of latent or other defects, accuracy, or the presence or absence of errors, whether or not known or discoverable. Where disclaimers of warranties are not allowed in full or in part, this disclaimer may not apply to You.**

2. **To the extent possible, in no event will the Licensor be liable to You on any legal theory (including, without limitation, negligence) or otherwise for any direct, special, indirect, incidental, consequential, punitive, exemplary, or other losses, costs, expenses, or damages arising out of this Public License or use of the Licensed Material, even if the Licensor has been advised of the possibility of such losses,**

**costs, expenses, or damages. Where a limitation of liability is not allowed in full or in part, this limitation may not apply to You.**

1. The disclaimer of warranties and limitation of liability provided above shall be interpreted in a manner that, to the extent possible, most closely approximates an absolute disclaimer and waiver of all liability.

**Section 6 – Term and Termination.**

1. This Public License applies for the term of the Copyright and Similar Rights licensed here. However, if You fail to comply with this Public License, then Your rights under this Public License terminate automatically.

2. Where Your right to use the Licensed Material has terminated under Section 6(a), it reinstates:

   *a*) automatically as of the date the violation is cured, provided it is cured within 30 days of Your discovery of the violation; or

   *b*) upon express reinstatement by the Licensor.

   For the avoidance of doubt, this Section 6(b) does not affect any right the Licensor may have to seek remedies for Your violations of this Public License.

3. For the avoidance of doubt, the Licensor may also offer the Licensed Material under separate terms or conditions or stop distributing the Licensed Material at any time; however, doing so will not terminate this Public License.

4. Sections 1, 5, 6, 7, and 8 survive termination of this Public License.

**Section 7 – Other Terms and Conditions.**

1. The Licensor shall not be bound by any additional or different terms or conditions communicated by You unless expressly agreed.

2. Any arrangements, understandings, or agreements regarding the Licensed Material not stated herein are separate from and independent of the terms and conditions of this Public License.

**Section 8 – Interpretation.**

1. For the avoidance of doubt, this Public License does not, and shall not be interpreted to, reduce, limit, restrict, or impose conditions on any use of the Licensed Material that could lawfully be made without permission under this Public License.

2. To the extent possible, if any provision of this Public License is deemed unenforceable, it shall be automatically reformed to the minimum extent necessary to make it enforceable. If the provision cannot be reformed, it shall be severed from this Public License without affecting the enforceability of the remaining terms and conditions.

3. No term or condition of this Public License will be waived and no failure to comply consented to unless expressly agreed to by the Licensor.

4. Nothing in this Public License constitutes or may be interpreted as a limitation upon, or waiver of, any privileges and immunities that apply to the Licensor or You, including from the legal processes of any jurisdiction or authority.

Creative Commons is not a party to its public licenses. Notwithstanding, Creative Commons may elect to apply one of its public licenses to material it publishes and in those instances will be considered the "Licensor." The text of the Creative Commons public licenses is dedicated to the public domain under the CC0 Public Domain Dedication. Except for the limited purpose of indicating that material is shared under a Creative Commons public license or as otherwise permitted by the Creative Commons policies published at creativecommons.org/policies, Creative Commons

does not authorize the use of the trademark "Creative Commons" or any other trademark or logo of Creative Commons without its prior written consent including, without limitation, in connection with any unauthorized modifications to any of its public licenses or any other arrangements, understandings, or agreements concerning use of licensed material. For the avoidance of doubt, this paragraph does not form part of the public licenses.

Creative Commons may be contacted at creativecommons.org.

Additional languages available: Bahasa Indonesia, Nederlands, norsk, suomeksi, etc. Please read the FAQ for more information about official translations.