Bioinformatics Advance Access published March 3, 2005

Fast tandem mass spectra–based protein identification regardless of the number of spectra or potential modifications examined.

Jayson Falkner and Philip Andrews[†]

[†]Dept. of Biological Chemistry and Program in Bioinformatics, University of Michigan, 1301 Catherine St., Ann Arbor, MI 48109

Abstract

Motivation: Comparing tandem mass spectra (MSMS) against a known data set of protein sequences is a common method for identifying unknown proteins; however, the processing of tandem mass spectra by current software often limits certain applications, including comprehensive coverage of post–translational modifications, non–specific searches, and real–time searches to allow result–dependent instrument control. This problem deserves attention as new mass spectrometers provide the ability for higher throughput and as known protein data sets rapidly grow in size. New software algorithms need to be devised in order to address the performance issues of conventional MSMS protein data set–based protein identification.

Methods: This manuscript describes a novel algorithm based on converting a collection of monoisotopic, centroided spectra to a new data structure, named "Peptide Finite State Machine", which may be used to rapidly search a known data set of protein sequences, regardless of the number of spectra searched or the number of potential modifications examined. The algorithm is verified using a set of commercially available tryptic digest protein standards analyzed using an ABI 4700 MALDI TOFTOF mass spectrometer, and a free, open–source Peptide Finite State Machine implementation. It is illustrated that a Peptide Finite State Machine can accurately search large collections of spectra against large data sets of protein sequences (e.g. NCBI nr) using a regular desktop PC; however, this paper only details the method for identifying peptide and subsequently protein candidates from a data set of known protein sequences. The concept of using a Peptide Finite State Machine as a peptide pre–screening technique for MSMS–based search engines is validated by using Peptide Finite State Machines with Mascot and XTandem.

Availability: Complete source–code, documentation, and examples for the reference Peptide Finite State Machine implementation are publicly available at the Proteome Commons, http://www.proteomecommons.org, and source–code may be used both commercially and non–commercially as long as the original authors are credited for their work.

Contact: jfalkner@umich.edu

1 Introduction

Tandem mass spectra–based protein identification using unique combinations of derived peptide (usually tryptic) sequences, each representing a fragment of a parent protein, is a common practice and it is sometimes referred to as bottom–up proteomics. The process relies on software packages built around tandem mass spectra (MSMS) database search and de novo peptide generation algorithms to identify both potential candidate peptides for a particular spectrum and, ultimately, a source protein that can account for the observed peptides. Several software packages currently exist (Clauser, Craig, Eng, Mann, Perkins, Tabb), but the software's application is often limiting due to the time it takes to perform a search on a large number of spectra or against a large data set of known proteins or due to the inability to accommodate multiple potential residue modifications. This manuscript describes a remedy to these issues by means of a novel data structure, a "Peptide Finite State Machine", abbreviated PFSM, and an algorithm for generation of the proposed data structure. A free, open–source Peptide Finite State Machine implementation is available, as described at the beginning of the manuscript, and it is used in all examples.

The main focus of this manuscript is to illustrate a method of quickly identifying candidate peptide sequences that match MSMS spectra. The data structure described by this manuscript provides a method of representing a monoisotopic, centroided spectrum (i.e "peak list") as a graph, where nodes represent peaks and arcs represent valid amino acid residues (with or without modification) that account for the distance between peaks. Initial construction of the data structure is near identical to that of a *spectrum graph* (Bartels, 1990), also see the SHERENGA paper (Dancik 1999) for an excellent introduction to tandem mass spectra and spectrum graphs; however, the finished data structure requires a conversion of the spectrum graph to a Non–Deterministic Finite Automata (NDFA) and subsequently a Deterministic Finite Automata (DFA) (Sipser 1996). The process of creating a DFA that represents a collection of spectra is both novel and incredibly practical because it allows one to screen amino acid sequences at a rate that is at worst as slow as looking at one state in the DFA for each amino acid in the sequence in the protein sequence. If one attempts the same process with a data structure similar to a spectrum graph, i.e. equivalent to a NDFA, the time it takes to screen an amino acid sequence can be at worst the amount of time it takes to examine every state in the entire graph and at best the same amount of time a DFA takes to examine the sequence. A simple, helpful analogy to individuals experienced in computer science is that a Peptide Finite State Machine provides a way to convert spectra to a single regular expression.

The entire thrust of this paper is succinctly summarized in section 2.4 and by reference to figure 5b. A Peptide Finite State Machine creates a DFA similar to figure 5b, and the DFA allows one to quickly, accurately determine if a peptide sequence might account for a spectrum as illustrated in section 2.4.

The rest of this manuscript is divided into two main sections. The first section details the precise algorithm for generation of a Peptide Finite State Machine from an arbitrary set of spectra. The second section of the manuscript demonstrates the feasibility of Peptide Finite State Machines as a tool to perform fast, sensitive protein identification using tandem mass spectra from a set of known proteins. All examples of Peptide Finite State Machines are uses of the Peptide Finite State Machine project which may be freely obtained with full source–code from the Proteome Commons, http://www.proteomecommons.org.

2 Algorithm

The Peptide Finite State Machine data structure is a NDFA and a DFA conversion of the NDFA, both of which represent any number of spectra and peptide candidates that may account for each spectrum. Both NDFA and DFA are common computer science concepts associated with the theory of computability or the Church–Turing thesis (Turing 1936, Church 1936), and the following section recapitulates the fundamentals.

A NDFA is a computer science concept associated with the expressive power of a language and it represents a finite number of states, each state linked to any of the other states by any number of transitions. Figure 5a illustrates the concept as a graph with a fixed number of nodes and arcs between nodes, each arc labeled with a single letter. The letters represent valid symbols in the alphabet that makes the language, in this paper's case the complete alphabet is the set of single–letter amino acid residue names. All NDFA have one start state, the solid black dot, and at least one finish state, the white–black bullseye. Arrows on arcs represent the direction that transitions proceed. If there is no arrow for a letter, assume the transition goes to a state that will never reach a finish state. To use a NDFA start with a sequence of symbols, e.g. a sequence of amino acid residues, and transition from state to state by sequentially using symbols from the sequence. At the end of the sequence, if one is on a finish state, the sequence is valid for the NDFA, if not, the sequence is invalid. In cases where a node has two or more transitions for a particular symbol follow all of the transitions simultaneously.

A DFA is equally as powerful as a NDFA in expressive power and is near identical in structure and use but differs by allowing one and only transition for every symbol in the alphabet for a given state. This difference makes a DFA more time efficient when implemented by a computer program because the program will never have to follow more than one transition per symbol in a sequence. More formally, this time efficiency is noted as $O(n)$ (Sipser 1996) where n is the length of the sequence being analyzed. A well–known algorithm exists for converting a NDFA to a DFA (Sipser 1996), and a succinct example of tracing amino acid sequences through a DFA is provided in section 2.4.

A Peptide Finite State Machine is a convenient name for describing a formal conversion from a set of mass spectra to a DFA, and one of the primary points of this manuscript is to illustrate how one may convert a spectrum to a NDFA. As further explained later, a spectrum can't be directly converted to a DFA if one accounts for mass accuracy errors of a mass spectrometer and if potential modifications of the standard amino acid residues are allowed. But a NDFA may be directly created from a spectrum. By creating the NDFA a DFA can be made, and with a DFA one can search through sets of protein sequences, such as any FASTA file, very quickly. Additionally, as also detailed later, two or more NDFA may be combined into a single NDFA and potential modifications may be accounted for when constructing a NDFA from a spectrum – this allows for a Peptide Finite State Machine to search a set of amino acid sequences in the same amount of time regardless of the number of spectra or the number of potential modifications being examined and with equivalent sensitivity as if searching each spectrum individually.

2.1 Conversion of a single tandem mass spectrometry peak list to a NDFA

Any tandem mass spectrometry peak list may be converted to a NDFA by assuming an arbitrary mass error tolerance for the accuracy of the peaks and by assuming an arbitrary tolerance for peaks not represented in the peak list but that should be present in a complete ion series. A tandem mass spectrum contains the peptide fragment ions observed at a given mass over charge ratio. It has been repeatedly demonstrated and well–reviewed that common peptide fragment patterns are exhibited by peptides being fragmented via tandem mass spectrometry (Biemann 1988, Papayannopoulos 1995, Roepstorff 1984). These peptide fragmentation patterns are commonly used to relate unknown peaks from a peak list to a peptide ion fragment in a mass spectrum. A progression of such identified peaks due to one peptide fragmentation pathway but separated by the atomic mass of an amino acid residue (or a number of residues) is referred to as an ion series for the observed peptide fragmentation pathway. A complete ion series is a set of peaks that span across an entire spectrum with a separation of a single residue between each peak. Given a complete ion series, identification of the source peptide is as simple as examining the ion series from start to finish and associating each peak to a sequential residue in the peptide by correlating the mass difference between peaks with the calculated residue masses.

Conversion of a peak list to a NDFA follows the assumption that ion series will be present in any given spectrum, and a complete ion series accurately represents a peptide that could fragment to produce the observed tandem mass spectrum, i.e. conversion is the same technique utilized to construct a spectrum graph (Bartels 1990, Dancik 1999). Construction of a Peptide Finite State Machine is started by the potential ion series in an observed peak list. For clarity, ion series will be represented in this paper by drawing the ion series as a

b–ion series, where peptide sequence may be obtained by reading off peak distances from left to right, assuming the left represents the N–terminus and the right represents the C–terminus. Figure 1a illustrates a b–ion series, and figure 1b illustrates how a y–ion series is converted to a b–ion series for illustration. In general, any ion series may be converted to a b–ion series by calculating the difference between the ion types while allowing for the mass accuracy of the mass spectrometer (e.g. bIonMZ = parentIonMZ - yIonMZ +1; see the Lutefisk97 paper's introduction (Taylor 1997) for an excellent example of the general technique). Conversion of a peak list to a NDFA is accomplished by tracing every valid ion series and converting it to a NDFA with states representing peaks and arcs representing distances between peaks that correspond to known residue mass over charge ratios. Figure 2a illustrates a peak list of b–ions and y–ions that have been converted to a NDFA, which is illustrated in 2b. Note that the conversion must be to a NDFA and not directly to a DFA because different ion series may result in multiple different but valid peptide sequences based on mass accuracy error of a mass spectrometer or when considering modifications of a standard residue, e.g. carboxyamidomethylation of a cysteinyl residue (an increase of 57 Da) corresponds to the m/z of a cysteinyl residue plus a glycyl residue (57 Da), which may result in a NDFA with two valid transitions for the C symbol (cysteinyl residue) as illustrated in figure 2b.

Tandem mass spectra–derived peak lists generated by analysis software do not normally come with the ion types identified, and multiple, incomplete ion series are almost always present in observed spectrum. Both of these complexities may be accounted for when constructing a Peptide Finite State Machine.

Incomplete ion series may be accounted for by allowing peaks separated by an arbitrary but identifiable m/z ratio to be considered part of an ion series. When constructing the NDFA from a peak list, missing residues in an ion series can be accounted for by using the technique in figure 3, identify a m/z that corresponds to a known combination of amino acid residues and insert all possibilities of the combination as transitions in the NDFA. This practice will introduce several combinations of possible amino acid sequences with likely only one being correct. It is demonstrated later in the manuscript, and assumed in general, that the inaccurate sequences can be distinguished from the correct sequence using a scoring function such as Mascot or XTandem.

Unidentifiable fragment ion types may be accounted for by assuming all peaks are of the ion type of interest. To examine an ion series of particular interest (e.g. b, y, a, c, x, or z), use the approach illustrated by figure 1 where the observed peaks are assumed to be a particular ion type and all of the peaks are converted to a b–ion series. If examining multiple ion series, say y–ions and b–ions, overlay the ion series as illustrated in figure 1c. This general technique

may also be applied for all the different ion types commonly observed via CID (Papayannopous 1995); however, this manuscript only focuses on most easily identifiable ion series, b– and y– ion series. Use of the technique in figure 1 may introduce inaccurate states in the subsequently generated NDFA, but the practice is required if ions are of an unknown type. It is also demonstrated later in the manuscript, and assumed in general, that inaccurate peptide sequences introduced from mislabeled ion series can be distinguished from the correct peptide sequence through use of a scoring function such as Mascot or XTandem.

It is suggested that sequences identified by a PFSM only be considered candidate sequences that may describe the observed peak list. Conversion of a tandem mass spectrum to a NDFA is viable, and an ideal tandem mass spectrum may be converted to a NDFA that will only identify correct sequences. However, tandem mass spectra often come with ambiguities that make conversion to an NDFA possible, using the techniques previously described, but at the expense of introducing erroneous sequences to the resulting NDFA. Any sequence identified by a PFSM should be further examined either by hand or with a suitable high–throughput scoring function. Two of such scoring functions are examined later in this manuscript.

2.2 Combining multiple peak lists into one NDFA

In many practical situations, multiple tandem mass spectra likely represent one protein (e.g. a single 2D gel plug spotted on a well of a MALDI plate) or a large collection of related peptides (e.g. a MuDPIT experiment). In these situations the peak lists generated from these spectra are often searched together in an attempt to identify the source protein(s). A Peptide Finite State Machine may represent more than one peak list and any number of related or unrelated peak lists may be multiplexed by combining all of the peak lists of interest into a single NDFA for the Peptide Finite State Machine.

By definition a NDFA state may transition to multiple states using a single symbol. This property allows a collection of NDFAs to be combined to a single NDFA by simply combining the start states. This concept is illustrated in figure 4, and it manifests itself in practical use by allowing a Peptide Finite State Machine to search for any number of peptides at the same time, without requiring multiple searches of the same amino acid sequence and with the guarantee that the same sequences will be identified as in a set of single spectrum searches. An entire set of spectra may be analyzed simultaneously by a Peptide Finite State Machine with just a single pass through a set of protein sequences. This feature is inherent in the conversion of a NDFA to a DFA, and it is a technique that is commonly used by regular expression matching algorithms.

2.3 Conversion of a NDFA to a DFA

A Peptide Finite State Machine requires both a NDFA, created by converting any number of peak lists, and a DFA that represents the NDFA. A well–established computer science algorithm exists for conversion of a NDFA to a DFA (Sipser 1996). The algorithm works by taking the set of states each transition in the NDFA can reach and treating it as a single state in a DFA. The concept is illustrated in Figure 5. The NDFA in figure 5a has two transitions from state 2, given the symbol C, therefore it can't be considered a DFA. In order to create a DFA from figure 5a, one starts by making a DFA state that represents the set of NDFA states that may be transitions from the NDFA start state, figure 5b's state {1}. Next, and for each subsequent DFA state created, a single DFA transition is made from an existing DFA state to another DFA state that represents the collection of NDFA states that may be transitioned to using a symbol from the alphabet, e.g. state {2} is the transition from state {1} given the symbol T and state {3,4} is the transition from state {2} given the symbol C. The process is repeated for every symbol in the NDFA's alphabet, and new DFA states are only created if the collection of NDFA states does not already exist. Once the entire NDFA has been converted, all DFA states that contain a finish state in the aggregate NDFA states are considered valid DFA finish states.

2.4 Peptide and protein identification through use of a Peptide Finite State Machine

The algorithm presented in this manuscript is not intended to be a stand alone system for identifying the best protein match for a particular MSMS spectrum; however, Peptide Finite State Machines are intended for use as a rapid method for identifying candidate peptide sequences that may account for a given MSMS spectrum. A list of such identified peptide sequences may then be sorted and ranked by an arbitrary scoring function in an attempt to correctly identify which sequence or sequences account for the observed MSMS spectrum. Additionally, a partial peptide sequence may be correlated with a full protein sequence in order to identify an unknown protein. This paper does not propose a formal scoring scheme for use with Peptide Finite State Machines, but the concept of using a Peptide Finite State Machine in conjunction with two well–known scoring schemes is validated in order to demonstrate that Peptide Finite State Machines are suitable as a foundation for building high–performance MSMS search engines

A Peptide Finite State Machine can rapidly identify valid sequences in a known data set of proteins by using the amino acid sequence(s) from the data set as input to the Peptide Finite State Machine's DFA. If at the end of a sequence the Peptide Finite State Machine's DFA is at the finish state, the sequence is a valid peptide that may have represented one of the observed peak lists. The peak list (s) itself may be identified by checking the collection of NDFA states that represents the DFA finish state. In cases where a DFA state was created from a set of NDFA finish states, the amino acid sequence represents a peptide that

may account for any of those peak lists.

As a simple example, assume the Peptide Finite State Machine DFA shown in figure 5b is being used on the tryptic fragments of the sequence "TCGKSGRTCKCAR". The tryptic fragments of the sequence, assuming no missed cleavages, would be "TCGK", "SGR", "TCK", and "CAR". Using the sequence "TCGK" or "TCK" as input to the Peptide Finite State Machine's DFA would result in the DFA reaching the finish state, meaning those amino acid sequences could represent two of the peak lists or the same peak list used to make the Peptide Finite State Machine; however, the sequences "SGR" and "CAR" would not end in the finish state of the Peptide Finite State Machine's DFA and therefore are not valid peptide sequences according to the mass accuracy error and amino acids and amino acid modifications used to construct the Peptide Finite State Machine.

Given a Peptide Finite State Machine, one may quickly examine a known set of peptides (e.g. a theoretical digest of proteins from a known database) and determine if the peptides could or could not account for an observed spectrum. Twenty commercially available tryptic digest standards (Michrom Bioresources Inc., Auburn, CA) were analyzed in order to verify that Peptide Finite State Machines could correctly identify peptide sequences. 50 fMol of each protein was individually spotted on a MALDI plate with an alpha–cyano–4–hydroxycinnamic acid matrix. Data were acquired using an ABI 4700 MALDI TOFTOF by obtaining tandem mass spectra of the ten most intense MS peaks and Mascot Distiller (Matrix Science, Boston, MA) was used to generate the appropriate peak list for each tandem mass spectrum. A Peptide Finite State Machine representing all the MSMS data was generated, assuming a mass error tolerance of 200 ppm and a potential gap of at most 3 sequential amino acids in an ion series, and the protein sequences for each of the 20 proteins was analyzed. For each protein multiple peptide sequences were identified that correlated well to both the observed MSMS spectrum.

Table 1 summarizes the results of the PFSM analysis of the known protein sequences. Three different numbers are shown for the PFSM that represent a PFSM constructed to bridge gaps in an observed ion series of up to 1, 2, and 3 residues. The numbers for Mascot and XTandem represent the number of peptides they correctly identified. Comparison against the existing search engines should not be taken as a literal measurement of sensitivity. The set of protein sequences searched is contrived to contain the only the known sequences. The comparision is intended to illustrate that a PFSM can identify the same candidate peptide sequences (not just one or two pristine MSMS fragmentations per protein), enforcing that a PFSM is an accurate representation of observed MSMS spectra.

3 Implementation

3.1 Performance Analysis of the Peptide Finite State Machine

In theory a Peptide Finite State Machine will search through any protein sequence in time directly proportional to the size of the sequence; "search" means transition from state to state in a DFA, which assuming one has already constructed the DFA, takes no more time regardless of the number of peak lists multiplexed or the number of potential modifications used to create the initial NDFA. However, there will always be the practical limitations of the amount of time and computer memory required in order to create the Peptide Finite State Machine. These limitations are addressed by this section with the common use case of searching a collection of peak lists against a collection of protein sequences. In all examples the Peptide Finite State Machine project from the Proteome Commons, http://www.proteomecommons.org, is used on a Pentium 4 2.6GHz with 1GB of RAM.

The time required to construct a Peptide Finite State Machine from a collection of peak lists and the time required to search through a set of protein sequences is illustrated in figure 6. The example implementation constructs the complete NDFA and then proceeds to create DFA states as they are needed during the search. The implementation's approach is modeled after the popular, open–source regular expression tool grep (Free Software Foundation). Search time is illustrated in figure 6a and it is split into two parts representing the time associated with constructing the DFA and the time required to search through protein sequences. Figure 6b illustrates the same data represented in search time per peak list.

The searches in figure 6 are performed using peak lists obtained from an ABI 4700 MALDI TOFTOF that was used to collect spectra from commercially available tryptic digest standards (Michrom Bioresources Inc., Auburn, CA). The top ten MS ions of each digest were analyzed using MSMS, and each protein was analyzed in triplicate without ignoring previously analyzed MS ions. 682 MSMS spectra were collected with approximately 227 unique spectra identified. Individual peak lists were generated using the Mascot Distiller software package (Matrix Science Inc., Boston, MA). The spectra and peak lists are freely available in the Michrom tryptic digest spectra and peak list projects at http://www.proteomecommons.org. Searches are performed against the tryptic peptide sequences from the August 2004 NCBI nr release. Tryptic sequences are selected by performing a theoretical trypsin digest of each FASTA entry, allowing up to two missed cleavages. Sequences between 8 and 30 residues long were considered, including c–terminal peptides. In total just over 72 million tryptic peptides, approximately a trillion residues, were searched. In all cases, the searches done in figure 6 identified the tryptic peptide sequences of the known proteins as candidate sequences. The Mascot search results are

freely available and may be obtained in the aforementioned Michrom tryptic digest peak list project.

The time requirements shown in figure 6 illustrate a few points. The search time for a Peptide Finite State Machine stays reasonably consistent as more peak lists are multiplexed. Clearly the time to multiplex and search is much less than the time required to search each peak list individually; however, search time does slightly increase with the number of peak lists multiplexed, presumably due to the implementation of the algorithm. The limiting factor is clearly the time required to construct the Peptide Finite State Machine, which does grow with the number of peak lists multiplexed, albeit if the time growth is relatively minuscule.

Figure 6c illustrates the memory requirements derived from multiplexing the peak lists for each search. The Peptide Finite State Machine in each search is performed with a reasonable set of potential modifications, which includes acetylation, carboxyamidomethylation, deamidation, oxidation, and pyroglutamyl from either glutamic acid or glutamine. The Peptide Finite State Machine in each search is also constructed with allowance of three missed ions in a series and a mass accuracy of 200 ppm. The memory requirements are reasonable, allowing up to 1000 peak lists to be loaded in little over 100MB of computer memory. Also depicted in figure 6c is the change in memory requirements as a variable number of potential modifications are considered. Multiple modifications may be used; however, adding more modifications is analogous to multiplexing more peak lists. Potential modifications will result in longer construction time of the Peptide Finite State Machine's NDFA and DFA and more memory requirements will increase.

It is important to realize that the information in Figure 6 is only illustrating the time and memory requirements for using a Peptide Finite State Machine to identify candidate peptide sequences. There is no attempt to rank candidate sequences in order to claim one peptide is a more suitable match for a particular MSMS spectrum, nor is there any attempt to identify the most likely protein from which the peptide sequences originated. In order to identify the most likely peptide sequences and the originating proteins a scoring function is required. Figure 7 illustrates that a Peptide Finite State Machine may be combined with existing scoring functions, Mascot (Matrix Science Inc., Boston, MA) and XTandem (Craig 2004), in order to provide more rapid peptide and protein identifications. Both software packages may search against FASTA files; however, figure 7 illustrates that significant time improvements are obtained when using a Peptide Finite State Machine to first identify candidate sequences and subsequently using the search engine to analyze the subset of FASTA entries that contained candidate sequences. In the small case of analyzing 10 spectra obtained from a single MALDI well, e.g. one protein, the candidate sequences identified by the Peptide Finite State Machine belong to 142 of the 2

million FASTA entries in NCBI nr. Searching Mascot or XTandem against the 142 entry FASTA file identified the same, correct entry that is identified by searching all of NBCI nr with Mascot or XTandem, respectively. The time difference is slightly favorable for using the Peptide Finite State Machine with either of the search engines instead of search engine on its own. When searching all 682 peak lists the Peptide Finite State Machine identifies 12,634 FASTA entires from NCBI nr, and using the Peptide Finite State Machine with either search engine saves a significant amount of time compared to using the search engine by itself.

Clearly Peptide Finite State Machines do provide a mechanism for rapidly searching protein sequences while considering large amounts of peak lists, optionally considering potential modifications. The time and memory requirements of a Peptide Finite State Machine, specifically the example implementation, are reasonable and allow a moderately fast computer to rapidly analyze large amounts of MSMS data against large data sets. The Peptide Finite State Machine by itself only identifies candidate peptide sequences, but it was shown that a Peptide Finite State Machine can be combined with a conventional MSMS search engine in order to more rapidly identify and rank candidate peptide sequences.

4 Summary and future directions

The Peptide Finite State Machine data structure is a novel, helpful tool for proteomics MSMS data analysis. The most beneficial aspect of a Peptide Finite State Machine is the regular expression–like approach that allows any number of peak lists to simultaneously be searched against a set of protein sequences. Multiplexing in such a fashion does not change the number of peptides that will be identified and it requires much less time compared to searching each peak list individually against the entire data set. Detailed in this manuscript is the complete algorithm for generating a Peptide Finite State Machine from a set of peak lists and examples of using a Peptide Finite State Machine to identify amino acid sequences that may account for an observed peak list(s).

Clearly an important future direction for Peptide Finite State Machines is an analysis which includes using a Peptide Finite State Machine's DFA for rapid sequence identification while scoring candidate sequences in order to determine the best peptide sequence match(es) for a given spectrum, especially since Peptide Finite State Machines may easily account for multiple potential modifications. Another important future direction for Peptide Finite State Machines is that of result dependent data acquisition and real–time MSMS data analysis. These topics were not explored by this paper, but upon examination of the performance characteristics of Peptide Finite State Machines (figure 6 and 7) it should be clear that candidate peptide sequences can be identified in near–real time using a modest computer. Given a smaller data set to examine (NCBI nr is one of the largest data sets available) or more computational power,

either a cluster of computers or an expensive server, it is reasonable to attempt using Peptide Finite State Machines to complement real–time MSMS data acquisition. A combination of a high–performance scoring function, a Peptide Finite State Machine and an interface with MSMS instrumentation is a promising avenue of future research.

Acknowledgments

Special thanks to Angela Walker for her assistance with the ABI 4700 MALDI TOFTOF and data collection, Edward Barbour for his assistance with Mascot Distiller, and to everyone else in the Andrews lab for providing a great place to learn and experiment.

Special thanks to Tom Blackwell, Pete Ulintz, Gary Rymar, Catherine Grasso, and Eric Simon for their accurate comments, prompt feedback, and time and effort spent on editing this manuscript.

Support from the NCRR for the National Resource for Proteomics and Pathways is gratefully acknowledged.

References

Bartels, C., "Fast algorithm for peptide sequencing by mass spectroscopy", Biomed. Environ. Mass Spectrom. 19, 363-368, 1990.

Biemann, K., "Contributions of Mass Spectrometry to Peptide and Protein Structure", Biomedical and Environmental Mass Spectrometry, Vol. 16, 99-111, 1988.

Church, A., "A Note on the Entscheidungsproblem", Journal of Symbolic Logic, 1, 40-41, 1936.

Clauser, K.R., et al., "Role of Accurate Mass Measurement (±10 ppm) in Protein Identification Strategies Employing MS or MS/MS and Database Searching", Anal. Chem. 71, 2871-2882, 1999.

Craig R. Beavis R.C., "TANDEM: matching proteins with tandem mass spectra." Bioinformatics, 2004.

Dancik, V., et al., "De Novo Peptide Sequencing via Tandem Mass Spectrometry", Journal of Computational Biology, Volume 6, Numbers 3/4, 1999.

Eng J, McCormack AL, Yates JR., "An approach to correlate tandem mass spectral data of peptides with amino acid sequences in a protein database." J

Am Mass Spectrom. 5:976-989, 1994.

Ducret A, et al., "High throughput protein characterization by automated reverse–phase chromatography/electrospray tandem mass spectrometry." Protein Sci. Mar;7(3):706-19, 1998.

Free Software Foundation, "Grep", http://www.gnu.org/software/grep/grep.html

Mann M., Wilm M., "Error–tolerant identification of peptides in sequence databases by peptide sequence tags", Anal Chem. Dec 15;66(24):4390-9, 1994.

Papayannopoulos, IA, "The interpretation of collision–induced dissociation tandem mass spectra of peptides." Mass Spectrom. Rev. 14(1) 49-73, 1995.

Perkins, D.N, et al., "Probability–based protein identification by searching sequence databases using mass spectrometry data.", Electrophoresis. Dec;20 (18):3551-67, 1999.

Roepstorff, P and Fohlman, J., "Proposal for a common nomenclature for sequence ions in a mass spectra of peptides." Biomed Mass Spectrom. 11(11) 601, 1984.

Sipser, M., "Introduction to the Theory of Computation", Brooks Cole; 1st Edition, ISBN 053494728X, 54-58, 225-226, 1996.

Tabb DL, et al., "GutenTag: High–throughput sequence tagging via an empirically derived fragmentation model." Anal Chem. Dec 1;75(23):6415-21, 2003.

Taylor, J.A, and Johnson, R.S., "Sequence Database Searches via de Novo Peptide Sequencing by Tandem Mass Spectrometry", Rap. Comm. in Mass Spec., Vol. 11 1067-1075, 1997.

Turing, A.M., "On Computable Numbers, with an Application to the Entscheidungs problem", Proceedings of the London Mathematical Society, series 2, 42 (1936-37), 230-265, 1936.

Table 1: PFSM peptide identifications of known proteins.

	PFSM (1 gap)	***PFSM(2 gap)***	***PFSM (3 gap)***	***Mascot***	***XTandem***
Correct Peptides	16	153	196	121	178

Peptide Finite State Machines were constructed from approximately 227 unique spectra of the 20 known proteins (preparation and analysis performed as stated in manuscript) and compared against Mascot and XTandem. Three different PFSM were constructed with a mass accuracy tolerance of 200ppm and with respective max gaps in ion series of up to 1, 2, and 3 residues. All searches were performed against the 20 known protein sequences, and search parameters were kept as identical as possible. The correct peptides identified is the summation of all of the peptide sequences matched from each protein's MSMS spectra that reasonably explained a tryptic peptide in that protein's sequence. Results were verified by hand.

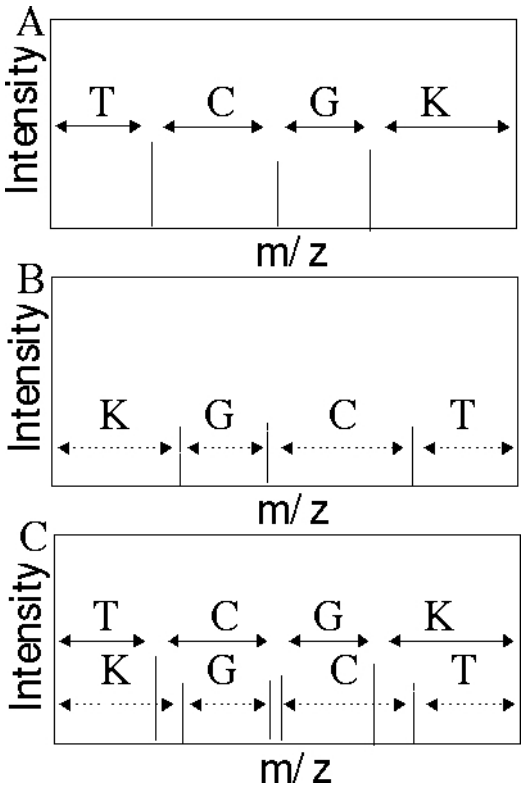

Figure 1: (A) a peak list assumed to be a b–ion series where each m/z gap between ions in the series is matched to an amino acid residue. (B) the same peak list assumed to be a y–ion series and draw as the conversion of a y-ion series to a b–ion series(i.e. parentIonMZ – yIonMZ +1). (C) The same peak list but assuming the ion series may be either a y–ion or a b–ion series, i.e. overlaying both possibilities.

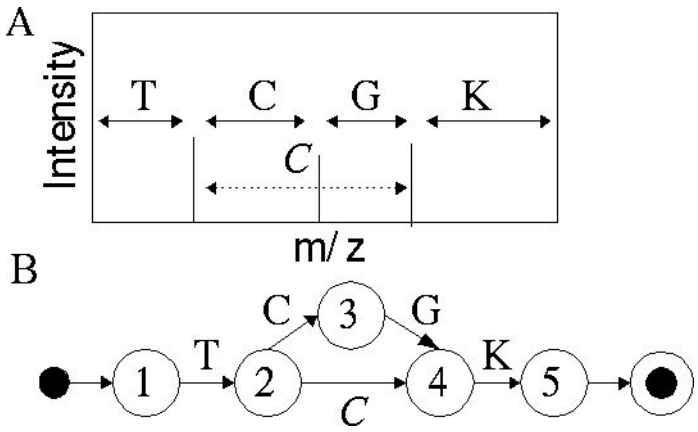

Figure 2: (A) A peak list with an assumed b–ion series and annotated by one letter amino acid residues that match m/z differences between sequential ions. Assume the *C* above the dotted line is a cysteinyl residue that has been Carboxyamidomethylated and corresponds to the same m/z of C+G, therefore bridging the ion series directly from T to K. (B) The NDFA conversion of the annotated peak list. Note that there are two routes from T to K, either CG or *C*. The CG route represents following the unmodified C with a subsequent G. The *C*–only route represents a Cyseinyl residue that has been Carboxyamidomethylat ed to have m/z of the unmodified CG pair. This is why a peak list can not be converted directly to a DFA – there may be two or more valid m/z differences between ions that use the same residue, either due to a potential modification or mass accuracy errors.

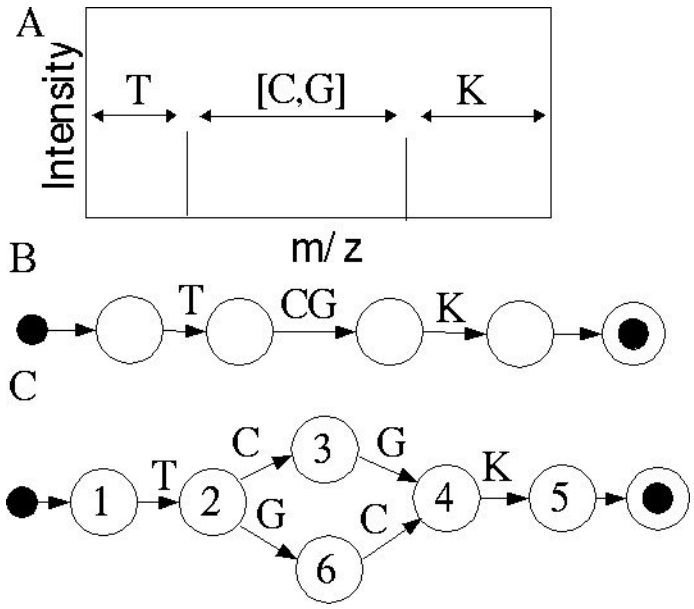

Figure 3 (A) A peak list similar to figure 2, but missing the second ion in the figure 2's ion series. The m/z difference is annotated with [C,G] because it is assumed that the m/z of adding the two residues gives the appropriate m/z to bridge the ion series. (B) The graph conversion of the given peak list allowing arcs to have multiple residues (e.g. [C,G]) instead of just one residue as in figure 2. (C) The NDFA conversion of the graph in part B, illustrating how to convert multiple residue to a single residue NDFA transitions – i.e. how to allow for ions in a series that aren't represented in the peak list. The solution is to create all combinations of the residues in the NDFA.

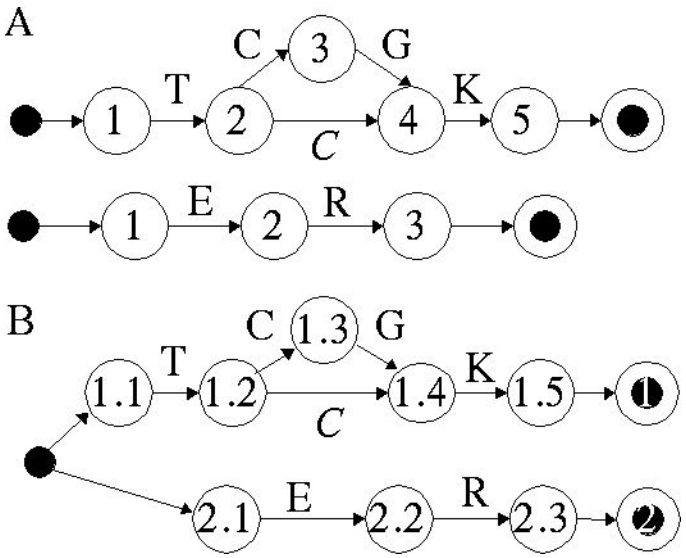

Figure 4: (A) Two NDFAs that are assumed to have been created from different peak lists. (B) A single NDFA that represents both the original NDFA, e.g. multiplex of the peak lists. Note the multiplex represents the union sequences each NDFA accepts. Sensitivity of a PFSM will not change due to multiplexing NDFAs, and it is not equivalent to combining multiple peak lists before generating an NDFA.

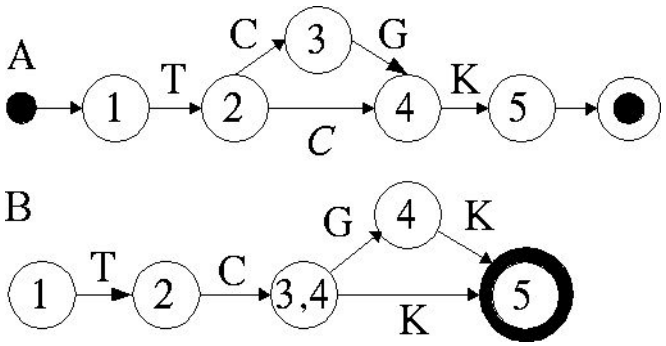

Figure 5: (A) The NDFA from figure 2 that is converted to a DFA (B). An NDFA may have multiple transitions for a single symbol (residue abbreviation) from a single state, note 2 has two transitions for C. A DFA must have exactly one transition for each symbol. (B) The DFA equivalent of A.. By modifying the graph in A it is possible to create a proper DFA with exactly one transition per symbol. This is the key to the performance benefit of PFSMs. If left as a NDFA, checking a residue sequence might require searching through multiple transitions per residue, in the worst case searching the entire graph; however, if searching with a DFA, there will always be exactly one transition per residue regardless of the number of peak lists multiplexed or potential modifications used to make the graph.

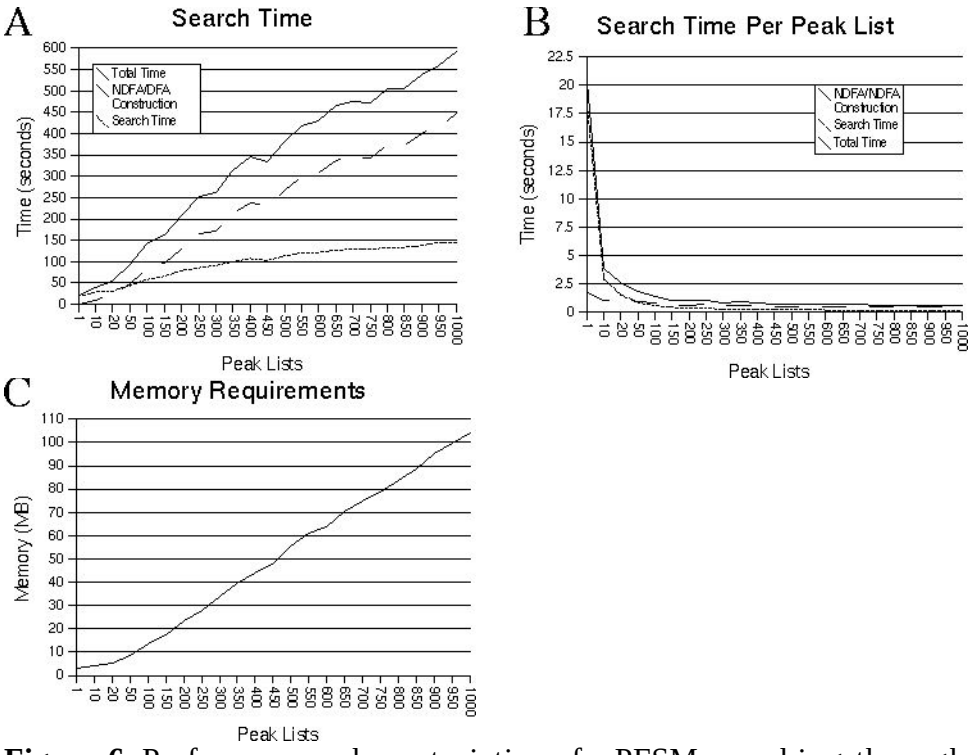

Figure 6: Performance characteristics of a PFSM searching through the tryptic peptides of NCBI nr, August 2004 release (~79 tryptic peptides, ~1 trillion residues). The time required to perform the search is shown in (A) and (B). The solid line is the complete program execution time, and the dashed and dotted lines represent the time required to make the PFSM and the time required to search, respectively. (C) illustrates the minimal memory requirements in order to search.

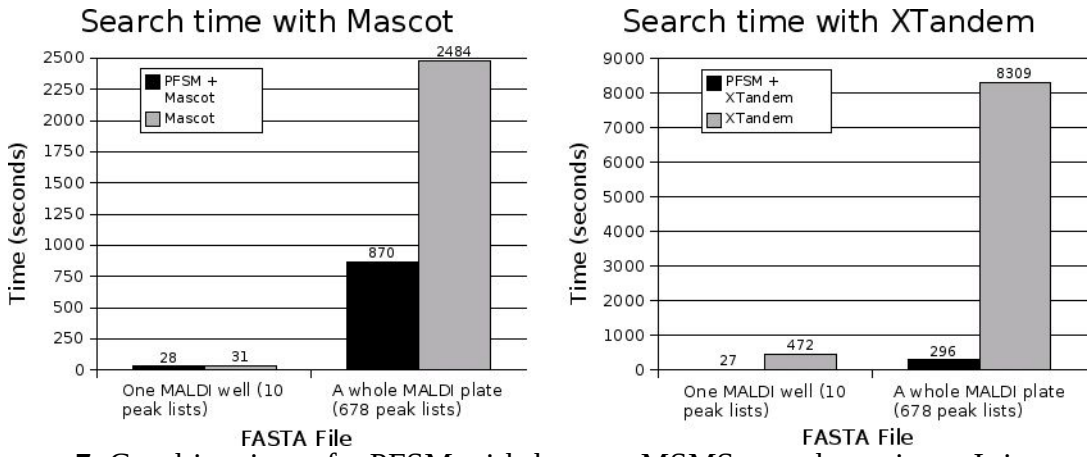

Figure 7: Combination of a PFSM with known MSMS search engines. It is shown that time can be saved by first using a PFSM to screen for candidate sequences and then scoring only those sequences. Time comparison is shown for searching a PFSM+Mascot or XTandem (black) against NCBI nr and for searching Mascot or XTandem (gray) directly against nr. Time measured includes total execution time of both algorithms in the cases of PFSM+Mascot and PFSM+XTandem. Known tryptic standards are used and the search results in both cases identify the same peptides. The Mascot and XTandem searches were performed with the same data but on different computers.