Universidade Federal do Rio Grande do Norte
Centro de Ciências Exatas e da Terra
Departmento de Informática e Matemática Aplicada
Disciplina de Linguagens Formais e Autômatos

Análise do uso de RegEx na biblioteca Glycowork

Andriel Vinicius de Medeiros Fernandes,

Gabrielle de Vasconcelos Borja,

Jeremias Pinheiro de Araújo Andrade,

Lucas Vinicius Dantas de Medeiros,

María Paz Marcato,

Ramon Cândido Jales de Barros

Natal-RN

Outubro, 2025

Nome completo do autor

Análise de uso de RegEx na biblioteca Glycowork

Pesquisa elaborada e apresentada para a disciplina DIM0606 - Linguagens Formais e Autômatos, ofertada pelo Departamento de Informática e Matemática Aplicada da Universidade Federal do Rio Grande do Norte e ministrada no semestre 2025.2 pelo Prof. Dr. Valdigleis da Silva Costa, como requisito parcial para a obtenção de nota para a 1ª unidade.

DIMAp – Departamento de Informática e Matemática Aplicada
CCET – Centro de Ciências Exatas e da Terra
UFRN – Universidade Federal do Rio Grande do Norte

Natal-RN

Outubro, 2025

Análise do uso de RegEx na biblioteca Glycowork

Autores: Andriel Vinicius de Medeiros Fernandes,
Gabrielle de Vasconcelos Borja,
Jeremias Pinheiro de Araújo Andrade,
Lucas Vinicius Dantas de Medeiros,
María Paz Marcato,
Ramon Cândido Jales de Barros
Professor: Valdigleis

Resumo

O resumo deve apresentar de forma concisa os pontos relevantes de um texto, fornecendo uma visão rápida e clara do conteúdo e das conclusões do trabalho. O texto, redigido na forma impessoal do verbo, é constituído de uma seqüência de frases concisas e objetivas e não de uma simples enumeração de tópicos, não ultrapassando 500 palavras, seguido, logo abaixo, das palavras representativas do conteúdo do trabalho, isto é, palavras-chave e/ou descritores. Por fim, deve-se evitar, na redação do resumo, o uso de parágrafos (em geral resumos são escritos em parágrafo único), bem como de fórmulas, equações, diagramas e símbolos, optando-se, quando necessário, pela transcrição na forma extensa, além de não incluir citações bibliográficas.

Palavras-chave: Palavra-chave 1, Palavra-chave 2, Palavra-chave 3.

Lista de figuras

Lista de tabelas

Lista de abreviaturas e siglas

ReGex – Expressões Regulares

Sumário

1 Referencial Teórico

O trabalho feito na Universidade de Gotemburgo, na Suéćia, tem como bases duas áreas distintas, a biologia e a ciência da computação. Na área biológica, o foco é sobre os glicanos, biopolímeros com inúmeras aplicações e funções biológicas. Já na computação, foi estudado o uso de expressões regulares (RegEx), padrões formados por sequências de caracteres utilizados para busca, análise e manipulação de textos. Dessa forma, os pesquisadores aplicaram expressões regulares para formalizar uma estrutura de busca voltada à identificação de padrões específicos dos glicanos.

1.1 Glicanos: Complexidade e Importância Biológica

Para compreender melhor a abordagem biológica do estudo, é importante explorar os glicanos, suas características e funções, bem como outros conceitos relacionados abordados no trabalho.

Definição 1.1.1 (Definição de Glicanos)**.** *Glicanos são polissacarídeos estruturais, longas cadeias formadas por unidades de açúcar (monossacarídeos) ligadas entre si por ligações glicosídicas, presentes abundantemente na Terra como componentes importantes de estruturas como glicoproteínas, glicolipídeos e proteoglicanos, além de fazerem parte de paredes celulares de fungos e leveduras.*

Essa diversidade de funções ocorre porque, em sua composição, existem subestruturas chamadas de motivos de glicanos, uma sequência ou arranjo particular de açúcares que funciona como um sinal de reconhecimento molecular, ou seja, a parte que carrega o significado mais importante e que é reconhecida por outras moléculas. São eles que trazem a importância biológica dos glicanos e, por isso, é de extrema importância que eles sejam entendidos pela ciência.

Contudo, a grande diversidade e versatilidade traz à estrutura interesse contínuo de pesquisas de diversas áreas, como a imunologia, biotecnologia e parasitologia. Porém,

como existem inúmeros motivos de glicanos, catalogá-los e reconhecê-los é um desafio.

1.2 RegEx na Ciência da Computação

Na ciência da computação, RegEx são uma sequência de caracteres que define um padrão de busca. Elas representam uma linguagem formal que pode ser utilizada para identificar, extrair e manipular subconjuntos de texto com base em regras e padrões específicos.

Essa notação envolve uma combinação de cadeias de símbolos do alfabeto da linguagem. Por exemplo, o operador main (+) é utilizado para denotar união, o asterisco (*) para o fecho estrela, indica zero ou mais ocorrências do caractere anterior, e o ponto (.) para concatenação. Essa sintaxe permite a criação de padrões que vão desde buscas simples, como encontrar uma palavra, até a validação de estruturas complexas em um texto.

1.3 A Aplicação Inovadora de RegEx em Glicanos

Diante do desafio na análise dos glicanos, o estudo propôs a aplicação de expressões regulares como ferramenta para buscar e extrair seus padrões estruturais de forma precisa e flexível. Essa técnica foi adaptada no trabalho para lidar com a estrutura ramificada e não linear dos glicanos, permitindo uma análise muito mais eficiente e escalável. O artigo demonstra que, embora as aplicações tradicionais de RegEx sejam conhecidas, sua versatilidade permite o uso em contextos menos convencionais, como a bioinformática.

2 Capítulo 2

Neste capítulo, descrevemos o funcionamento do sistema `Glycan RegEx` implementado no módulo `glycowork.motif.regex`, suas funções principais, o papel das expressões regulares e as vantagens do uso desse sistema.

O `Glycan RegEx`, introduzido por Bennett e Bojar (2024) no pacote `glycowork`, representa um avanço significativo na análise computacional de glicanos ao acrescentar o uso de RegEx para identificação e extração de motifs. Tal sistema foi proposto como uma adaptação das tradicionais RegEx de ciência da computação, permitindo a detecção de motifs na estrutura não linear dos glicanos, possibilitando buscas precisas e otimizadas nestes elementos.

2.1 Estrutura e funções do módulo

O `Glycan RegEx` se baseia na tradução de padrões RegEx para operações de isomorfismo de subgrafos dentro das estruturas moleculares de glicanos. Quando o usuário fornece um padrão o sistema decompõe esse padrão em unidades menores, chamadas de módulos homogêneos, correspondentes a monossacarídeos e ligações individuais. Cada módulo é então processado para identificar as possíveis correspondências no grafo do glicano, permitindo localizar subestruturas equivalentes de forma independente da forma textual em que o glicano foi representado. Isso é essencial, já que o módulo aceita diversos formatos de entrada, como GlycoCT, Oxford e demais notações.

Estas RegEx tem uma sintaxe muito similar às clássicas RegEx textuais mas a semântica é bastante diferente: como os glicanos não são simples cadeias de caracteres mas sim estruturas que podem ser representadas por grafos, o `Glycan RegEx` deve buscar não por símbolos (a, b, c) mas por monossacarídeos (GlcNac, Man) e suas ligações; como exemplo, buscar pelo motif $A\alpha 1-3B$, onde A e B são monossacarídeos quaisquer, requer buscar em qualquer ponto do grafo do glicano onde essa ligação $\alpha 1-3$ ocorre.

Desse modo, este modelo de RegEx permite suporte a modificadores, quantificadores, operadores de busca contextual e curingas mas com novos significados, como:

- (): representa uma ramificação/subárvore dentro do subgrafo. Exemplo: $Man(HexNAc)$, indicando um motif com uma unidade Man com uma ramificação $HexNAc$;
- []$*$: representa um motif que pode ocorrer zero ou mais vezes. Exemplo: $[HexNAc]*$;
- []+: ao menos uma repetição de uma unidade. Exemplo: $[HexNaC]$;
- : especifica a quantidade de ocorrências. Exemplo: $[HexNAc]2,$, indicando um motif com ao menos duas ocorrências deste monossacarídeo.

O funcionamento interno do `Glycan RegEx` ocorre, a princípio, com a segmentação da RegEx em módulos homogêneos: cada módulo é classificado como simples quando não há a presença de modificadores ou quantificadores ou complexo quando há; módulos simples são armazenados no formato de `string`, enquanto os complexos são salvos em dicionários. Em seguida, cada módulo é convertido em um grafo e por meio de operações de isomorfismo de subgrafos é detectado onde cada subgrafo se encaixa no glicano completo. Durante tal processo de detecção ocorre também a aplicação dos modificadores e quantificadores dos módulos complexos, o que permite descartar de forma otimizada subgrafos que não atinjam aos requisitos definidos, como número específico de ocorrências do padrão.

Ao fim, com o processamento de todos os módulos, ocorre a construção do caminho através do glicano que une todas as correspondências parciais e validadas pelos requisitos definidos na expressão completa reconstruindo, dessa forma, o motif exato a ser buscado. Essa abordagem possibilita representar ligações específicas, como α1-3 ou β1-6, bem como ambiguidade estrutural e ramificações expressas por meio de parênteses. Dessa forma, a notação `RegEx` é adaptada à topologia molecular, permitindo uma modelagem altamente expressiva e precisa das estruturas glicosídicasuscado.

Todo este processo é executado por diversas funções internas da biblioteca, dentre as quais se destacam:

- `preprocess_pattern()`: Particiona a RegEx fornecida em uma lista de padrões menores (módulos);
- `process_complex_pattern()`: Verifica por meio de isomorfismo de subgrafos se um padrão complexo (com modificadores/quantificadores) está presente no glicano;

- `process_simple_pattern()`: Verifica por meio de isomorfismo de subgrafos se um padrão simples (sem modificadores/quantificadores) está presente no glicano;
- `match_it_up()`: Para cada módulo, formata este em uma cadeia de caracteres ou dicionário, executa a função `process_complex_pattern()` ou `process_simple_pattern()` e retorna os módulos que representam os subgrafos que foram encontrados;
- `trace_path()`: Conecta cada subgrafo encontrado na estrutura do glicano e retorna o caminho completo;
- `get_match()`: Extrai trechos da estrutura do glicano que correspondem ao motif buscado pelo usuário, concentrando todo o processo por chamadas às demais funções.

2.2 Vantagens da RegEx no contexto do `glycowork`

Como comentado, a inovação trazida pelo `Glycan RegEx` ao framework `glycowork` consiste na união do método tradicional de busca por motifs com isomorfismo de subgrafo e o uso de um sistema de expressões regulares específico para glicanos.

Sem o uso das expressões regulares, há a necessidade de uma base estática de motifs já conhecidos. Essa base pode ser observada no arquivo `common_names.json` do próprio framework. Além disso, torna-se necessário o uso de múltiplas combinações de padrões para detectar motifs complexos e específicos do glicano, bem como lidar com a dificuldade de capturar contextos mais amplos, como a ocorrência de um motif presente apenas em um ramo do glicano. Todos esses fatores culminam em um uso bastante rígido e, por vezes, difícil de representar de forma computacional.

Com a introdução do sistema `Glycan RegEx` é possível escrever padrões de buscas de motifs em um formato mais declarativo e eficaz, com uso de operadores clássicos que permitem negação, repetição e demais operações. Dessa forma, o algoritmo de busca de motifs tem a capacidade de gerar diversos subgrafos intermediários, ao contrário do mecanismo anterior que gera somente um por motif, encontrando a combinação ideal de motifs de forma refinada e mais simples para o programador.

Esta camada adicional que o sistema traz pode gerar um *overhead* em estruturas mais simples do glicano quando comparado com a busca tradicional do `glycowork` por motifs estáticos. Entretanto, quando aplicado em glicanos mais complexos ou maiores, tal

mecanismo se sobressai tanto em eficiência computacional quanto em simplicidade, pois será necessária somente uma expressão para encontrar o motif desejado.

3 Funções chave e eventos de código

Para ilustrar o funcionamento do framework glycowork para expressões regulares de glicanos, exploraremos algumas de suas funções chave e apresentaremos exemplos de código.

3.1 preprocess_patern

A função *preprocess_pattern* é responsável por transformar uma expressão regular de glicano em uma lista de "pedaços"ou componentes. Isso é crucial porque as expressões de glicanos podem conter modificadores, quantificadores e estruturas condicionais que precisam ser interpretadas corretamente antes da correspondência de padrões.

```
def preprocess_pattern(pattern: str) -> List[str]:
    """Transform glyco-regular expression into chunks"""
```

Figura 1: Implementação da função process_pattern

3.1.1 Exemplo de uso

Considere a expressão regular de glicano $ex-HexNAc-([Hex|Fuc])\{1,2\}-HexNAc$. Esta expressão busca um padrão que começa com Hex, seguido por $HexNAc$, depois um ou dois Hex ou Fuc, e termina com $HexNAc$.

```python
import re

def specify_linkages(pattern_component: str):
    if re.search(r"[\d|\?]\(|\d$", pattern_component):
        pattern = re.compile(r"([ab\?])(\d+/\d+|\d|\?)\(\?1-\?\)")
        def replacer(match):
            letter, number = match.groups()
            return f'({letter}1-{number})'
        pattern_component = pattern.sub(replacer, pattern_component)
    return re.sub(r'(5Ac|5Gc|Kdn|Sia)\(a1', r'\1(a2', pattern_component)

# Exemplos de entrada
component1 = "Mana6"
component2 = "Galb3/4"

# Aplicação da função
specified_linkage1 = specify_linkages(component1)
specified_linkage2 = specify_linkages(component2)

print(f"Componente original 1: {component1} -> Ligação especificada:
{specified_linkage1}")
print(f"Componente original 2: {component2} -> Ligação especificada:
{specified_linkage2}")
```

Figura 2: Aplicação da função process_pattern

3.1.2 Comportamento de entrada

Para a entrada $Hex{-}HexNAc{-}([Hex|Fuc])\{1,2\}{-}HexNAc$, a função *preprocess_pattern* irá:

1. Substituir qualquer . por Monosaccharide;
2. Dividir a string em componentes com base em um padrão de expressão regular que identifica modificadores e quantificadores $([Hex|Fuc], 1, 2)$;
3. Remover strings vazias e espaços em branco, além de hífens (-) das extremidades dos componentes.

O resultado esperado será uma lista de strings, onde cada string representa um "pedaço"da expressão regular de glicano, pronto para ser processado por outras funções do framework.

3.2 specify_linkages

A função *specify_linkages* converte notações abreviadas de ligações em sua forma completa. Isso é essencial para padronizar as representações de glicanos e garantir que as operações de grafo possam interpretar corretamente as ligações entre os monossacarídeos.

```
def specify_linkages(pattern_component: str) -> str:
    """Convert expression linkages from shorthand to full notation, such as
Mana6 to Man(a1-6) or Galb3/4 to Gal(b1-3/4)"""
```

Figura 3: Implementação da função specify_linkages

3.2.1 Exemplo de uso

Considere um componente de padrão como $Mana6$ ou $Galb3/4$.

```
import re

def specify_linkages(pattern_component: str):
    if re.search(r"[\d|\?]\(|\d$", pattern_component):
        pattern = re.compile(r"([ab\?])(\d+/\d+|\d|\?)\(\?1-\?\)")
        def replacer(match):
            letter, number = match.groups()
            return f'({letter}1-{number})'
        pattern_component = pattern.sub(replacer, pattern_component)
    return re.sub(r'(5Ac|5Gc|Kdn|Sia)\(a1', r'\1(a2', pattern_component)

# Exemplos de entrada
component1 = "Mana6"
component2 = "Galb3/4"

# Aplicacão da funcão
specified_linkage1 = specify_linkages(component1)
specified_linkage2 = specify_linkages(component2)

print(f"Componente original 1: {component1} -> Ligação especificada:
{specified_linkage1}")
print(f"Componente original 2: {component2} -> Ligação especificada:
{specified_linkage2}")
```

Figura 4: Aplicação da função specify_linkages

3.2.2 Comportamento de entrada

Para $Mana6$, a função identificaria a notação abreviada e a converteria para $Man(a1-6)$. Similarmente, $Galb3/4$ seria convertida para $Gal(b1-3/4)$. Esta padronização é vital para a consistência na representação das ligações glicosídicas.

3.3 get_match

A função get_match é a principal interface para encontrar correspondências de uma expressão regular de glicano em uma estrutura de glicano. Ela utiliza as funções auxiliares como $preprocess_pattern$ e $specify\_linkages$ internamente para realizar a correspondência de padrões baseada em grafos.

```python
def get_match(pattern: Union[str, List[str]],
              glycan: Union[str, nx.DiGraph],
              return_matches: bool = True
             ) -> Union[bool, List[str]]:
    """Find matches for glyco-regular expression in glycan"""
```

Figura 5: Implementação da função get_match

3.3.1 Exemplo de uso

Para um exemplo prático de *get_match*, precisaríamos de um ambiente glycowork completo, incluindo a capacidade de converter strings de glicanos em objetos networkx.DiGraph. No entanto, podemos ilustrar o conceito com um exemplo simplificado de como a função seria invocada e o tipo de resultado esperado.

```python
# Este é um exemplo conceitual e requer o pacote glycowork instalado e configurado
# from glycowork.motif.regex import get_match
# from glycowork.glycan_data.loader import glycan_to_nxGraph

# Exemplo de expressão regular de glicano
pattern_to_find = "Hex-HexNAc"

# Exemplo de estrutura de glicano (representação simplificada para ilustração)
glycan_structure = "GlcNAc(b1-2)Man(a1-3)[GlcNAc(b1-2)Man(a1-6)]Man(b1-4)GlcNAc(b1-4)GlcNAc"

# Converter a string do glicano para um grafo (passo interno do glycowork)
# ggraph = glycan_to_nxGraph(glycan_structure)

# Aplicação conceitual da função
# matches = get_match(pattern_to_find, glycan_structure, return_matches=True)

# print(f"Padrão a ser encontrado: {pattern_to_find}")
# print(f"Estrutura do glicano: {glycan_structure}")
# print(f"Correspondências encontradas: {matches}")
```

Figura 6: Aplicação da função specify_linkages

3.3.2 Comportamento de entrada

Quando *get_match* é chamada com um pattern (expressão regular de glicano) e um glycan (estrutura de glicano), ela realiza os seguintes passos:

1. Pré-processamento do Padrão: Utiliza *preprocess_pattern* para quebrar a expressão regular em componentes;

2. Conversão do Glicano: Se o glicano for fornecido como string, ele é convertido para um objeto *networkx.DiGraph*, que é a representação interna usada para operações de grafo;

3. Correspondência de Padrões Baseada em Grafo: A função então executa um algoritmo de correspondência de subgrafo para encontrar todas as ocorrências do padrão dentro do grafo do glicano;

4. Formatação dos Resultados: Se $return\_matches$ for True, a função retorna uma lista de strings, onde cada string representa uma subestrutura de glicano que corresponde ao padrão. Caso contrário, retorna um booleano indicando se alguma correspondência foi encontrada.

Este processo permite que o framework glycowork identifique padrões complexos em estruturas de glicanos, superando as limitações das expressões regulares textuais tradicionais ao considerar a topologia e as ligações químicas dos glicanos.

4 Considerações finais

As considerações finais formam a parte final (fechamento) do texto, sendo dito de forma resumida (1) o que foi desenvolvido no presente trabalho e quais os resultados do mesmo, (2) o que se pôde concluir após o desenvolvimento bem como as principais contribuições do trabalho, e (3) perspectivas para o desenvolvimento de trabalhos futuros, como listado nos exemplos de seção abaixo. O texto referente às considerações finais do autor deve salientar a extensão e os resultados da contribuição do trabalho e os argumentos utilizados estar baseados em dados comprovados e fundamentados nos resultados e na discussão do texto, contendo deduções lógicas correspondentes aos objetivos do trabalho, propostos inicialmente.

4.1 Principais contribuições

Texto.

4.2 Limitações

Texto.

4.3 Trabalhos futuros

Texto.