UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO (UFERSA)
CENTRO DE CIÊNCIAS EXATAS E NATURAIS (CCEN)
DEPARTAMENTO DE COMPUTAÇÃO
DISCIPLINA: COMPILADORES
PROFESSOR: PATRÍCIO DE ALENCAR SILVA
PROJETO: ANALISADOR LÉXICO PARA TEXTUAL ONTOLOGY LANGUAGE

OBJETIVO: Projetar um analisador léxico para dar suporte à análise de corretude de uma ontologia especificada textualmente com a linguagem TONTO (*Textual Ontology Language*).

CONTEXTUALIZAÇÃO: TONTO é o acrônimo para *Textual Ontology Language* (Linguagem de Ontologia Textual), sendo uma forma de especificar textualmente uma ontologia como artefato computacional. Ontologias computacionais são grafos de conhecimento que conectam conceitos que detêm alguma relação de sentido, sendo normalmente especificadas em linguagem lógica interpretável por máquina, tais como *Resource Description Framework* (RDF) [1] ou *Web Ontology Language* (OWL) [2]. Ontologias computacionais constituem blocos de construção da chamada Web 3.0 (ou Web Semântica), onde cada nó do grafo da ontologia corresponde a algum recurso na Web identificável com um Identificador Universal de Recursos (Uniform Resource Identifier – URI). Quando recursos da Web tais como vídeos, áudio, texto, imagens e código são identificados por URIs, e quando tais recursos são instanciados em uma ontologia, cria-se uma base de conhecimento altamente dinâmica e extensível que pode ser interpretada por máquina para inferência de novas relações. A Web 3.0 é a base das redes sociais e das plataformas multimídia e de comércio eletrônico.

Entretanto, novas linguagens têm sido propostas com o objetivo de facilitar o entendimento do que é uma ontologia computacional e permitir que especialistas de várias áreas do conhecimento possam se engajar na tarefa de construir esse tipo de modelo. Por exemplo, a linguagem OntoUML, proposta por Guizzardi et al. (2018) [3] é baseada na semântica dos diagramas de classe UML, mas acrescenta a esses modelos um conjunto de estereótipos de ordem lógica e filosófica de forma dar suporte à construção de modelos com significado mais profundo. Mais recentemente, o trabalho de Coutinho et al. (2024) [4] propôs um novo jeito de modelar ontologias: a Textual Ontology Language (TONTO), passível de análise formal e de geração de vários outros tipos de modelos de saída, incluindo a própria OntoUML, o formato JSON e o meta-modelo gUFO (serialização OWL de uma ontologia especificada em OntoUML). A linguagem TONTO dispõe de uma extensão para o ambiente VSCode. Com essa extensão, um ontologista poderá criar uma ontologia em módulos (que serão salvos com a extensão **.tonto**), realizar a junção desses módulos com o Tonto Package Manager (que usa um código de orquestração que especifica as dependências entre os módulos – o arquivo tonto.json) e gerar vários formatos da ontologia completa, p.ex.: um modelo completo em JSON, ou um modelo em gUFO (que pode ser visualizado e continuado em um ambiente de edição de ontologias como o Protégé). É possível ainda importar arquivos JSON externos e transformá-los em TONTO, para realizar a reengenharia do modelo. Todos os detalhes de uso da ferramenta podem ser encontrados na monografia original de Matheus Lenke (ex-aluno da UFES), autor da ferramenta [5].

DESCRIÇÃO DO PROBLEMA: Projetar um analisador léxico para a linguagem TONTO para reconhecer os elementos da linguagem. Considerando que o autor da ferramenta já disponibiliza um analisador para tal, alguns requisitos mais específicos serão dados neste trabalho de implementação e pesquisa, de forma a permitir que um ontologista crie um documento TONTO com um formato bem definido para cada um dos elementos da linguagem. O analisador deve reconhecer os seguintes casos:

- **Estereótipos de classe**: `event, situation, process, category, mixin, phaseMixin, roleMixin, historicalRoleMixin, kind, collective, quantity, quality, mode, intrisicMode, extrinsicMode, subkind, phase, role, historicalRole.`
- **Estereótipos de relações**: `material, derivation, comparative, mediation, characterization, externalDependence, componentOf, memberOf, subCollectionOf, subQualityOf, instantiation, termination, participational, participation, historicalDependence, creation, manifestation, bringsAbout, triggers, composition, aggregation, inherence, value, formal, constitution.`
- **Palavras reservadas**: `genset, disjoint, complete, general, specifics, where, package, import, functional-complexes.`
- **Símbolos especiais**: “{“, “}”, “(“, “)”, “[“, “]”, “..”, “<>--” , “--<>”, “*”, “@”, “:”.
- **Convenção para nomes de classes**: iniciando com letra maiúscula, seguida por qualquer combinação de letras, ou tendo sublinhado como subcadeia própria, sem números. Exemplos: *Person, Child, Church, University*, *Second_Baptist_Church*.
- **Convenção para nomes de relações**: começando com letra minúscula, seguida por qualquer combinação de letras, ou tendo sublinhado como subcadeia própria, sem números. Exemplos: *has*, *hasParent, has_parent, isPartOf, is_part_of*.
- **Convenção para nomes de instâncias**: iniciando com qualquer letra, podendo ter o sublinhado como subcadeia própria e terminando com algum número inteiro. Exemplos: *Planeta1, Planeta2, pizza03, pizza123*.
- **Tipos de dados nativos**: `number, string, boolean, date, time, datetime.`
- **Novos tipos**: iniciando com letra, sem números, sem sublinhado e terminando com a subcadeia “DataType”. Exemplo: *CPFDataType, PhoneNumberDataType*.
- **Meta-atributos**: `ordered, const, derived, subsets, redefines.`

REQUISITOS: O trabalho deverá ser elaborado e avaliado de acordo com os seguintes critérios:

- O projeto será parcialmente automatizado com o uso da ferramenta LEX e suas variações: FLEX e PLY. A implementação deverá ser feita em C++ ou Python.
- Exemplos de teste disponíveis em: https://github.com/patricioalencar/Compiladores_UFERSA
- Os trabalhos deverão estar no GitHub, com documentação apropriada.
- A saída da análise deverá compreender duas visualizações: (1) visão analítica de todos os tokens, com indexação de linha e coluna de localização no código-fonte; e (2) tabela de síntese, contendo quantidades de classes, relações, palavras-chave, indivíduos (instâncias, se houver), palavras reservadas e meta-atributos.
- Erros deverão ser tratados pela linha onde foram encontrados, com sugestões de tratamento compatíveis com a especificação da linguagem. Não deixar estourar erros sem explicação ou tratamento.
- A atividade de cada membro do grupo será verificada no GitHub.
- O vídeo explicativo de 05 (cinco) minutos é obrigatório.

DIREÇÕES DE PESQUISA

- Estudar sobre a ferramenta LEX e suas variantes FLEX ou PLY.

- Elaborar gramática da linguagem no LEX como fachada de teste de corretude da linguagem, codificar programa principal para leitura dos dados de entrada e usar os códigos disponibilizados para teste.
- Uma forma de realmente aprender sobre a linguagem é tentando fazer seus próprios exemplos. A extensão TONTO para VSCode permite criar o código da ontologia, validá-lo quanto à sua corretude e gerar código completo do modelo, tal como o arquivo JSON da ontologia completa. Uma outra possibilidade é usar a ferramenta Visual Paradigm, com o plugin OntoUML. A partir de um modelo gráfico gerado nessa ferramenta, é possível exportá-lo para JSON. Esse formato serve de entrada para a extensão TONTO do VSCode, o qual o transforma em uma versão TONTO da ontologia. Ou seja, as ferramentas VSCode e Visual Paradigm são interoperáveis, mas há alguns caminhos unidirecionais, pois nem todas as transformações são feitas de uma ferramenta para outra. Tanto o VSCode quanto o Visual Paradigm exportam um modelo ontológico para gUFO, que é uma serialização em OWL da ontologia, a qual pode ser trabalhada e estendida no ambiente Protégé, de edição de ontologias.

REFERÊNCIAS

1. W3C. (2025). Resource Description Framework – Concepts and Abstract Data Model. Disponível online em: https://www.w3.org/TR/rdf12-concepts/
2. W3C. (2012). Web Ontology Language Conformance (Second Edition). Disponível online em: https://www.w3.org/TR/owl2-conformance/
3. Guizzardi, G., Fonseca, C. M., Benevides, A. B., Almeida, J. P. A., Porello, D., & Sales, T. P. (2018, September). Endurant types in ontology-driven conceptual modeling: Towards OntoUML 2.0. In *International Conference on Conceptual Modeling* (pp. 136-150). Cham: Springer International Publishing.
4. Coutinho, M. L., Almeida, J. P. A., Sales, T. P., & Guizzardi, G. (2024). A Textual Syntax and Toolset for Well-Founded Ontologies. In *14th International Conference on Formal Ontology in Information Systems, FOIS 2024* (pp. 208-222). IOS.
5. Lenke, M., Tonto: A Textual Syntax for OntoUML – A textual way for conceptual modeling. Available online at: https://matheuslenke.github.io/tonto-docs/