

**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 SEMÂNTICO PARA A TEXTUAL ONTOLOGY LANGUAGE

**OBJETIVO:** Projetar um analisador semântico para validação de padrões de projeto de ontologias (*ontology design patterns* – ODPs) especificados na Textual Ontology Language (Tonto).

**CONTEXTUALIZAÇÃO:** Neste semestre, estudamos como aplicar técnicas do processo de compilação para análise de construtos da linguagem Tonto (Textual Ontology Language) [1]. A análise semântica envolve a validação de estruturas de código em função de seu contexto. Há vários escopos que definem o contexto de um operador de uma linguagem. Esse contexto pode ser imediato (p.ex.: parâmetros próximos, nomes de métodos ou funções, argumentos e outros “complementos” do operador) ou pode ser de ordem maior (p. ex.: importação de conceitos ou classes de múltiplos pacotes ou relações lógicas entre classes). A linguagem Tonto é “declarativa”, ou seja, é usada para representar conhecimento na forma de proposições lógicas a serem analisadas, interpretadas ou como base de inferência de agentes de software inteligentes [2]. Na fase de análise sintática, foram verificadas as formas corretas de se declarar classes, relações, cardinalidades e outros construtos da linguagem Tonto. Na fase de análise semântica, o objetivo será validar padrões de projeto de ontologias (*ontology design patterns* - ODPs) que possam estar presentes no código. Os ODPs conferem uma estrutura lógica mais formal à especificação da ontologia, pois definem regras de combinação dos conceitos e relações [3].

**DESCRIÇÃO DO PROBLEMA:** Projetar um **analisador semântico** para a linguagem TONTO para validação dos seguintes padrões de projeto de ontologias:

1. **Subkind pattern:**

```
Subkind_Pattern.tonto > ...
1  package Subkind_Pattern
2
3  kind ClassName
4  subkind SubclassName1 specializes ClassName
5  subkind SubclassName2 specializes ClassName
6
7
8  disjoint complete genset Kind_Subkind_Genset_Name {
9      general ClassName
10     specifics SubclassName1, SubclassName2
11 }
12 // "complete" is optional, but "disjoint" applies to subkinds
```

## 2. Role pattern:

```
Role_Pattern.tonto > ...
1 package Role_Pattern
2
3 kind ClassName
4 role Role_Name1 specializes ClassName
5 role Role_Name2 specializes ClassName
6
7 complete genset Class_Role_Genset_Name {
8     general ClassName
9     specifics Role_Name1, Role_Name2
10 }
11 /"complete" is optional, but "disjoint" doesn't apply to roles
```

## 3. Phase pattern:

```
Phase_Pattern.tonto > ...
1 package Phase_Pattern
2
3 kind ClassName
4 phase Phase_Name1 specializes ClassName
5 phase Phase_Name2 specializes ClassName
6 phase Phase_NameN specializes ClassName
7
8 disjoint complete genset Class_Phase_Genset_Name {
9     general ClassName
10    specifics Phase_Name1, Phase_Name2, Phase_NameN
11 }
12 /"disjoint" is mandatory for phases, but "complete" is optional
```

## 4. Relator pattern:

```
Relator_Pattern.tonto > ...
1 package Relator_Pattern
2
3 kind ClassName1
4 kind ClassName2
5
6 role Role_Name1 specializes ClassName1
7 role Role_Name2 specializes ClassName2
8
9 relator Relator_Name {
10     @mediation [1..*] -- [1..*] Role_Name1
11     @mediation [1..*] -- [1..*] Role_Name2
12 }
13
14 @material relation Role_Name1 [1..*] -- relationName -- [1..*] Role_Name2
15 /"relationName" can be replaced by a specific name for the relation
```

## 5. Mode pattern

```
Mode_Pattern.tonto > Mode_Pattern > Mode_Name1
1 package Mode_Pattern
2
3 kind ClassName1
4 kind ClassName2
5
6 mode Mode_Name1 {
7     @characterization [1..*] -- [1] ClassName1
8     @externalDependence [1..*] -- [1] ClassName2
9 }
```

## 6. RoleMixin pattern:

```
RoleMixin_Pattern.tonto > RoleMixin_Pattern
1 package RoleMixin_Pattern
2
3 kind ClassName1
4 kind ClassName2
5
6 roleMixin RoleMixin_Name
7
8 role Role_Name1 specializes ClassName1, RoleMixin_Name
9 role Role_Name2 specializes ClassName2, RoleMixin_Name
10
11 disjoint complete genset RoleMixin_Genset_Name {
12     general RoleMixin_Name
13     specifics Role_Name1, Role_Name2
14 }
```

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

- **Testes** continuam disponíveis em: [https://github.com/patricioalencar/Compiladores\\_UFERSA](https://github.com/patricioalencar/Compiladores_UFERSA);
- Os trabalhos deverão estar no GitHub, com **documentação** apropriada (Valor: 2 pontos);
- A saída da análise deverá conter: (1) **padrões completos identificados no código**; (2) **padrões incompletos identificados por sobrecarregamento** (Valor: 5 pontos);
- O **vídeo** explicativo de 05 (cinco) minutos continua **obrigatório** (Valor: 1 ponto);
- **Tratamento de erros:** uso de técnica de coerção para corrigir erros (Valor: 2 pontos).

## REFERÊNCIAS

1. 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.
2. Lenke, M., Tonto: A Textual Syntax for OntoUML – A textual way for conceptual modeling. Disponível online em: <https://matheuslenke.github.io/tonto-docs/>
3. Ruy, F. B., Guizzardi, G., Falbo, R. A., Reginato, C. C., & Santos, V. A. (2017). From reference ontologies to ontology patterns and back. *Data & Knowledge Engineering*, 109, 41-69.