Tabelas de decisão Pré-processador em Python

Proposta de trabalho

Este trabalho de conclusão de curso propõe o desenvolviemento de um pré-processador para tabelas de decisão (TDs) em Python, em que a abordagem e método de tradução de TDs se baseia na dissertação de Satoshi Nagayama [Nagayama, 1990]. A dissertação apresenta diversos modelos de tradução de TDs, como o switch method, que foi utilizado no pré-processador deste trabalho. Além disso, ressalta-se a relevância da documentação, utilizando uma técnica de auto-documentação do Prof. Dr. Valdemar W. Setzer [Setzer, 1988].

Introdução

A programação, em sua essência, envolve a criação de algoritmos que fazem escolhas com base em condições lógicas. Tradicionalmente, essas escolhas lógicas são implementadas por meio de árvores de decisão com aninhamentos de 'if...then...else', que podem se tornar complexas e difíceis de manter. Nesse contexto, as TDs emergem como uma alternativa poderosa, oferecendo uma forma compacta, gráfica e organizada de especificar escolhas lógicas [Setzer, 2024].

Exemplo de TD

O exemplo abaixo tem 3 condições com 8 colunas, que corresponderiam às folhas de uma árvore de decisão, que teria 15 nós. Além da simplificação da aparência visual, note-se a separação das condições das ações.

Hierarquia	Gestor	Gestor	Gestor	Gestor	Analista	Analista	Analista	Analista
Setor	Comercial	Comercial	Administrativo	Administrativo	Comercial	Comercial	Administrativo	Administrativo
Metas atingidas	Y	N	Y	N	Y	N	Y	N
Inscrição em cursos de desenvolvimento	1	1	1	1	1	1	1	1
Bonificação de 5% do salário	-	-	-	-	2	2	-	-
Bonificação de 10% do salário	2	2	-	-	-	-	2	2
Bonificação de 15% do salário	-	-	2	2	-	-	-	-
3 dias de folga	3	-	3	_	3	_	3	-

TD no código-fonte

```
exemplos > tds_traduzidas > 🏚 poster_TD.py > ...
      hierarquia = ''
      setor =
      meta atingida = ''
      def inscreve em cursos():
          print(f'Funcionário inscrito em cursos de desenvolvimento')
      def define bonificacao(valor:int):
          print(f'Funcionário recebe bonificacção de {valor}% do salário')
      def atribui 3dias de folga():
          print(f'Funcionário recebe 3 dias de folga')
          name == ' main ':
          hierarquia = input('').upper()
          setor = input('').upper()
          meta atingida = (input('').upper() == 'Y')
          #TD decision table td estrutura de incentivos
 19
 20
                                                             {"GESTOR"}
                     hierarquia gestor
 21
          #TD
                                                             "ANALISTA"}
                     hierarquia analista
 22
          #TD
                                                             {"COMERCIAL"}
                     setor comercial
 23
                     setor administrativo
                                                            {"ADMINISTRATIVO"
          #TD conditions
 24
 25
          #TD
                     hierarquia
                                               hierarquia gestor hierarquia gestor
                                                                                      hierarquia gestor
                                                                                                                hierarquia qe
 26
          #TD
                                               setor comercial
                                                                   setor comercial
                                                                                       setor administrativo
                                                                                                               setor adminis
                     setor
 27
          #TD
                     meta atingida
          #TD actions
 29
                     inscreve em cursos()
 30
                     define bonificacao(5)
 31
          #TD
                     define bonificacao(10)
 32
          #TD
                     define bonificacao(15) θ
 33
                     atribui 3dias de folga() 3
          #TD end table
 34
```

Código gerado

```
exemplos > tds_traduzidas > 🏺 poster_TD_processado.py > 📵 hierarquia
          #2 #TD end table
36
37
          def decision table td estrutura de incentivos() ->None:
              I θ = θ #Inicialização do auxiliar da condição hierarquia
              I_1 = 0 #Inicialização do auxiliar da condição setor
              I_2 = θ #Inicialização do auxiliar da condição meta_atingida
              I = θ #Inicialização do número da regra
              if hierarquia in set(["GESTOR"]):
                I \theta = \theta
              elif hierarquia in set(["ANALISTA"]):
                 I \theta = 1
              if setor in set(["COMERCIAL"]):
                 I 1 = 0
              elif setor in set(["ADMINISTRATIVO"]):
                 I 1 = 1
              if meta atingida == True:
                 I 2 = 0
              elif meta_atingida == False:
54
              I = (2*2*1)*I_0 + (2*1)*I_1 + (1)*I_2
55
              match I:
                      inscreve em cursos(
58
                      define bonificacao(10)
59
                      atribui 3dias de folga(
60
                      inscreve em cursos(
62
                      define bonificacao(10)
63
                  case 2:
                      inscreve em cursos(
                      define bonificacao(15)
                      atribui 3dias de folga(
67
                      inscreve em cursos(
69
                      define bonificacao(15)
                      inscreve em cursos(
 72
                      define bonificacao(5)
73
                      atribui_3dias_de_folga(
                      inscreve em cursos()
                      define bonificacao(5)
 77
                      inscreve em cursos()
                      define bonificacao(10)
                      atribui 3dias de folga()
 81
                      inscreve em cursos()
83
84
85
          decision_table_td_estrutura_de_incentivos()
```

Resultados

O pré-processador desenvolvido foi capaz de traduzir TDs em código Python executável, utilizando o método de tradução switch method, foi projetado usando o Strategy Pattern[Freeman et al., 2004], possibilitando a fácil extensão de novos métodos de tradução; A modularidade do sistema permitiu uma fácil manutenção e evolução do projeto; E a integração de práticas de auto-documentação foi outro marco importante do trabalho por acarretar em uma documentação clara, atualizada e concisa.

Bibliografia

- ► Freeman, Eric et al. (Nov. 2004). *Head First Design Patterns: A Brain-Friendly Guide*. Sebastopol, CA, USA: O'Reilly Media.
- ▶ Nagayama, Satoshi (1990). "Implementação de Gerador I-M-E com Tabelas de Decisão". PhD thesis. São Carlos, SP, Brasil: Instituto de Ciências Matemáticas e de Computação, USP.
- ▶ Setzer, Valdemar W. (setembro 1988). *Um Sistema Simples para Documentação Semi-Automática de Programas*. Tech. rep. RT-MAC-8808. Relatório Técnico. São Carlos, SP, Brasil: Instituto de Matemática e Computação, Universidade de São Paulo (USP).
- ▶ (2024). *Comunicação pessoal*. Orientador do Trabalho de Conclusão de Curso.

Aluno: Eduardo Ribeiro Silva de Oliveira Orientador: Prof Dr Valdemar W. Setzer

eduardo.rso784@usp.br
setzerv@gmail.com

