Universidade de Pernambuco Escola Politécnica de Pernambuco

Engenharia da Computação

Relatório Compiladores

Professor: Luiz Menezes

André Filipe Menezes

Elias Amaro da Silva Junior

Guilherme Novaes

João Paulo

Sumário

1 Motivação

Para simplificar a construção de componentes html, criaremos uma linguagem que facilitará a construção destes elementos e um compilador capaz de converter o código gerado a partir desta linguagem em um código HTML com os componentes prontos. Com isso, acreditamos que seja feita de forma mais intuitiva e veloz a construção de tags HTML.

2 Exemplos da Linguagem e do Resultado Esperado

A linguagem foi definida através da estrutura abaixo, onde:

- Atributos: nomeAtributo = valor
- Filhos: #nome(propriedades)

Construção de uma entrada(É possível utilizar quebras de linha)

```
tag(
  at1=a1,
  at2 = a2,
  ...,
  #(tag(...)),
  #(tag(...))
)
```

Exemplo :

- Entrada

```
<!-- Input: -->
table(
    id=table1,
    class=tableOne,
        #tr(
            id=trTeste,
                #td(
                    id = teste1
                ),
                #td(
                    id = teste2,
                )
        )
)
```

- Saída

```
<!-- Output - HTML: -->
<table id="table1" class="tableOne">
    <tr id="trTeste">
        <td id="tdTeste"></td>
        <td id="tdTeste2"></td>
    </tr>
</table>
```

3 Gramática da Linguagem

```
Exp = Tag ("\\n"? Tag)*
Tag = ("(" text ")")                          --text
    | (name "(" Prop ("," Prop)* ")")         --tag

Prop = PropTag                                --tag
    |  Children                               --children
    |  ""                                     --empty
Children = "#" Tag
PropTag = propName "=" propValue
propName = letter+
propValue = alnum+
name = letter+
text = (alnum | space)+
```

3 Descrição do Processo

Nós decidimos gerar o código a partir de um objeto gerado através de uma árvore semântica. De início, nós criamos a semântica e adicionamos uma operação na mesma para criar um objeto em um formato adequado através da árvore. Com o objeto criado, nós extraímos suas propriedades e construímos as tags. Portanto, a saída do nosso programa foi um objeto gerando as tags html.

- Passo a passo:

1. Criação da gramática;
2. Criação da semântica;
3. Chamada da função de validação e geração de código a partir das entradas;
4. Se a entrada for inválida, a execução é interrompida e uma mensagem de erro é mostrada no terminal. Se a entrada for válida, a execução continua;
5. Adiciona a operação de criação da árvore semântica na semântica. A árvore será gerada e convertida em um objeto.
6. Chamada da função que converte o objeto gerado através da árvore no resultado esperado.
7. Ao entrar na chamada do método de conversão, será extraído do objetos o array de expressões que armazena todas as tags.
8. Após isso são extraídas, de cada tag, suas propriedades e filhos.
9. Sabendo as informações da tag, ela será criada.
10. Todos os dados extraídos serão formatados e convertidos em string.
11. Todos as strings geradas no passo anterior serão adicionadas na string result que armazenará as tags prontas.
12. Antes de fechar a tag, será avaliado a forma que deverá ser fechada e se ela tem filhos.
13. Os filhos serão adicionados dentro da tag atual de forma recursiva.
14. Será retornado um objeto com o html.

15. Por último, os dados serão salvos nos arquivos que estão na pasta generated.

4 Manual de uso do software

- Abrir a aba do Terminal pelo VSCode;
- Rodar o **npm install** para instalar as dependências;
- Digitar **npm run start**;
- Ao concluir a execução, a mensagem abaixo será exibida: ~/ProjetoCompiladoresComOhm$ Arquivo html criado com sucesso! Diretório: "src/generated/html.html"
- Todo o código gerado estará na pasta generated. No arquivo html.html estará todo o HTML.