


<div><div>Universidade Federal do Ceará – Campus de Russas</div></div>			AV1	AV2	PF
		Trabalho		X	
		Prova			
Curso: C.C. e E. S.	Disciplina: Linguagens Formais e Autômatos	Data Sigaa Submissão: 21/08/2024 Data Resultados: 20/08 até 27/08/2024			
Professor: Cenez Araújo de Rezende					
Alunos(as):					

1 Instruções preliminares:

1. A atividade deve ser realizada em equipe de até 4 **participantes**;
2. Forma de Entrega: Projeto Aplicado de uma gramática formal, com a realização de testes em ferramentas apropriadas, indicadas neste documento. Para isso, testar e discutir as dificuldades e facilidades encontradas no contexto das gramáticas e autômatos com pilha;
3. A submissão dos resultados ocorre **até a data do Sigaa**;
4. **A discussão** dos resultados ocorre em sala de aula a partir **da data de submissão no Sigaa**, via breve seminário objetivo. Agendar participação no link [2] nas referências deste documento.

2 Contexto do Trabalho Aplicado (TA)

Problema: A disciplina de Linguagens Formais e Autômatos busca permitir que os estudantes (e pesquisadores) conheçam as técnicas e estudos utilizados nas definições e reconhecimento de linguagens, sejam elas de programação de computadores ou não. Porém, na prática, e em tecnologia, a tendência é linguagem de programação, já que está diretamente ligada à computação. Um grande nome responsável pelas ideias inseridas na disciplina é o linguista *Noam Chomsky*. Ele não é uma figura direta da Ciência Computacional, mas uniu elementos da matemática à sua área de atuação, que é linguística. Computação é toda e qualquer forma de fazer cálculo (ou *computar*). Estabelecer regras e técnicas para construção de linguagens, visando computação, possui grande relevância para as tecnologias contemporâneas. Isso fez surgir vários artefatos, que ajudam no entendimento especialmente de gramáticas e autômatos com pilha (*PDA* ou *Pushdown Automata*). Podemos citar, por exemplo, as ferramentas JFlap[3] e BNFPlayground[4], bem como poderíamos em uma rápida pesquisa perceber várias iniciativas de ferramentas com vocação para o ensino e aprendizagem. Neste trabalho, busca-se oportunizar a discussão e o entendimento da parte gramatical/autômatos com pilha através das ferramentas e códigos que disponibilizamos nos anexos. Para isso, observe nossa atividade na próxima seção.

3. Atividades

3.1 [70%] Estudo e relatório individual para subir no Sigaa: Ver as gramáticas do Anexo I e Inserir no software *Jflap*. Note as questões seguintes para discussão:

- Inserir várias entradas de testes nas gramáticas;
- Gerar o autômato com pilha, e inserir seus testes feitos nas gramáticas;
- Discuta sua experiência e as diferenças que encontrou nessas gramáticas. Qual achou melhor e como chegou nesta conclusão?

3.2 [20%] Grupo - Trabalhando nos Projetos Anexos[6,7,8] deste Trabalho:

Nos arquivos anexos deste trabalho há codificações de um PDA em Java (*online[5]*, bem como compactado para a plataforma eclipse da Oracle). Parcialmente, temos também um projeto de PDA em Python[7] e um completo AFD em Python[8], sendo este uma arquitetura base, semelhante à do PDA Java citado. Ou seja, com ajustes pode ser transformado também em um PDA Python. O PDA pode receber tanto gramáticas como os tradicionais símbolos já estudados nos autômatos. Notar nos projetos anexos:

- Há gramáticas e problemas de exemplo, dentro da classe “Test”:
 - Vejam os exemplos (na classe Test) para testar também outras gramáticas e problemas, inserindo-as na classe Test a partir do exemplar disponível Test. Sugerem-se novos exemplos, tais como gramáticas para operações matemáticas, ou até elementos de linguagens de programação, como um *if*. Traga seus testes para compartilhar em nossa aula. Lembrem-se que as variáveis das gramáticas possuem lado direito, que são empilhados inversamente. EX: $S \rightarrow 0S1$, sai S e fica 1S0 na pilha. Além disso, no “loop”, todo símbolo terminal é lido e desempilhado;
- Tente criar uma gramática para simular um simples **enquanto** (while). Veja o exemplo seguinte, teste-o no *Jflap* para sua gramática. Depois insira a gramática no projeto PDA (Por exemplo no PDA Java) com a seguinte entrada:
 - `w = eqt(a){eqt(a){}}`
 - `w = eqt(a){eqt(a){}}eqt(a){}`
 - onde: “eqt” é uma abreviação de enquanto; e “a” simula uma variável “a”.

3.3 [10%] Trabalhando no Projeto PDA: Tente adaptar uma pilha ao código *Python* ou *Java*: dentro da classe PDA.py ou PDAImpl.java. Essa adaptação ocorre no método “run”, que foi deixado intencionalmente vazio (*TODO*). Feita sua adaptação e inserção de sua pilha, basta trocar o construtor. Veja exemplo para o projeto Java (disponível na classe Test):

IPDA pda = new PDA(q0); //troca para:

IPDA pda = new PDAImpl(q0); // após corrigir o método “run” dessa classe PDAImpl.

- Teste seu PDAImpl adaptado.

3.4 Apresentação em sala: trazer o que o grupo conseguiu executar no projeto e no *Jflap*, bem como o que não conseguiu, para tentarmos solucionar quando possível ou para que possamos informar o motivo.

3.5 Documento de entrega do grupo no Sigaa (Anexar ao documento Individual 3.1):

Descrever em breve relatório a experiência observada na execução do trabalho, pontuando as questões seguintes:

- Quais itens de atividades foram importantes em termos teóricos e práticos?
- Das atividades que conseguiu concluir, classificaria como fácil, mediano ou difícil?

- Das que não conseguiu concluir, quais foram as dificuldades encontradas?
- Utilizou IA para gerar alguma codificação? Caso sim, necessitou de ajustes no código gerado, como foi a experiência?

4. Referências

- [1] Vídeo instrução [aqui](https://youtu.be/2A-SkRM1S00): <https://youtu.be/2A-SkRM1S00>
- [2] Agendamento [aqui](https://docs.google.com/spreadsheets/d/1sQYeNMu6FKp-5mGFAIQIv6g6b6IELrCS5tdScUOHdEg/edit?usp=sharing): <https://docs.google.com/spreadsheets/d/1sQYeNMu6FKp-5mGFAIQIv6g6b6IELrCS5tdScUOHdEg/edit?usp=sharing>
- [3] Jflap: <https://www.jflap.org/>
- [4] Playground: <https://bnfplayground.pauliankline.com/>

Codigos anexos neste trabalho:

- [5] PDA online em Java: <https://replit.com/@CenezAraujo/PDA>
- [6] **Codigos/UFC-PDA-Java/**
- [7] **Codigos/pda_python/**
- [8] **Codigos/afd-python/**

Anexo I

Gramática 1:

S	→	λ
S	→	0 S 1

Gramática 2:

S	→	E \$
E	→	T
E	→	T + E
F	→	(E)
F	→	a
T	→	F
T	→	F * T

Gramática 3:

S	→	E
A	→	λ
A	→	+ T A
B	→	λ
B	→	* F B
E	→	T A
F	→	(E)
F	→	a
T	→	F B

