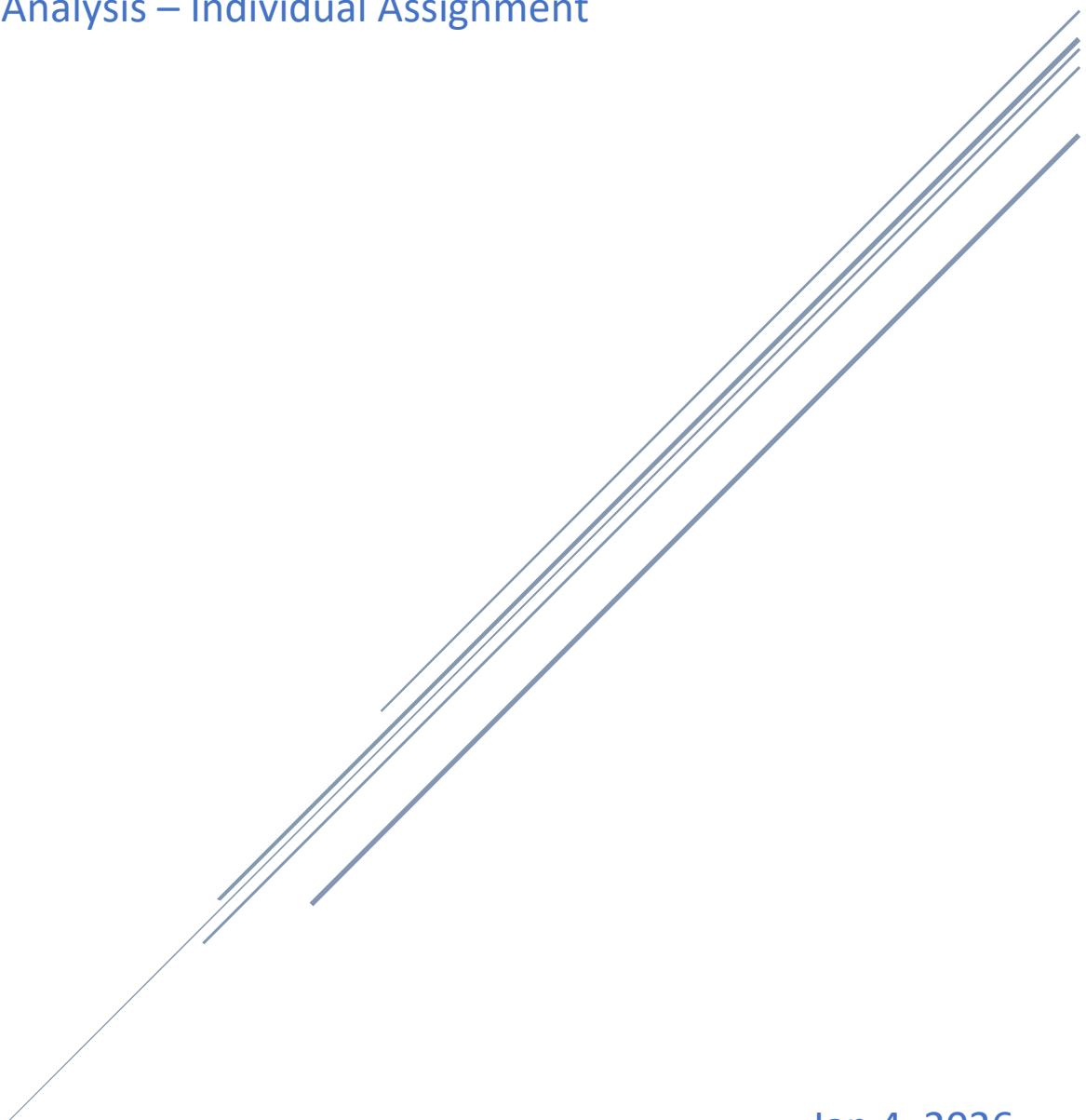




Bahir Dar University
Institute of Technology /BiT

Principles of Compiler Design

Syntax Analysis – Individual Assignment



Jan 4, 2026
Yonatan Ayisheshim [BDU1508377]

Table of Contents

Introduction.....	2
1. Problem Solving.....	2
1.1 Grammar Analysis and Parse Trees	2
1.2 Parse Trees	2
1.3 Observation.....	3
Conclusion	4
References.....	4

Introduction

Syntax analysis is a critical phase of compiler design that verifies whether a sequence of tokens generated by the lexical analyzer conforms to the grammatical structure of the programming language. This assignment focuses on understanding syntactic structures through grammar-based problem solving.

1. Problem Solving

1.1 Grammar Analysis and Parse Trees

Given Grammar: $S \rightarrow aS \mid bS \mid \epsilon$

Where:

- Terminals: {a, b}
- Non-terminal: S
- Start symbol: S
- ϵ denotes the empty string

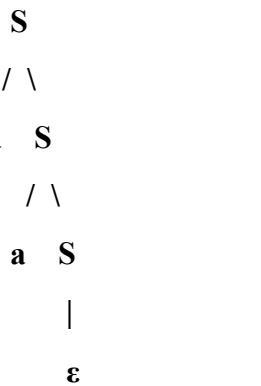
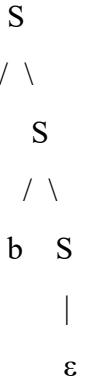
Derivations:

1. **String "aa":** $S \rightarrow aS \rightarrow aaS \rightarrow aa\epsilon = aa$
2. **String "ab":** $S \rightarrow aS \rightarrow abS \rightarrow ab\epsilon = ab$
3. **String "ba":** $S \rightarrow bS \rightarrow baS \rightarrow ba\epsilon = ba$
4. **String "bb":** $S \rightarrow bS \rightarrow bbS \rightarrow bb\epsilon = bb$

The list of strings of length 2 is: aa, ab, ba, bb.

1.2 Parse Trees

Below are the textual representations of the parse trees for each generated string.

A. Parse Tree for "aa"  <pre> graph TD S1[S] --- a1[a] S1 --- a2[a] a1 --- a3[a] a2 --- a4[a] a3 --- e[ε] </pre>	B. Parse Tree for "ab"  <pre> graph TD S2[S] --- a5[a] S2 --- b5[b] a5 --- b6[b] a5 --- e2[ε] </pre>
C. Parse Tree for "ba"  <pre> graph TD S3[S] --- b1[b] S3 --- a1[a] b1 --- a2[a] a1 --- e3[ε] </pre>	D. Parse Tree for "bb"  <pre> graph TD S4[S] --- b4[b] S4 --- b5[b] b4 --- b6[b] b5 --- e4[ε] </pre>

1.3 Observation

The grammar is **right-recursive** and generates **all possible strings** of **a** and **b**, including the empty string. Each character corresponds to a recursive expansion of **s** until termination via ϵ .

Conclusion

This assignment demonstrated grammar-driven string generation and parse tree construction.

Understanding this concept is essential for mastering compiler front-end design and lays the foundation for advanced topics such as semantic analysis and code generation.

References

- Aho, A. V., Lam, M. S., Sethi, R., & Ullman, J. D. *Compilers: Principles, Techniques, and Tools* (2nd ed.)
- Compiler Design Lecture Notes