Campus Ciudad de México

E2 Parallel syntax highlighter

Arturo Utrilla Hernández **A01174331**

Daniel Esteban Hernández García **A01652966**

Rodrigo Martínez Vallejo **A00573055**

Ximena Silva Bárcena **A01785518**

Prof. Sergio Ruiz Loza

Implementation of Computational Methods

Grupo **641**

Tuesday, June 10th, 2025

| Brief report |

The lexical analyzer was implemented in two versions: sequential and parallel processing using Racket's futures mechanism. Performance measurements were conducted on five C++ source files of varying lengths (10, 20, 30, 50, and 100 lines). The following execution times were recorded:

Sequential run:

- **Measure 1**: 7 ms
- **Measure 2**: 8 ms
- **Measure 3**: 3 ms
- **Measure 4**: 3 ms

Average: ***5.25 ms***

Parallel run:

- **Measure 1**: 3 ms
- **Measure 2**: 3 ms
- **Measure 3**: 3 ms
- **Measure 4**: 8 ms

Average: 4.25 ms

Speedup is equal to: ***1.24*** (sequential time / parallel time)

The parallel implementation demonstrates consistent performance improvement across different input sizes, with speedup remaining stable around 1.24x. While the improvement is moderate, it becomes more significant as the number of files increases, validating the effectiveness of the parallel approach for batch processing scenarios.

The lexer uses a recursive algorithm to analyze each token or character. Since it processes one unit per call, the complexity per file is O(L), where L is the number of lines or tokens. Given N files, each processed independently, the total complexity is:
O(N × L)

To validate this, tests were run with more files:

- **10 files sequential**:
  - Measures: 7 ms, 10 ms, 10 ms, 11 ms
  - Average: ***9.5 ms***
- **15 files sequential**:
  - Measures: 15 ms, 16 ms, 17 ms, 15 ms
  - Average: ***15.75 ms***
- **10 files parallel**:
  - Measures: 9 ms, 7 ms, 7 ms, 8 ms
  - Average: ***7.75 ms***
- **15 files parallel**:
  - Measures: 14 ms, 12 ms, 11 ms, 14 ms
  - Average: ***12.75 ms***

These results support the theoretical complexity, showing linear growth as the input size increases.

The algorithm performs efficiently and scales as expected. Parallel execution provides a measurable speedup. The solution benefits from Racket's parallel constructs while keeping the implementation relatively simple.
The observed execution times align well with the linear time complexity. This confirms that the algorithm is suitable for lexing large numbers of files with predictable performance.

| Ethical Reflection |

Arturo Utrilla Hernández:

Tools like lexical analyzers and syntax highlighters can enhance code quality, education, and productivity. However, they also raise ethical concerns. For instance, such tools could be incorporated into surveillance systems to scan codebases for policy violations without developer consent, or be misused to extract or transform proprietary code. It is essential to use such technologies transparently, respecting intellectual property and promoting responsible usage aligned with the values of the software engineering profession.

Daniel Esteban Hernández García:

It is natural to want the fastest tools, especially when they can make our work more efficient and comfortable. In the case of the parallel lexer that we developed, tests showed that it performs slightly better than the sequential version, with the performance gap becoming more significant as the size of the input increases. However, as someone once pointed out to me, what benefits one person may not always be the best for everyone.

Although we do not have precise measurements of the energy used by the parallel version compared to the sequential one, it is reasonable to assume that it consumes more resources. When scaled up, this increased energy use could have a noticeable environmental impact. As future engineers, we have a responsibility to consider not only performance but also sustainability. From an ethical perspective, we should ask ourselves whether the benefits of parallel processing are worth the potential harm to the environment.

One possible approach is to focus on optimizing the tool to balance performance and energy consumption. This would require more engineering effort and a thoughtful formal thinking process, but it aligns with the ethical responsibility we have to create efficient and environmentally conscious solutions.

Rodrigo Martínez Vallejo:

In reflecting on the ethical implications of our parallel computing findings, I reaffirm that while technology itself is neutral, its applications are not, a duality evident in our results showing parallel execution's speed advantages over sequential processing. As engineers, we must critically weigh these performance gains against their broader costs: the energy trade-offs of multi-core systems, where faster execution may come at environmental expense; the accessibility barriers posed by specialized hardware requirements; and the risk of misuse, where efficiency could enable harmful applications like brute-force attacks. Our project's educational focus aligns with ethical intent, but this demands proactive stewardship. Ultimately, technological progress must be measured not just in speed, but in responsibility.

Ximena Silva Bárcena:

Efficient lexical analysis tools with parallel processing capabilities democratize code analysis while potentially displacing entry-level programming tasks. The ability to process large codebases in milliseconds accelerates development but may devalue fundamental programming skills and contribute to job displacement in the technology sector.

The scalability of these analyzers raises concerns about intellectual property protection and unauthorized analysis of proprietary codebases. Mass analysis capabilities could facilitate IP theft or inadvertent reproduction of copyrighted patterns. Additionally, malicious actors could exploit these tools to identify vulnerabilities across multiple projects simultaneously, emphasizing the need for responsible deployment and ethical guidelines in automated code analysis.