University of Copenhagen
Computer Science Department

# Data-Parallel Compilation
Lexical analysis & Syntax Tree Construction

William Henrich Due (mcj284)
Submitted: 5th of April 2024

**Abstract**

Abstract.

## 1 Introduction

Introduction.

## 2 Theory

Hills paper "Parallel lexical analysis and parsing on the AMT distributed array processor" [1] describes a method to obtain the path in a deterministic finite automata given a input string. This section will describe the theory of this method and extend the it for tokenization.

### 2.1 Data-parallel Lexical Analysis

To explain the theory of parallel lexical analysis we first remind the reader of the definition of a deterministic finite automaton.

**Definition 2.1** (DFA)**.** A deterministic finite automata [2] [3] is given by a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where.

1. $Q$ is the set of states where $|Q| < \infty$.
2. $\Sigma$ is the set of symbols where $|\Sigma| < \infty$.
3. $\delta : \Sigma \times Q \to Q$ is the transition function.
4. $q_0 \in Q$ is the initial state.
5. $F \subseteq Q$ is the set of accepting states.

This definition is fine as is but we will need to reformualte it to develop data-parallel lexical analysis. We would want the definition to use a curried transition function. But for this to hold then the DFA would also have to be total.

**Definition 2.2** (Total DFA)**.** A DFA $(Q, \Sigma, \delta, q_0, F)$ is said to be total if and only if

$$\delta(a, q) \in Q : \forall (a, q) \in \Sigma \times Q$$

If a DFA is total we may use a curried transition function $\delta : \Sigma \to Q \to Q$.

This is needed since else the the function would not be fully defined in the domains $\Sigma$ and $Q$.

The reason for doing so is because if we have any two functions $g = \delta(a)$ and $f = \delta(a')$ then it follows from composition that.

$$g(f(q)) = (g \circ f)(q)$$

This allows for an alternative way of determining if a string can be produced by an DFA. Instead of first evaluating $f(q)$, then $g(f(q))$ and then checking if this state is a member of $F$. We could instead partially apply $\delta$ to the symbols and then compose them to a single function which could be used to determine if a string is valid. This sets the stage for data-parallel lexing, we want to find a way to make the problem into a `map-reduce`. We want to do this because it can be computed using a data-parallel implementation unlike the normal way of traversing a DFA.

For the ability to use a data-parallel `map-reduce` we must have a monoidal structure. Here a set $\Delta$ of all the composed partially applied $\delta$ functions needs to be closed under function composition.

**Proposition 2.1** (DFA Composition Closure)**.** Given a total DFA, the set of all partially applied compositions that is closed under function composition $\Delta : \{Q \to Q\}$. $\Delta$ is the solution to the following equation such that $j$ is smallest index where it holds that $\Delta_j = \Delta_k$ for all $k > j$.

$$\begin{aligned}\Delta_1 &= \{\delta(a) : a \in \Sigma\} \\ \Delta_{i+1} &= \Delta_i \cup \{f \circ g : f, g \in \Delta_i\}\end{aligned}$$

*Proof.* We will start by showing that an solution $\Delta$ exists. First note that $\Delta_i \subseteq \Delta_{i+1}$ since $\Delta_{i+1}$ is a union of $\Delta_i$ and another set. Secondly note that since $|Q| < \infty$ then a finite amount of functions of the form $Q \to Q$ can be created. This leads to at some point alle sets will be equal $\Delta_i = \Delta_{i+1}$ and this point is the solutaion $\Delta$.

For $\Delta$ to be closed under function composition, then for arbitray $f, g \in \Delta$ it must hold that $f \circ g \in \Delta$. As a consequence of this all combinations $\Delta_1$ of a given length must be constructable. To show this we will need to show that every combination of length $i$ or less is found in

(I have to think a little harder.) □

**Corollary 2.1** (DFA monoid)**.** DFA Composition Closure induces a monoid $(\Delta \cup \{id\}, \circ)$ where $id : Q \rightarrow Q$ and $id(q) = q$.

### 2.2 Parallel Tokenization

## 3 Conclusion

Conclusion.

## References


[1] Jonathan M.D Hill. "Parallel lexical analysis and parsing on the AMT distributed array processor". In: *Parallel Computing* 18.6 (1992), pp. 699–714. ISSN: 0167-8191. DOI: https://doi.org/10.1016/0167-8191(92)90008-U. URL: https://www.sciencedirect.com/science/article/pii/016781919290008U.

[2] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. USA: Addison-Wesley Longman Publishing Co., Inc., 2006. ISBN: 0321455363.

[3] Wikipedia contributors. *Deterministic finite automaton — Wikipedia, The Free Encyclopedia*. [Online; accessed 4-February-2024]. 2023. URL: https://en.wikipedia.org/w/index.php?title=Deterministic_finite_automaton&oldid=1192025610.