# Earley Algorithm as Evaluator for the Correctness of Language Expression

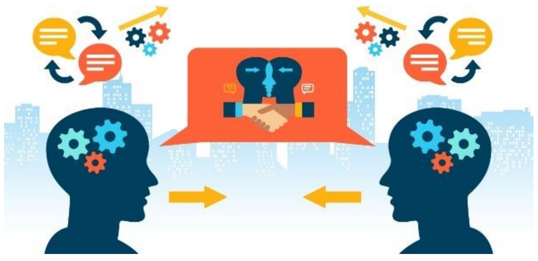
Naufal Prima Yoriko - 13518146[1]
*Program Studi Teknik Informatika*
*Sekolah Teknik Elektro dan Informatika*
*Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia*
*[1]13518146@std.stei.itb.ac.id*, *primayoriko@gmail.com*

***Abstract* — Language always being important part of our daily life since we always use it to communicate with others. However, as life turned out to be more complex time by time, we find out that our need of well-structured language is become more important at some degrees, from writing formal letter, until parsing a computer program. The amount and complexity of the text that being analyzed also increased overtime, and we are in the point that we cannot solve every problem manually, some tasks must be done automatically and computerized. So, figure an effective method with high precision and throughput to solve this problem is a must.**

***Key Words* — Earley Algorithm, Top-Down Dynamic Programming, Formal Language, Right-sided Parser.**

## I. Introduction

Language is a form or medium that used by all living being to communicate with each other. Although it hasn't any concrete or real form, its existence is so important to do things like that, especially for us, humans, as a being with highest level of intelligence. This intelligence make communication, as a method of information propagation, is a crucial thing, maybe one of our basic need, as it says in Maslow's Diagram of Human Needs. Communication is in the middle one, the third level, just after physical needs and secureness needs. By knowing its importance, so we would do all that we've got to transmit the information to others.



Picture 1.1. Language as a form of communication medium (Source: https://www.salesforce.com/ca/blog/2016/10/sales-language-affects-bottom-line.html)

But, transmitting information isn't all the whole thing, since there is some aspect beside of information transmitting that we must take care of. You must know that naturally, us, Indonesian people can't talk (or transmit the information) directly to British people, so same thing American people can't talk directly to Japanese people. There is one common thing, its the language understanding. So, beside of transmission mechanism, we should make sure that, the other side, know the language's rules, or semantics, of the information's language. So, that is the reason why there must be a language that many people know, not must universally, but could ensure that every people of related fields know it well, because we have the rules and semantics that must be obeyed, so that what we called Grammar.

Picture 1.2. An example of language ambiguity (Source: https://www.thoughtco.com/syntactic-ambiguity-grammar-1692179)

Language itself, as I've said, have many rules or semantics on it. However, its expression sometimes have ambiguous meaning, that different people maybe have different interpretation about it. So, it will be more dangerous that if people don't have good understanding with the context, they could be led to the false mean of the expression. Someone with good understanding should be able to identify correctness of the expression or sentence easily, but it will be different story to someone with limited knowledge or extremely someone which just learn a language and want to verify it. Moreover, someone who expert to could find it difficult and time consuming to verify the expression if the text is a very long one, and a person, of course easily to being careless if they become not focus. One last thing, we could say that this method will very slow if done manually. So, we should find a method to evaluating expression in more automatic, fast, and precise.

## II. Language

### *A. Definition and Characteristics*

Language actually have very broad usage, since its not only human way of communicating, but its used by many existences, even abstract existence that is not a living being at all, such as computation machine. So, there is many definitions of a language, so before continue further explanation about language evaluation method, we must ensure that we have same general view about language itself. There are some different mean of language from different source

1. From Wikipedia, language is structured system of communication, and in broader sense is method of communication that involves the use of particularly human-like languages.
2. From Merriam-Webster website, there are many definitions of language such as
   - 2.1. The words, their pronunciation, and the methods of combining them used and understood by a community.
   - 2.2. A systematic means of communicating ideas or feeling
   - 2.3. The suggestion by objects, actions, or conditions of associated ideas or feelings
   - 2.4. a formal system of signs and symbols (such as FORTRAN or a calculus in logic) including rules for the formation and transformation of admissible expressions
3. From Britannica online, language is a system of conventional spoken, manual (signed), or written symbols by means of which human beings, as members of a social group and participants in its culture, express themselves. The functions of language include communication, the expression of identity, play, imaginative expression, and emotional release.

Like what I've mentioned before, the definition of the language very broad as the broadness of its domain. But from here maybe we can tell or conclude some characteristics of a language such

1. Language is something that used to be medium for some sort of communication (include monologue).

2. Language use specific set of symbols that make it more real and rigid, so it could be written and documented well.

3. Language has specific method and rules to ensure its job as communication's medium by specifying how others are supposed to catch the meaning of that language.

### B. *Type of Language*

As I have just said before, language is very broad topic. So, it could be classified into many categorical and different aspects that being compared. So, here I will group language into two parts based on language in programming aspect.

1. Natural Language
   Natural language is language where we usually speak up in daily conversation or generally speaking its evolutionary man-made language to speak up his mind in usual manner and deliver it to another person. So, the language in this category is language that most people have known, like Bahasa Indonesia, English, French, Spanish, Tagalog, etc.

2. Formal Language
   Formal language is language that made for more specific application/field, for example math, computer science, or chemistry, and that language has been recognized by people in that field to be universally used. Formal language has more rigid rule that construct its expression, but aside from its rigidness, the expression is well-formed and easily understand by people who know the rules/grammar. In Computer Science itself, there is specific study that focused in this subject, that is *formal language theory*.

To make it clearer, from those two kinds of language above, there are some difference that are being reason to separate them, they are

1. Ambiguity
   Natural language is far-likely being ambiguous (something like double-meaning from an identical expression) than formal language since formal language has more rigid and well-formed grammar.
2. Redundancy
   To counter its ambiguity, natural language sometimes does extra statement or expression, so its redundant, where formal language is no need for that.
3. Literalness
   Formal language tells us exactly what its meaning from its phrase or expression, no way there is any hidden meaning. This not applicable on natural language since they have many metaphor-like things.

## III. Grammar

### *A. Definition*

Taken from Wikipedia, in linguistics, grammar is the set of structural rules governing the composition of clauses, phrases and words in a natural language. But, language itself, as we have known from previous section, there is formal language too, aside from natural language. So, definition of grammar here maybe to general, so that why there is formal grammar, too.

Formal grammar not too different from grammar. Actually, grammar is formal grammar when the context is not clear, but formal grammar has convention about how we write it (since formal grammar are specifically studied in formal language theory too), and of course made for specifying rules for formal language. However, the essence of both are about how we define rules for the specific language.

### *B. Term and Syntax of Formal Grammar*

Formal language has some terms and syntax that we must understand in order to grasp the formal language theory better. Here is some term that we must know

1. Formal grammar (G) usually represented by G = (N,

E, P, S) notation, where N is non-terminal symbol, E is terminal symbol, P is production rule (usually we refer it as 'rule' only), and S is the 'start' symbol.

2. Start symbol is the symbol that being start point that being root or start point of where our grammar should come. So, the language would be invalid if that comes from another symbol.
3. Non-terminal symbol is a symbol that used to produce other symbol, not a 'actual' symbol that being text/symbol of the language itself, but usually being a specific structure of the language. Non-terminal symbol usually written by fully-capitalized alphabet(s), and used to be short and clear.
4. Terminal symbol is a symbol that being actual part of the symbols that construct the language in a text or expression, and can't be extended into another symbol as it's a 'terminal'. Terminal symbol written by alphabet(s) that contain lower-case character, number(s), or special character(s) such '(', '+', etc.
5. Production rule is a rule or regulation that regulate how expression or sequence of symbols are extended into another expression in order to find goal or desired expression. We could write production rule as

$$LS \rightarrow RS$$

This is mean that LS symbol(s) could be transformed into RS symbol(s).

### *C. Type of Formal Grammar*

There is popular hierarchy about formal grammar in formal language theory that composed by Noam Chomsky in 1956. The hierarchy itself is to show how complex is the language that defined by this grammar. So, there was four types of grammar from the hierarchy, they are

1. Type-0 Grammar, that support recursively enumerable language.
2. Type-1 Grammar, that support context-sensitive language.
3. Type-2 Grammar, that support context-free language.
4. Type-3 Grammar, that support regular language.

## IV. DYNAMIC PROGRAMMING

### *A. Overview*

Dynamic Programming is a mathematical optimization method, also computer programming optimization method at same time. This method help solving many problem that usually have high complexity when doesn't implement this optimization.

Dynamic Programming is usually seen as 'Brute-forcing with memoization' or 'optimization with memory', because its main characteristics is using memory that contains the value of some previous calculation that might be used in the future, and avoid recalculating the same thing again.

### *A. Types*

Dynamic Programming has two kinds of pattern when solving problem, that are

1. Top-Down Approach

   This approach trying to solve problem from the general one, and decompose it to its sub-problem one by one as a part of the general solution that needed. In programming, this is usually done by 'recursive' approach that might be easier than bottom-up, but a little bit complex in its computation.
2. Bottom-Up Approach

   Opposite from top-down, this approach trying to solve from its base case and smallest sub-problem, and then try to build bigger solution from the computed value before until get the general solution. In programming, this is usually done by 'iterative' approach.

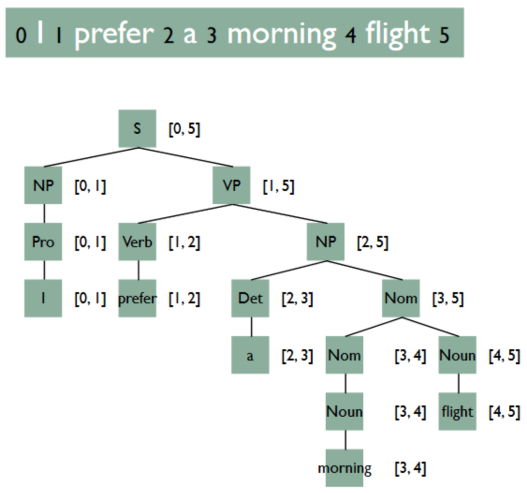
## V. EARLEY ALGORITHM

### *A. Overview*

Earley Algorithm is an algorithm that usually runs for being a parser of a language, even programming languages. As we have discussed before, this parser could understand language by iterating from its grammar. So that would look like it is 'understanding' the language like what human does, although it is just by some kind of 'trial-and-error' by matching the grammar.

So, Earley algorithm have some characteristics which make it stand out from other algorithms such

- Its algorithm top-down approach, or in generalized form it uses a dynamic programming top-down approach. So, this algorithm would run an expression by iterating from its start symbol, and then try possible ways to get the result.



Picture 5.1.1. Illustrate structure of Earley algorithm's parser
(Source: https://cl.lingfil.uu.se/~sara/kurser/5LN455-2014/lectures/5LN455-F5.pdf)

Picture 5.1.1. shows us that the algorithm will start from the start symbol 's' and then make a tree to find possible structure according to the text. This is just an overview, so I will tell the details later.

- Earley Algorithm is left-handed algorithm, or its read and processing text from left side into the right side of a text. If we referring to picture 5.1.1 again, we could say that the algorithm would start by making left branch first until finish, before constructing the right one.
- This algorithm mainly used to parse Context-free Grammar (CFG), but due to its broadness of implementation, it is even could be used to parse Context Sensitive Grammar (CSG).

## *B. Syntax and Structure of the Algorithm*

Before we continue to how Earley algorithm works, we should know the syntax, structure, and also term that used in this algorithm, first. So generally speaking, there will be three data structures that will be build by this algorithm that is

1. States

   State is a structure that represent a state in the process of evaluating a grammar rule. There is a syntax that usually used and looks like being a convention in writing state in Earley algorithm. For example

   $$\boldsymbol{S \rightarrow A \text{ @ } book\ B}, [\boldsymbol{start}, \boldsymbol{end}]$$

   As in every grammar, capitalized alphabets used to be non-terminal symbol, when lower-case one being terminal symbol. 'S' is used as a start non-terminal symbol, too. The different parts are, there are '@' symbol that I meant to be 'big' dot symbol, but I will write it as '@' or 'at' symbol because it is not used here to represent anything. This symbol tell us how far the parse has done. In the example @ written after 'A', so 'A' token has been parsed and the next one that going to be parsed is 'book'. The rule (or state at one point) in Earley have dot, so we usually could say it as *dotted rule*.

   The next difference is [start, end] notation which tell us which range of text that being symbolized by this expression. 'start' here is index of starting part, when 'end' being the ending index of text of this expression. Index here representing one token (terminal/non-terminal) that has ordered from the left side of text. For example, "I have been sleeping since morning" text has 'S' or start terminal with index [0, 6] if the text being zero-indexed.

2. Statelist or Stateset

   Statelist is a collection of ordered states, and since its element are unique, its being a set too and called stateset at the same time. State in statelist are not deleted or modified, instead we always can add new state into this statelist.

3. Chart

   Chart is a set of statelist. There is only single statelist on every single point in the input, from before first word until after last word (we could say single statelist in one point of dot). At a single point of time too, there is only one statelist that is being processed, whilst there is new one that being created too. The chart, we could say it as lookup table of data, that like structure that we use on dynamic programming. So, if the value we want to search already in the chart we can use it, don't need to search it from the start.

Other than three data structures just I've mentioned there should be processes method that will be used, and here we could structure it into three main parts, too. They're

1. The Predictor

   The job of the Predictor is to identify every possible extension of the expression by reading state with first symbol after dot is non-terminal using rules in the grammar and then add the rule into statelist.

2. The Scanner

   The job of the scanner is to identify state with symbol after dot is terminal symbol, so the scanner will add state with dot moved upfront of the terminal symbol into statelist.

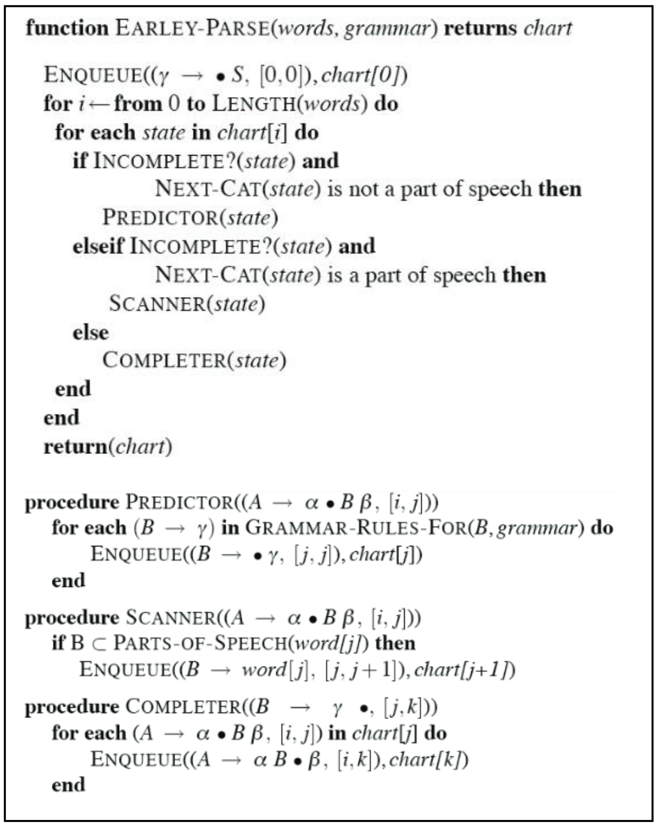
3. The Completer

   The job of the completer is to identify if there is dot symbol that become right-most symbol in the expression (no symbol afterwards in the right-side) or we could say as condition that not covered by the predictor and the scanner, if there is any of it, the completer will reduce the state and recognize it as a complete one.

## *C. Algorithm's Steps*

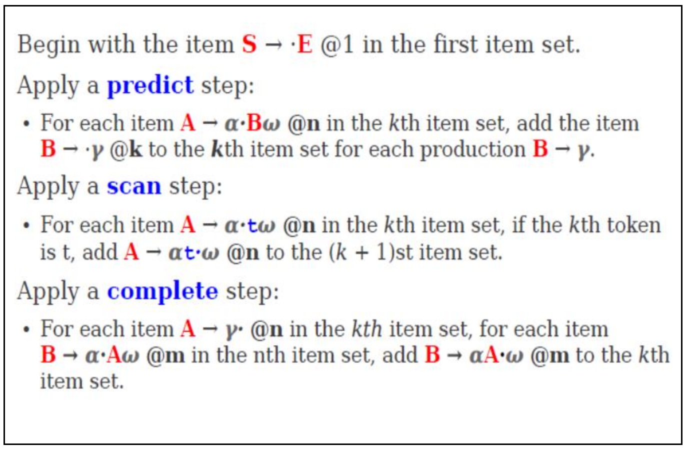
In part B, we have known all of the structure that would be used here. So actually, the pseudo-code of the algorithm is not too complex, a simple one, but might be a little complex to implement it in real program.

So, this is how Earley Algorithm looks like



```
function EARLEY-PARSE(words, grammar) returns chart

  ENQUEUE(($\gamma \rightarrow \bullet S$, [0,0]), chart[0])
  for $i \leftarrow$ from 0 to LENGTH(words) do
    for each state in chart[i] do
      if INCOMPLETE?(state) and
              NEXT-CAT(state) is not a part of speech then
          PREDICTOR(state)
      elseif INCOMPLETE?(state) and
              NEXT-CAT(state) is a part of speech then
          SCANNER(state)
      else
          COMPLETER(state)
    end
  end
  return(chart)

procedure PREDICTOR(($A \rightarrow \alpha \bullet B \beta$, [i, j]))
    for each ($B \rightarrow \gamma$) in GRAMMAR-RULES-FOR(B, grammar) do
        ENQUEUE(($B \rightarrow \bullet \gamma$, [j, j]), chart[j])
    end

procedure SCANNER(($A \rightarrow \alpha \bullet B \beta$, [i, j]))
    if B $\subset$ PARTS-OF-SPEECH(word[j]) then
        ENQUEUE(($B \rightarrow$ word[j], [j, j+1]), chart[j+1])

procedure COMPLETER(($B \rightarrow \gamma \bullet$, [j,k]))
    for each ($A \rightarrow \alpha \bullet B \beta$, [i, j]) in chart[j] do
        ENQUEUE(($A \rightarrow \alpha B \bullet \beta$, [i,k]), chart[k])
    end
```

This is general and well-known pseudo-code of Earley Algorithm that widely (fetched from Edinburgh, Uppsala, and Washington University Slide) used to be implemented in various language to make different applications of parser.



Begin with the item **S** → ·**E** @1 in the first item set.

Apply a **predict** step:

- For each item **A** → **α**·**B**ω @**n** in the *k*th item set, add the item **B** → ·γ @**k** to the **k**th item set for each production **B** → γ.

Apply a **scan** step:

- For each item **A** → **α**·**t**ω @**n** in the *k*th item set, if the *k*th token is t, add **A** → **αt**·ω @**n** to the (*k* + 1)st item set.

Apply a **complete** step:

- For each item **A** → γ· @**n** in the *kth* item set, for each item **B** → **α**·**A**ω @**m** in the nth item set, add **B** → **αA**·ω @**m** to the kth item set.

This is another great explanation of the task about three method that made Earley algorithm (predictor, scanner, and completer) that fetched from Stanford University Slides.

### *D. Complexity of the Algorithm*

Earley algorithm's complexity is O($n^3$) in the worst or general case without restriction, but can perform better in unambiguous grammar in O($n^2$), even in special case LR(k) (this is an advanced parsing technique, so we don't push it further) could be O(n) complexity. So overall, this algorithm complexity somewhat good, and usually preferred to parse simplified natural language because this algorithm has diverse implementation. However, it's implementation on programming language start to being left because there are more powerful algorithms for this purpose (like LL, LR that I've mentioned before).

### *E. Pros and Cons of the Algorithm*

From what we have discussed from the algorithm, you should have big picture of how this algorithm works. So, we can conclude some of its algorithm pros or strength such as

- Earley algorithm could identify broad type of languages and grammars, such ambiguous grammar which many of algorithm out there can't identify it (so it can parse natural language until some degree).
- Relatively fast for parsing non-programming language.
- It wouldn't explore path that doesn't lead to solution or doesn't valid at all.
- The grammar that used doesn't need to be in special format (doesn't like CYK algorithm that need grammar written in Chompsky Normal Form).

Aside from its strength, there should be some weakness or cons since this algorithm isn't perfec t. So, the cons of this algorithm

- Relatively slow for parsing programming language (since there is more powerful and complex algorithm for this use, such as LR parser that could run parsing in O(n) complexity).
- Valid path that doesn't match with input text might be explored.

## VI. IMPLEMENTATION OF EARLEY ALGORITHM

There are many implementations of this algorithm, but here I'll show one of many implementations that widely used as the demonstration of Earley algorithm, it is Simplified English. But actually, the different part of one with another language should be its grammar should be only the grammar that being used. So, here are the grammar's rules that being used:

```
S -> NP VP
NP -> NP PP
NP -> Noun
VP -> Verb NP
VP -> VP PP
PP -> Prep NP
Noun -> Hasan
Noun -> Basketball
Noun -> Field
Verb -> played
Prep -> on
```

So, the text that I'm going to show how to parse it, it is "Hasan played Basketball on Field", and here are the steps of implementation (the notation is same as we have discussed so far and also in addition with the process/method that handle the state (being one of predictor, scanner, or completer)):

```
    S0: [($ -> @ S, [0, 0]) start state,
    (S -> @ NP VP, [0, 0]) predictor,
    (NP -> @ NP PP, [0, 0]) predictor,
    (NP -> @ Noun, [0, 0]) predictor,
    (Noun -> @ Hasan, [0, 0]) predictor,
    (Noun -> @ Basketball, [0, 0]) predictor,
    (Noun -> @ Field, [0, 0]) predictor]

    S1: [(Noun -> Hasan @, [0, 1]) scanner,
    (NP -> Noun @, [0, 1]) completer,
    (S -> NP @ VP, [0, 1]) completer,
    (NP -> NP @ PP, [0, 1]) completer,
    (VP -> @ Verb NP, [1, 1]) predictor,
    (VP -> @ VP PP, [1, 1]) predictor,
    (PP -> @ Prep NP, [1, 1]) predictor,
    (Verb -> @ played, [1, 1]) predictor,
    (Prep -> @ on, [1, 1]) predictor]

    S2: [(Verb -> played @, [1, 2]) scanner,
    (VP -> Verb @ NP, [1, 2]) completer,
    (NP -> @ NP PP, [2, 2]) predictor,
    (NP -> @ Noun, [2, 2]) predictor,
    (Noun -> @ Hasan, [2, 2]) predictor,
    (Noun -> @ Basketball, [2, 2]) predictor,
    (Noun -> @ Field, [2, 2]) predictor]

    S3: [(Noun -> Basketball @, [2, 3]) scanner,
    (NP -> Noun @, [2, 3]) completer,
    (VP -> Verb NP @, [1, 3]) completer,
    (NP -> NP @ PP, [2, 3]) completer,
    (S -> NP VP @, [0, 3]) completer,
    (VP -> VP @ PP, [1, 3]) completer,
    (PP -> @ Prep NP, [3, 3]) predictor,
    ($ -> S @, [0, 3]) completer,
    (Prep -> @ on, [3, 3]) predictor]

    S4: [(Prep -> on @, [3, 4]) scanner,
    (PP -> Prep @ NP, [3, 4]) completer,
    (NP -> @ NP PP, [4, 4]) predictor,
    (NP -> @ Noun, [4, 4]) predictor,
    (Noun -> @ Hasan, [4, 4]) predictor,
    (Noun -> @ Basketball, [4, 4]) predictor,
    (Noun -> @ Field, [4, 4]) predictor]

    S5: [(Noun -> Field @, [4, 5]) scanner,
    (NP -> Noun @, [4, 5]) completer,
    (PP -> Prep NP @, [3, 5]) completer,
    (NP -> NP @ PP, [4, 5]) completer,
    (NP -> NP PP @, [2, 5]) completer,
    (VP -> VP PP @, [1, 5]) completer,
    (PP -> @ Prep NP, [5, 5]) predictor,
    (VP -> Verb NP @, [1, 5]) completer,
    (NP -> NP @ PP, [2, 5]) completer,
    (S -> NP VP @, [0, 5]) completer,
    (VP -> VP @ PP, [1, 5]) completer,
    (Prep -> @ on, [5, 5]) predictor,
($ -> S @, [0, 5]) completer]
```

```
Success: true
```

As the length of the text is 5 symbols (Hasan, played, Basketball, on, Field) so if this is being general case in Earley algorithm, this should take about $5^3$ steps, or around 125 steps.

## VII. Thank-You Note

I, as writer, say many thanks and gratitude to Allah Swt. as our Creator, because of Him, I am still alive and healthy until nowadays, and had chance to wrote and finish this paper. Furthermore, many thanks to Mrs. Nur Ulfa Maulidevi, Mr. Rinaldi Munir, and Mrs. Masayu Leylia Khodra as my lecturer that always taught us, even in this pandemic in *Strategi Algoritma* lesson. I hope this knowledge will everlasting and could be taught to someone else in the future.

## Video Link at YouTube

As part of supporting the explanation of this paper, I have included video link on youtube that can be accessed from here https://youtu.be/kjyvpdM5pMU.

## References

[1] https://www.britannica.com/topic/language, accessed on 1st – 2nd of May 2020.
[2] https://en.wikipedia.org/wiki/Language, accessed on 1st – 2nd of May 2020.
[3] https://en.wikipedia.org/wiki/Formal_language, accessed on 1st – 2nd of May 2020.
[4] https://www.merriam-webster.com/dictionary/language, accessed on 1st – 2nd of May 2020.
[5] https://cl.lingfil.uu.se/~sara/kurser/5LN455-2014/lectures/5LN455-F5.pdf, accessed on 1st – 2nd of May 2020.
[6] https://www.cs.jhu.edu/~jason/465/PDFSlides/lect10-earley.pdf, accessed on 1st – 2nd of May 2020.
[7] https://courses.cs.washington.edu/courses/cse401/16wi/lectures/10-CYK-Earley-Disambig-wi16.pdf, accessed on 1st – 2nd of May 2020.
[8] https://courses.washington.edu/ling571/ling571_fall_2010/slides/parsing_earley.pdf, accessed on 1st – 2nd of May 2020.
[9] https://medium.com/100-days-of-algorithms/day-94-earley-parser-3fffdb33edc7, accessed on 1st – 2nd of May 2020.
[10] http://www.martinbroadhurst.com/earleys-algorithm-in-c.html, accessed on 1st – 2nd of May 2020.
[11] http://www.martinbroadhurst.com/context-free-grammar-in-c.html, accessed on 3rd of May 2020.
[12] https://en.wikipedia.org/wiki/Grammar, accessed on 3rd of May 2020.
[13] https://en.wikipedia.org/wiki/Chomsky_hierarchy, accessed on 3rd of May 2020.
[14] https://en.wikipedia.org/wiki/Formal_grammar, accessed on 3rd of May 2020.
[15] https://runestone.academy/runestone/books/published/thinkcspy/GeneralIntro/FormalandNaturalLanguages.html, accessed on 3rd of May 2020.
[16] https://web.stanford.edu/class/archive/cs/cs143/cs143.1128/lectures/07/Slides07.pdf, accessed on 4th of May 2020.

## Statement

With this statement, I hereby declare that this paper is my own write, not a adaptation nor a copy, and nor a translation too, from other papers, so there isn't any form of plagiarism here.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bekasi, 4 Mei 2020

[illegible]

Naufal Prima Yoriko - 13518146