

UNIVERSITY OF COPENHAGEN
Computer Science Department
Parallel Parsing using Futhark
Subtitle

Author: William Henrich Due
Advisor: Troels Henriksen
Submitted: June 12, 2023

Contents

1	Introduction	1
2	Theory	1
2.1	<i>LL(k)</i> Grammar	1
2.2	<i>LLP(q, k)</i> Grammar	3
3	Implementation	3
4	Testing	3
5	Conclusion	3

1 Introduction

2 Theory

2.1 *LL(k)* Grammar

For the construction of a *LLP(q, k)* parser generator the construction of first and follow-set[1, p. 5] and a *LL(k)* parser generator is needed. A short explanation of the construction of a *LL(k)* parser generator will be given since in the research of this project $k = 1$ was quite often explained but never $k > 1$ in a manner the author found understandable.

Definition 2.1 (Truncated product). Let $G = (N, T, P, S)$ be a grammar and $A, B \subseteq P((N \cup T)^*)$ be sets of symbol strings. The truncated product is defined in the following way.

$$A \odot_k B \stackrel{\text{def}}{=} \{\alpha\beta : \alpha \in A, \beta \in B, |\alpha\beta| \leq k\}$$

Definition 2.2 (Nonempty substring pairs). Let $G = (N, T, P, S)$ be a grammar and $\omega \in (N \cup T)^*$ be a symbol string. The set of every nonempty way to split ω into two substrings is defined to be.

$$\varphi(\omega) \stackrel{\text{def}}{=} \{(\alpha, \beta) : \alpha \in (T \cup N)^+, \beta \in (T \cup N)^+, \alpha\beta = \omega\}$$

Algorithm 2.1 (Solving $FIRST_k$ set). Let $G = (N, T, P, S)$ be a grammar, the first-sets can be solved as followed.

$$\begin{aligned}
FIRST_k(\epsilon) &= \{\epsilon\} \\
FIRST_k(t) &= \{t\} \\
FIRST_k(A) &= \bigcup_{\delta: A \Rightarrow \delta \in P} FIRST_k(\delta) \\
FIRST_k(\omega) &= \bigcup_{(\alpha, \beta) \in \varphi(\omega)} FIRST_k(\alpha) \odot_k FIRST_k(\beta)
\end{aligned}$$

This may result in an infinite loop if implemented as is so fixed point iteration is used. Let $\mathcal{M} : N \rightarrow P(T^*)$ be a surjective function i.e. a dictionary which maps nonterminals to sets of terminal strings. $FIRST'_k$ is then the following modified version of $FIRST_k$.

$$\begin{aligned}
FIRST'_k(\epsilon, \mathcal{M}) &= \{\epsilon\} \\
FIRST'_k(t, \mathcal{M}) &= \{t\} \\
FIRST'_k(A, \mathcal{M}) &= \mathcal{M}(A) \\
FIRST'_k(\omega, \mathcal{M}) &= \bigcup_{(\alpha, \beta) \in \varphi(\omega)} FIRST'_k(\alpha, \mathcal{M}) \odot_k FIRST'_k(\beta, \mathcal{M})
\end{aligned}$$

This function is then used to solve for a $FIRST_k$ function for a fixed k with fixed point iteration the following way.

1. Initialize a dictionary \mathcal{M}_0 such that $\mathcal{M}_0(A) = \emptyset$ where $\forall A \in N$.
2. A new dictionary $\mathcal{M}_{i+1} : N \rightarrow P(T^*)$ is constructed by $\mathcal{M}_{i+1}(A) = \bigcup_{\delta: A \Rightarrow \delta \in P} FIRST'_k(\delta, \mathcal{M}_i)$ for all $A \in N$ where \mathcal{M}_i is the last dictionary that was constructed.
3. If $\mathcal{M}_{i+1} = \mathcal{M}_i$ then terminate the algorithm terminates else recompute step 2.

Let \mathcal{M}_f be the final dictionary after the algorithm terminates then it holds that $FIRST_k(\omega) = FIRST'_k(\omega, \mathcal{M}_f)$ if k stays fixed.

Algorithm 2.2 (Solving $FOLLOW_k$ set). Let $G = (N, T, P, S)$ be a grammar, the follow-sets can be solved as followed.

$$FOLLOW_k(A) = \bigcup_{B: B \Rightarrow \alpha A \beta \in P} FIRST_k(\beta) \odot_k FOLLOW_k(B)$$

Once again this may not terminate so fixed point iteration can be used with following altered $FOLLOW_k$ and letting $\mathcal{M} : N \rightarrow P(T^*)$ be a surjective function.

$$FOLLOW_k(A, \mathcal{M}) = \bigcup_{B: B \Rightarrow \alpha A \beta \in P} FIRST_k(\beta) \odot_k \mathcal{M}(B)$$

This $FOLLOW_k$ function for a fixed k can then be computed using the following algorithm.

1. Extend grammar $G = (N, T, P, S)$ using $G' = (N', T', P', S') = (N' \cup \{S'\}, T' \cup \{\square\}, P \cup \{P \rightarrow S\square^k\}S')$.
2. Initialize a dictionary \mathcal{M}_0 such that $\mathcal{M}_0(A) = \emptyset$ where $\forall A \in N \setminus \{S\}$ and $\mathcal{M}_0(S) = \{\square^k\}$.
3. A new dictionary $\mathcal{M}_{i+1} : N \rightarrow P(T^*)$ is constructed by $\mathcal{M}_{i+1}(A) = \bigcup_{B: B \Rightarrow \alpha A \beta \in P} FIRST_k(\beta) \odot_k \mathcal{M}_i(B)$ for all $A \in N$ where \mathcal{M}_i is the last dictionary.
4. If $\mathcal{M}_{i+1} \neq \mathcal{M}_i$ then recompute step 3.
5. Let \mathcal{M}_f be the final dictionary after step 4. is completed. Let \mathcal{M}_u be another dictionary where $\mathcal{M}_u(A) = \{\alpha : \alpha\square^* \in \mathcal{M}_f(A)\}$ for all $A \in N \setminus \{S'\}$

It then holds that $FOLLOW_k(A) = \mathcal{M}_u(A)$ if k stays fixed for grammar G .

2.2 $LLP(q, k)$ Grammar

3 Implementation

4 Testing

5 Conclusion

References

- [1] Ladislav Vagner and Bořivoj Melichar. “Parallel LL parsing”. In: *Acta Informatica* 44.1 (Apr. 2007), pp. 1–21. ISSN: 1432-0525. DOI: [10.1007/s00236-006-0031-y](https://doi.org/10.1007/s00236-006-0031-y). URL: <https://doi.org/10.1007/s00236-006-0031-y>.