

Sequentially Indexed Grammars

Jan van Eijck

CWI and ILLC, Amsterdam, Uil-OTS, Utrecht

To appear in the *Journal of Logic and Computation*

Abstract

This paper defines the grammar class of sequentially indexed grammars (SIGs) that results of a change in the index stack handling mechanism of indexed grammars [Aho68, Aho69]. SIGs are different from linear indexed grammars [Gaz88] (the rule format is simpler) and they generate a strictly larger language class. We give a polynomial algorithm for parsing with SIGs that is a rather straightforward extension of the Earley algorithm for parsing with context free grammars. SIGs are attractive because of the simple rule format, the natural correspondence between indices and traces, and the perspicuity of the parsing scheme.

Keywords: Deductive parsing, context free grammars, indexed languages, nested stack automata, Earley parsing algorithm, polynomial parsing.

1 Introduction

Indexed grammars [Aho68] result from adding a stack index mechanism to context free grammars.

Definition 1 *Indexed grammars are quadruples $G = (N, T, P, S)$, where N is a finite nonterminal alphabet, T is a finite terminal alphabet, $N \cap T = \emptyset$, P is a finite set of productions of the form (X, α) with $X \in N \cup N^\circ$, $\alpha \in (N \cup N^\circ \cup T)^*$, where N° is the set of all X_Y with $X, Y \in N$, and $S \in N$ is the start symbol.*

A production (X, α) of an indexed grammar is written as $X \rightarrow \alpha$.

In what follows, we use $[]$ for the empty stack, and $X : \zeta$ for the stack with top element X . A stack $X_1 : X_2 : \dots : X_n : []$ will be written in the usual list notation, as $[X_1, \dots, X_n]$, or as X_1, \dots, X_n .

Let $G = (N, T, P, S)$ be an indexed grammar. A pair $(X, [X_1, \dots, X_n])$, with $X, X_1, \dots, X_n \in N$ is called an indexed nonterminal. Indexed nonterminals are written as $X_{X_1 \dots X_n}$. Let N^\bullet be the set of all $X_{X_1 \dots X_n}$, with $X, X_1, \dots, X_n \in N$. Then a sentential form for G is a string α in $(N^\bullet \cup T)^*$.

To define the one step derivation relation, we need a preliminary definition:

Definition 2 If $\delta \in (N \cup N^\circ \cup T)^*$ and $\zeta \in N^*$, then δ_ζ is given by the following recursion:

$$\begin{aligned}\epsilon_\zeta &= \epsilon \\ (w : \delta)_\zeta &= w : \delta_\zeta \text{ if } w \in T, \\ (Y : \delta)_\zeta &= Y_\zeta : \delta_\zeta \text{ if } Y \in N, \\ (Y_Z : \delta)_\zeta &= Y_{(Z:\zeta)} : \delta_\zeta \text{ if } Y_Z \in N^\circ,\end{aligned}$$

Note that as a special case of this, we have that $(Y_Z)_\theta = Y_{Z:\theta}$.

Using the definition of δ_ζ , we can define one step derivations:

Definition 3 Let α, β be sentential forms for indexed grammar G . Then $\alpha \xrightarrow{G} \beta$ iff

1. $\alpha = \gamma_1 X_\zeta \gamma_2$, $X \rightarrow \delta$ is a production of the grammar, and $\beta = \gamma_1 \delta_\zeta \gamma_2$, or,
2. $\alpha = \gamma_1 X_{(Y:\zeta)} \gamma_2$, $X_Y \rightarrow \delta$ is a production of the grammar, and $\beta = \gamma_1 \delta_\zeta \gamma_2$

In terms of this, $\alpha \xrightarrow{G^*} \beta$ is defined in the usual way. This definition is equivalent to the definition in [Aho68].

2 Example Grammars, Example Derivations

An example of an indexed grammar:

$$\begin{aligned}S &\rightarrow A_X \\ A &\rightarrow aA_Y \mid B \\ B_Y &\rightarrow bBc \\ B_X &\rightarrow \epsilon\end{aligned}$$

This grammar generates the non-CF language $\{a^n b^n c^n \mid n \in \mathbb{N}\}$.

$$\begin{aligned}S &\Rightarrow A_X \\ &\Rightarrow aA_{YX} \\ &\Rightarrow aaA_{YYX} \\ &\Rightarrow aaB_{YYX} \\ &\Rightarrow aabB_{YXc} \\ &\Rightarrow aabbB_Xcc \\ &\Rightarrow aabbcc\end{aligned}$$

Here is another example.

$$\begin{aligned}
S &\rightarrow A_X \\
A &\rightarrow aA_Y \mid bA_Z \mid B \\
B_Y &\rightarrow Ba \\
B_Z &\rightarrow Bb \\
B_X &\rightarrow \epsilon
\end{aligned}$$

This grammar generates the non-CF language $\{ww \mid w \in \{a,b\}^*\}$.

$$\begin{aligned}
S &\Rightarrow A_X \\
&\Rightarrow aA_{YX} \\
&\Rightarrow aaA_{YYX} \\
&\Rightarrow aabA_{ZYYX} \\
&\Rightarrow aabB_{ZYYX} \\
&\Rightarrow aabB_{YYX}b \\
&\Rightarrow aabB_{YX}ab \\
&\Rightarrow aabB_Xaab \\
&\Rightarrow aabaab
\end{aligned}$$

Note that in this and the previous example, the index X serves the purpose of triggering the erasing rule for B . Suppose in the last grammar we had the production $B \rightarrow \epsilon$ instead of $B_X \rightarrow \epsilon$. Then we could have derived:

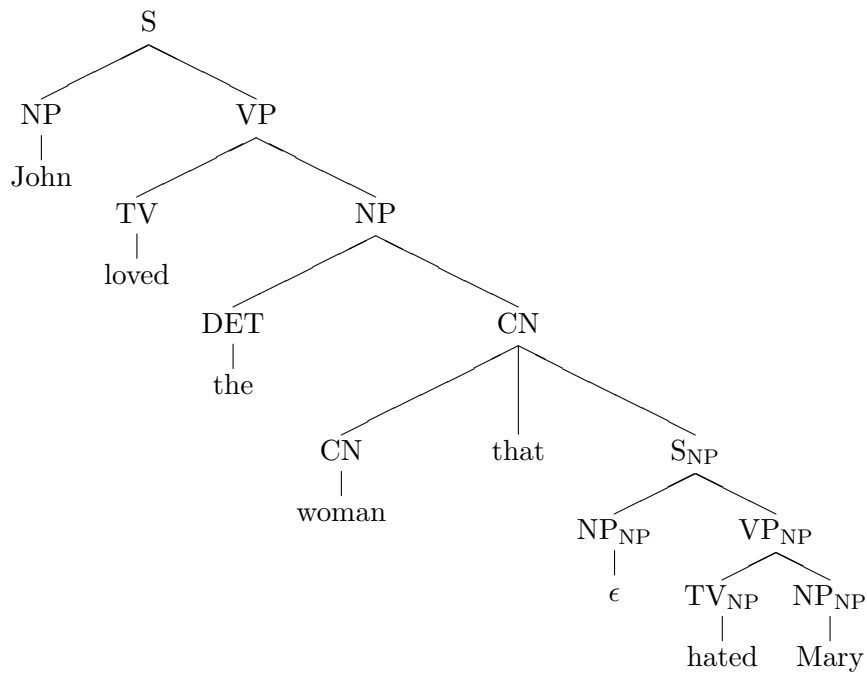
$$\begin{aligned}
S &\Rightarrow A_X \\
&\Rightarrow aA_{YX} \\
&\Rightarrow aaA_{YYX} \\
&\Rightarrow aabA_{ZYYX} \\
&\Rightarrow aabB_{ZYYX} \\
&\Rightarrow aabB_{YYX}b \\
&\Rightarrow aabb
\end{aligned}$$

This is because $B_{YYX} \Rightarrow \epsilon$ is a one-step derivation with rule $B \rightarrow \epsilon$, for the index stack gets copied on all the nonterminals in the righthand side of a rule and disappears on all terminals.

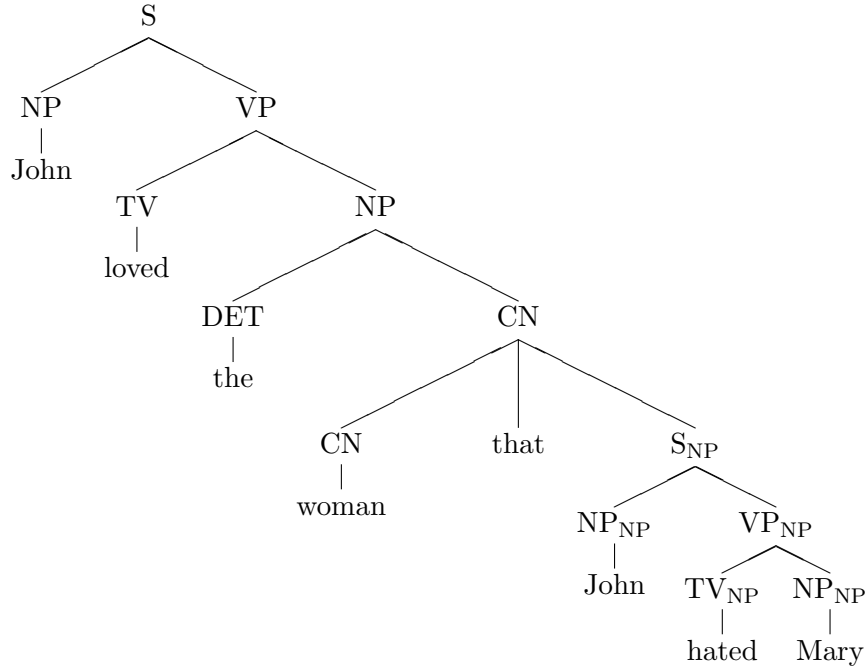
A grammar for a toy fragment of natural language:

$S \rightarrow NP VP$
 $VP \rightarrow TV NP \mid \text{talked} \mid \text{smiled}$
 $NP \rightarrow DET CN \mid \text{John} \mid \text{Mary}$
 $TV \rightarrow \text{loved} \mid \text{hated}$
 $DET \rightarrow \text{the} \mid \text{some}$
 $CN \rightarrow \text{man} \mid \text{woman} \mid \text{CN that } S_{NP}$
 $NP_{NP} \rightarrow \epsilon$

This grammar allows us to illustrate why indexed grammars are not quite what we want for natural language analysis. The grammar uses the index mechanism to define a relative sentence as a sentence with an NP gap in it. When parsing with this grammar using the stack mechanism of indexed grammars, this gap is propagated *everywhere* in the relative sentence, due to the stack copying mechanism in the definition of indexed grammar derivations:



Thus, we will also get:



3 Sequentially Indexed Grammars

Sequentially indexed grammars use indices that get pushed to an *arbitrary* nonterminal in the righthand side of a production. Sequentially indexed grammars look just like indexed grammars, but there are some restrictions on the production rule format, and the definition of *derivation* is slightly different.

Definition 4 *Sequentially indexed grammars are quadruples $G = (N, T, P, S)$, where N is a finite nonterminal alphabet, T is a finite terminal alphabet, $N \cap T = \emptyset$, P is a finite set of productions of the form (X, α) with $X \in N \cup N^\circ$, $\alpha \in (N \cup N^\circ \cup T)^*$, where N° is the set of all X_Y with $X, Y \in N$, and $S \in N$ is the start symbol, subject to the following restrictions:*

- *If $S \rightarrow \epsilon \in P$, then S (without index) does not occur in the righthand side of any rule in P .*
- *If $A \in N$ and $A \neq S$ then $A \rightarrow \epsilon \notin P$.*
- *If $A \rightarrow \alpha B_X \beta \in P$ then either α or β contains a symbol $\sigma \notin N^\circ$.*

So $S \rightarrow \epsilon$ may be in P , but all other rules with righthandside ϵ in P should have a lefthand side with an index. Additionally, rules with indexed symbols in their righthand side should have at least one non-indexed symbol in their righthand side too.

The intuition behind the restrictions on the rule format is that we want an indexed symbol A_X in the righthand side of a production to express a link to the context in which A_X appears. For that it is needed and sufficient that A_X has a definite context, i.e., that the context of A_X contains at least one element that is not itself in need of a context.

The following definition uses list concatenation. If ζ is the result of concatenating ζ_1 and ζ_2 , we denote this as $\zeta = \zeta_1 ++ \zeta_2$.

Definition 5 *If $\delta \in (N \cup N^\circ \cup T)^*$ and $\zeta \in N^*$, then $(\delta)_\zeta$ is the subset of $(N^\bullet \cup T)^*$ defined recursively as:*

$$\begin{aligned} (\epsilon)_\square &= \{\epsilon\} \\ (\epsilon)_\zeta &= \emptyset \text{ if } \zeta \neq \square, \\ (w : \delta)_\zeta &= \{w : \delta' \mid \delta' \in \delta_\zeta\} \text{ if } w \in T, \\ (C : \delta)_\zeta &= \{C_{\zeta_1} : \delta' \mid \zeta = \zeta_1 ++ \zeta_2, \delta' \in \delta_{\zeta_2}\} \text{ if } C \in N \\ (C_Y : \delta)_\zeta &= \{C_{Y:\zeta_1} : \delta' \mid \zeta = \zeta_1 ++ \zeta_2, \delta' \in \delta_{\zeta_2}\} \text{ if } C_Y \in N^\circ. \end{aligned}$$

The relation of one-step derivation is defined in terms of $(\delta)_\zeta$, as follows:

Definition 6 *Let α, β be sentential forms for sequentially indexed grammar G . Then $\alpha \xRightarrow{G} \beta$ iff*

1. $\alpha = \gamma_1 B_\zeta \gamma_2$, $B \rightarrow \delta$ is a production of the grammar, and $\beta = \gamma_1 \delta' \gamma_2$, where δ' in $(\delta)_\zeta$, or
2. $\alpha = \gamma_1 B_{(Y:\zeta)} \gamma_2$, $B_Y \rightarrow \delta$ is a production of the grammar, and $\beta = \gamma_1 \delta' \gamma_2$, where $\delta' \in (\delta)_\zeta$.

In derivations with sequentially indexed grammars, stacks are never allowed to disappear, and stacks are never allowed to get duplicated. In particular, a production $B_Y \rightarrow \epsilon$ will not allow a one-step derivation like $B_{YYX} \Rightarrow \epsilon$, and a production $B \rightarrow CD$ will not allow a one-step derivation like $B_{YYX} \Rightarrow C_{YYX} D_{YYX}$ (but it will allow $B_{YYX} \Rightarrow C_{YYX} D$, $B_{YYX} \Rightarrow C_{YY} D_X$, $B_{YYX} \Rightarrow C_Y D_{YX}$, and $B_{YYX} \Rightarrow CD_{YYX}$). A production like $B_X \rightarrow \epsilon$ can lead to a one-step derivation $B_X \Rightarrow \epsilon$. This effectively treats X as a ‘trace’.

Sequentially indexed grammars are different from an earlier proposal for a restricted form of indexed grammars (called linear indexed grammars), in [Gaz88]. Gazdar proposed to use index lists that get copied to a single nonterminal in the righthand sides of productions, but in such a way that this heir-nonterminal has to be indicated in the rule. Below we will see that the sequentially indexed grammar formalism has greater expressive power than linear indexed grammars.

4 Modification of Earlier Example Grammars

Because of the restrictions on the rule format we have to modify the earlier grammars for $\{a^n b^n c^n \mid n \in \mathbb{N}\}$ and $\{ww \mid w \in \{a, b\}^*\}$ somewhat. We can make of the fact that stacks are not allowed to disappear.

Here is the earlier example grammar for $\{a^n b^n c^n \mid n \in \mathbb{N}\}$.

$$\begin{aligned} S &\rightarrow A_X \\ A &\rightarrow aA_Y \mid B \\ B_Y &\rightarrow bBc \\ B_X &\rightarrow \epsilon \end{aligned}$$

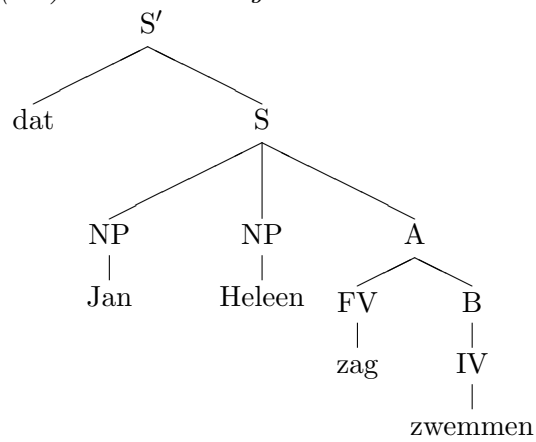
5 Dutch Cross Serial Dependencies

A sequentially indexed grammar for Dutch cross serial dependencies (see, e.g., [BKPZ82]):

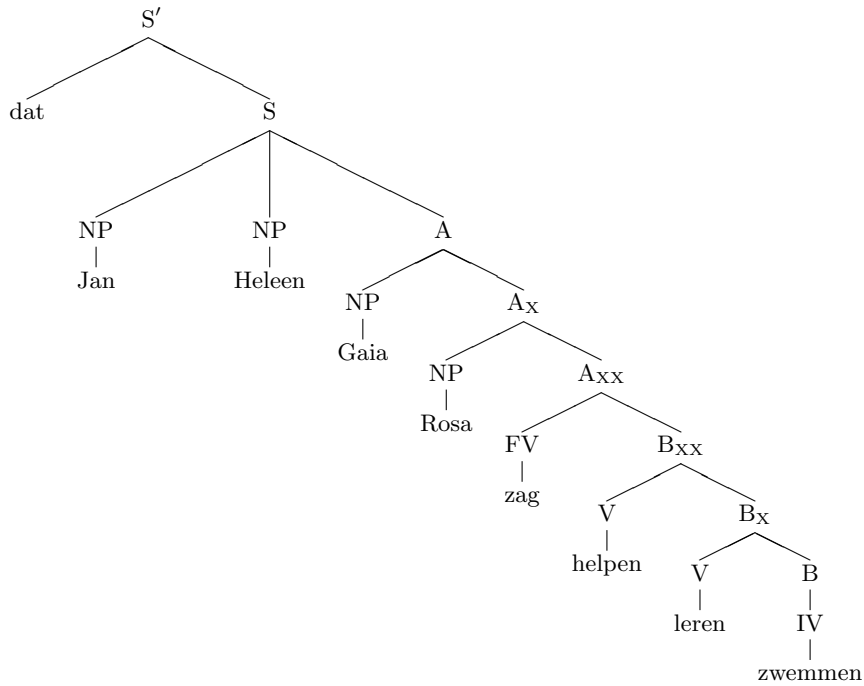
- $S' \rightarrow \text{dat } S$
- $S \rightarrow \text{NP NP A}$
- $\text{NP} \rightarrow \text{Jan} \mid \text{Heleen} \mid \text{Gaia} \mid \text{Rosa}$
- $A \rightarrow \text{NP } A_X$
- $A \rightarrow \text{FV B}$
- $\text{FV} \rightarrow \text{zag} \mid \text{hoorde} \mid \text{liet} \mid \text{leerde} \mid \text{hielp}$
- $B_X \rightarrow \text{V B}$
- $B \rightarrow \text{IV}$
- $V \rightarrow \text{zien} \mid \text{horen} \mid \text{laten} \mid \text{leren} \mid \text{helpen}$
- $\text{IV} \rightarrow \text{zwemmen} \mid \text{lopen} \mid \text{rennen}$

This gives the following parses for Dutch subordinate clauses:

(dat) Jan Heleen zag zwemmen.



(dat) Jan Heleen Gaia Rosa zag helpen leren zwemmen.



6 Deductive Earley-style Parsing with CF Grammars

Earley items for context free parsing [Ear70] are of the form

$$i, A \rightarrow \alpha \bullet \beta, j.$$

$A \rightarrow \alpha\beta$ is a grammar production. \bullet indicates the part of its righthand side that was recognized so far, i points to the position in the input where the rule $A \rightarrow \alpha\beta$ was invoked (the position of the parent node), and j points to the position in the input that recognition has reached.

If an input string of terminals has length n , we denote that string as w_0, \dots, w_{n-1} . After recognition of j terminals (i.e., after recognition of w_0, \dots, w_{j-1}), the recognition pointer has value j .

There is one axiom. It has the form

$$0, S' \rightarrow \bullet S, 0$$

where S is the start symbol and S' is a new start symbol.

There is one goal. It has the form

$$0, S' \rightarrow S \bullet, n$$

where S, S' are as in the axiom, and n is the length of the input. This goal item says that the rule $S' \rightarrow S$ was invoked at the start of the input, n input symbols were recognized (i.e., all of the input was recognized), and the righthand side of rule $S' \rightarrow S$ was completed.

Scanning The scanning rule for Earley parsing with CF grammars is the rule that shifts the bullet across a terminal:

$$\frac{i, A \rightarrow \alpha \bullet w_j \beta, j}{i, A \rightarrow \alpha w_j \bullet \beta, j + 1}$$

Prediction The prediction rule for Earley parsing is the rule that initializes a new rule $B \rightarrow \gamma$ on the basis of a premise indicating that B is expected at the current point in the input:

$$\frac{i, A \rightarrow \alpha \bullet B \beta, j}{j, B \rightarrow \bullet \gamma, j} B \rightarrow \gamma$$

Completion The completion rule for Earley parsing is the rule that shifts the bullet across a non-terminal. It has two premises:

$$\frac{i, A \rightarrow \alpha \bullet B \beta, k \quad k, B \rightarrow \gamma \bullet, j}{i, A \rightarrow \alpha B \bullet \beta, j}$$

Setting up the Deduction Engine Follow the set-up of [SSP95]. A chart plus agenda is a pair of item lists. Use the agenda for those items that have been proved, but whose direct consequences have not yet been derived, Use the chart for the proved items the direct consequences of which have also been computed.

The Parsing Process

- Initialisation: empty chart, and list of all axioms on the agenda.
- Next, successively tackle each item I on the agenda:
 - add the direct consequences of I to the agenda.
 - move I from the agenda to the chart (as the direct consequences of I have been computed).
- Finally, check whether a goal item has been found, and return the list of goal items.

7 Adaptation for Parsing with Indexed Grammars

Adapting this for parsing with indexed grammars, we extend the nonterminals with stacks. The item format becomes

$$i, A_\eta \rightarrow \alpha \bullet \beta, j,$$

with $\eta \in N^*$, $\alpha, \beta \in (N^\bullet \cup T)^*$. The item $i, A_\eta \rightarrow \alpha \bullet \beta, j$ indicates that at the point of invocation of rule $A \rightarrow \alpha\beta$, the index stack has shape η , or, in case $\eta = (X : \eta')$, that at the point of invocation of rule $A_X \rightarrow \alpha\beta$, the index stack has shape η' .

Again, there is one axiom. It takes the shape

$$0, S'_\square \rightarrow \bullet S_\square, 0,$$

indicating that the stack component is empty at the beginning of the parse.

There is a single goal, The goal shape becomes:

$$0, S'_\square \rightarrow S_\square \bullet, n.$$

Again, there are three kinds of consequence rules, for scanning, prediction and completion.

Scanning Scanning does not change the index stacks, so we get:

$$\frac{i, A_\eta \rightarrow \alpha \bullet w_j \beta, j}{i, A_\eta \rightarrow \alpha w_j \bullet \beta, j + 1}$$

Prediction In the case of Earley-style parsing with indexed grammars prediction splits into two rules. The first variant on the prediction rule has as side condition a grammar production with lhs in N (the lhs is a nonterminal without index). This rule leaves the index stack unaffected.

$$\frac{i, A_\eta \rightarrow \alpha \bullet B_\zeta \beta, j}{j, B_\zeta \rightarrow \bullet \gamma_\zeta, j} B \rightarrow \gamma$$

The second variant of the prediction rule pops the index stack. This variant has as side condition a grammar production with lhs in N° (the lhs is a nonterminal with as its index the nonterminal at the top of the index stack).

$$\frac{i, A \rightarrow \alpha \bullet B_{X:\zeta} \beta, j}{j, B_{X:\zeta} \rightarrow \bullet \gamma_\zeta, j} B_X \rightarrow \gamma$$

Completion For the case of Earley-style parsing with indexed grammars, this takes the following shape:

$$\frac{i, A_\eta \rightarrow \alpha \bullet B_\zeta \beta, k \quad k, B_\zeta \rightarrow \gamma \bullet, j}{i, A_\eta \rightarrow \alpha B_\zeta \bullet \beta, j}$$

Simple-minded parsing with sequentially indexed grammars A simple-minded way of adapting the above algorithm to the case of parsing with sequentially indexed grammars is as follows.

Nothing changes, except for the definition of the prediction rules:

$$\frac{i, A_\eta \rightarrow \alpha \bullet B_\zeta \beta, j}{j, B_\zeta \rightarrow \bullet \delta, j} B \rightarrow \gamma$$

where $\delta \in (\gamma)_\zeta$, with $(\gamma)_\zeta$ as in Definition 5.

$$\frac{i, A_\eta \rightarrow \alpha \bullet B_{X:\zeta} \beta, j}{j, B_{X:\zeta} \rightarrow \bullet \delta, j} B_X \rightarrow \gamma$$

again with $\delta \in (\gamma)_\zeta$.

This approach works, but it has the disadvantage that it yields an algorithm that is, like the algorithm for Earley parsing with indexed grammars, exponential in the size of the list of input tokens.

Even if we limit the size of the index stacks to the length of the remaining input, a grammar with k index symbols has k^j index stacks of length j . The number of relevant items to consider if the input has length n is $c \cdot k^n$.

8 Some Results on Stack Bound

To develop a polynomial algorithm for parsing with sequentially indexed grammars we need a bound on stack growth. The following definition is useful to formulate the appropriate theorem. In what follows we assume throughout that $G = (N, T, P, S)$ is a sequentially indexed grammar.

Definition 7 *Let α be a sentential form for G , i.e., let α be a string in $(N^\bullet \cup T)^*$. Then $\#\alpha$ equals the number of non-indexed symbols in α (either terminals or nonterminals with an empty stack).*

Examples: $\#(aAb) = 3$, $\#(aAB_Y) = 2$, $\#(aA_{XX}B_Y) = 1$.

The following theorem uses this definition to impose a bound on stack lengths inside derivations.

Lemma 8 *Let $A_\zeta \Rightarrow^* \alpha_1 C_{\theta+\zeta} \alpha_2$. Then $\#\alpha_1 + \#\alpha_2 \geq |\theta|$.*

Proof. Induction on n . If $n = 0$ then the property holds.

Suppose the property holds for n . Assume $A_\zeta \Rightarrow^* \alpha_1 C_{\theta+\zeta} \alpha_2$ with $|\theta| = n + 1$. Then $\theta = X : \rho$ and there has to be a grammar rule $B \rightarrow \gamma_1 D_X \gamma_2$ and a derivation $D_{X:\rho+\zeta} \Rightarrow^* \delta_1 C_{X:\rho+\zeta} \delta_2$ (during which the stack $X : \rho+\zeta$ gets transferred from D to C , possibly with other nonterminals in between) with $\alpha_1 = \beta_1 \gamma_1 \delta_1$ and $\alpha_2 = \delta_2 \gamma_2 \beta_2$, and

$$A_\zeta \Rightarrow^* \beta_1 B_{\rho+\zeta} \beta_2 \Rightarrow \beta_1 \gamma_1 D_{X:\rho+\zeta} \gamma_2 \beta_2.$$

Applying the induction hypothesis to $A_\zeta \Rightarrow^* \beta_1 B_{\rho+\zeta} \beta_2$ we find that $\#\beta_1 + \#\beta_2 \geq |\rho|$. From the restriction on the rule format we get that $\#\gamma_1 + \#\gamma_2 \geq 1$. Together with $\alpha_1 = \beta_1 \gamma_1 \delta_1$, $\alpha_2 = \delta_2 \gamma_2 \beta_2$ this gives that $\#\alpha_1 + \#\alpha_2 \geq n + 1$. \square

Lemma 9 *Let $\alpha \Rightarrow^* \beta$ be a derivation with G such that $\beta \in T^*$. Then $\#\alpha \leq |\beta|$.*

Proof. Immediate from the fact that nonterminals without index that appear in the righthand side of a grammar rule do not derive the empty string. \square

Theorem 10 *Let $A_\zeta \Rightarrow^* \alpha_1 C_{\theta+\zeta} \alpha_2$ and $\alpha_1 \Rightarrow^* \beta_1$ and $\alpha_2 \Rightarrow^* \beta_2$ be derivations for G with $\beta_1, \beta_2 \in T^*$. Then $|\theta| \leq |\beta_1 \beta_2|$.*

Proof. Combine the result from Lemma 8 that $|\theta| \leq \#\alpha_1 + \#\alpha_2$ with the result from Lemma 9 that $\#\alpha_1 \leq |\beta_1|$ and $\#\alpha_2 \leq |\beta_2|$. \square

Theorem 11 *Let $S \Rightarrow^* \alpha_1 A_\theta \alpha_2 \Rightarrow^* \beta$ be a derivation for G with $\beta \in T^*$. Then $|\theta| \leq |\beta|$.*

Proof. If $S \Rightarrow^* \alpha_1 A_\theta \alpha_2 \Rightarrow^* \beta$ then there are $\beta_1, \beta_2, \beta_3$ with $\beta = \beta_1 \beta_2 \beta_3$ and $\alpha_1 \Rightarrow^* \beta_1$ and $A_\theta \Rightarrow^* \beta_2$ and $\alpha_2 \Rightarrow^* \beta_3$. Applying Theorem 10 we see that $|\theta| \leq |\beta_1 \beta_3|$. It follows from this and $\beta = \beta_1 \beta_2 \beta_3$ that $|\theta| \leq |\beta|$. \square

9 Polynomial Parsing with Sequentially Indexed Grammars

Earley items For polynomial parsing of sequentially indexed languages, Earley items will be adapted by using three extra components:

1. A stack of the indices at the point where the rule was invoked,
2. A stack of indices for the first nonterminal to the right of \bullet ,
3. A stack of indices for the tail of the nonterminal list to the right of \bullet .

We will use Greek letters η, ζ, θ for index stacks,

The item format now becomes:

$$i, \theta, A \rightarrow \alpha \bullet \beta, \eta, \zeta, j$$

where θ, η, ζ are stacks of indices (nonterminals). The item indicates the following:

- grammar rule $A \rightarrow \alpha\beta$ was invoked at point i ,
- at the point of invocation, the top node A has associated stack θ ,
- at point j , part α of the righthand side of the rule has been successfully recognized,
- η is the stack for the first nonterminal among β (if β has no nonterminals, then η is empty),
- ζ is the stack for the remainder of the nonterminals in β (if β has less than two nonterminals, then ζ is empty).

Axiom In the case of Earley parsing with CF grammars, there is one axiom. It has the form $0, S' \rightarrow \bullet S, 0$, where S is the start symbol of the grammar and S' is a new start symbol. Adapting this to the case of parsing with sequentially indexed grammars, the axiom takes the shape

$$0, [], S' \rightarrow \bullet S, [], [], 0,$$

indicating that at the beginning of the parse, there is one pending nonterminal, and all stack components are empty.

Goal In the case of Earley parsing with CF grammars, there is one goal. It has the form $0, S' \rightarrow S\bullet, n$, where S is the start symbol of the grammar, S' is the new start symbol used in the axiom, and n is the length of the input. For the case of Earley style parsing with indexed grammars, we also require that the index stack components are empty at the end of the parse, so the goal shape becomes:

$$0, [], S' \rightarrow S\bullet, [], [], n.$$

Consequences As in the case of Earley parsing with CF grammars, there are three kinds of consequences, for scanning, prediction and completion.

Scanning The scanning rule for Earley parsing with CF grammars is the rule that shifts the bullet across a terminal. For parsing sequentially indexed languages, three index stack components are added to this. Scanning does not change the index stacks θ, η, ζ .

$$\frac{i, \theta, A \rightarrow \alpha \bullet w\beta, \eta, \zeta, j}{i, \theta, A \rightarrow \alpha w \bullet \beta, \eta, \zeta, j + 1}$$

Prediction The prediction rule for Earley parsing is the rule that initializes a new rule $B \rightarrow \gamma$ on the basis of a premise indicating that B is expected at the current point in the input. In the case of Earley-style parsing with sequentially indexed grammars this splits into four rules. The rules split the first index stack. For this we need some terminology.

If γ is a list of grammar symbols and η, η', η'' are index stacks, then $c(\gamma)$ is the number of nonterminals in γ , and $C(\eta, \eta', \eta'', \gamma)$ is the following constraint:

$$\eta = \eta' ++ \eta'' \wedge (c(\gamma) = 0 \Rightarrow \eta = []) \wedge (c(\gamma) = 1 \Rightarrow \eta'' = []).$$

The first prediction rule covers the case of an expected nonterminal B matched against a rule with head B . The rule distributes the appropriate stack over the new item, in accordance with the constraint imposed by the number of nonterminals in the righthand side of the grammar rule used in the prediction.

$$\frac{i, \theta, A \rightarrow \alpha \bullet B\beta, \eta, \zeta, j}{j, \eta, B \rightarrow \bullet \gamma, \eta', \eta'', j} B \rightarrow \gamma, C(\eta, \eta', \eta'', \gamma)$$

The second rule covers the case of an expected nonterminal B matched against a rule with head B_X . This rule pops the index stack associated with B .

$$\frac{i, \theta, A \rightarrow \alpha \bullet B\beta, (X : \eta), \zeta, j}{j, \eta, B_X \rightarrow \bullet \gamma, \eta', \eta'', j} B_X \rightarrow \gamma, C(\eta, \eta', \eta'', \gamma)$$

The third rule covers the case of an expected nonterminal B_Y matched against a rule $B \rightarrow \gamma$:

$$\frac{i, \theta, A \rightarrow \alpha \bullet B_Y\beta, \eta, \zeta, j}{j, (Y : \eta), B \rightarrow \bullet \gamma, \eta', \eta'', j} B \rightarrow \gamma, C(Y : \eta, \eta', \eta'', \gamma), |\eta| < n$$

Note the side condition on the rule. The side condition prevents unlimited growth of the stack. This is needed to prevent a rule like $A \rightarrow A_Y a$ from causing an unbounded number of pushes. It follows from Theorem 11 that imposing this side condition does not compromise completeness.

The fourth rule covers the case of an expected nonterminal B_Y matched against a rule $B_Y \rightarrow \gamma$:

$$\frac{i, \theta, A \rightarrow \alpha \bullet B_Y \beta, \eta, \zeta, j}{j, \eta, B_Y \rightarrow \bullet \gamma, \eta', \eta'', j} B_Y \rightarrow \gamma, C(\eta, \eta', \eta'', \gamma)$$

Completion The completion rule for Earley parsing is the rule that shifts the bullet across a non-terminal. For the case of Earley-style parsing with sequentially indexed grammars, this splits into four rules, as follows.

The first rule checks that the lefthand tail index stack of the first premise matches the head index stack of the second premise, for the case of a match of expected symbol B against completed rule $B \rightarrow \gamma$.

$$\frac{i, \theta, A \rightarrow \alpha \bullet B \beta, \eta, \zeta, k \quad k, \eta, B \rightarrow \gamma \bullet, \square, \square, j}{i, \theta, A \rightarrow \alpha B \bullet \beta, \zeta', \zeta'', j} C(\zeta, \zeta', \zeta'', \beta)$$

The second rule covers the case of a match of expected symbol B against completed rule $B_Y \rightarrow \gamma$.

$$\frac{i, \theta, A \rightarrow \alpha \bullet B \beta, (Y : \eta), \zeta, k \quad k, \eta, B_Y \rightarrow \gamma \bullet, \square, \square, j}{i, \theta, A \rightarrow \alpha B \bullet \beta, \zeta', \zeta'', j} C(\zeta, \zeta', \zeta'', \beta)$$

The third rule covers the case of a match of expected symbol B_Y against completed rule $B \rightarrow \gamma$.

$$\frac{i, \theta, A \rightarrow \alpha \bullet B_Y \beta, \eta, \zeta, k \quad k, (Y : \eta), B \rightarrow \gamma \bullet, \square, \square, j}{i, \theta, A \rightarrow \alpha B_Y \bullet \beta, \zeta', \zeta'', j} C(\zeta, \zeta', \zeta'', \beta)$$

The fourth rule covers the case of a match of expected symbol B_Y against completed rule $B_Y \rightarrow \gamma$.

$$\frac{i, \theta, A \rightarrow \alpha \bullet B_Y \beta, \eta, \zeta, k \quad k, \eta, B_Y \rightarrow \gamma \bullet, \square, \square, j}{i, \theta, A \rightarrow \alpha B_Y \bullet \beta, \zeta', \zeta'', j} C(\zeta, \zeta', \zeta'', \beta)$$

This completes the description of the inference rules. In terms of these, we define item derivability.

Definition 12 (Derivability) *Let G be a sequentially indexed grammar, and w_0, \dots, w_{n-1} a list of terminal symbols. Then*

$$\vdash_{w_0, \dots, w_{n-1}}^G i, \theta, A \rightarrow \alpha \bullet \beta, \eta, \zeta, j$$

expresses that item $i, \theta, A \rightarrow \alpha \bullet \beta, \eta, \zeta, j$ is derivable in the parsing calculus from the axiom by means of the inference rules.

10 Soundness and Completeness of the Parsing Calculus

Soundness We have to show that the parsing calculus is sound: if

$$\vdash_{w_0, \dots, w_{n-1}}^G 0, [], S' \rightarrow S \bullet, [], [], n,$$

then

$$S \xRightarrow{G^*} w_0, \dots, w_{n-1}.$$

For this, we will use Correctness Lemma 14, but first we introduce notation for the right adjoint of $\#$, as follows:

Definition 13 *If $\eta, \zeta \in N^*$ and ζ is a suffix of η , then η/ζ is the stack θ with $\eta = \theta \# \zeta$.*

This definition is used in the formulation of Lemma 14.

Lemma 14 (Correctness) *Let an indexed grammar G and an input sequence w_0, \dots, w_{n-1} be given. Then*

$$\vdash_{w_0, \dots, w_{n-1}}^G i, \theta, A \rightarrow \alpha \bullet \beta, \eta, \zeta, j$$

implies that the following conditions are fulfilled:

1. $A \rightarrow \alpha\beta$ is a production of G ,
2. there are $\gamma, \delta \in (N^\bullet \cup T)^*$ with

$$S \xRightarrow{G^*} \gamma A \theta \delta \text{ and } \gamma \xRightarrow{G^*} w_0, \dots, w_{i-1},$$

3. θ has $\eta \# \zeta$ as a suffix,
4. there is an $\alpha' \in (N^\bullet \cup T)^*$ with $\alpha' \in (\alpha)_{\theta/(\eta \# \zeta)}$ and

$$\alpha' \xRightarrow{G^*} w_i, \dots, w_{j-1}.$$

Proof. Induction on the number of items which were added to the item database before item $i, \theta, A \rightarrow \alpha \bullet \beta, \eta, \zeta, j$ got added.

For the basis, observe that the first items that got in were triggered by the axiom $0, [], S' \rightarrow \bullet S, [], 0$, so they must have been the result of a predict action for S . In other words, these items are of the form $0, [], S \rightarrow \bullet \gamma, [], [], 0$ or $0, [], S_X \rightarrow \bullet \gamma, [], [], 0$. It is clear that in both cases γ, δ, α' can be found with the required properties (namely, $\gamma = \delta = \alpha' = \epsilon$).

Induction step. Suppose that the induction hypothesis holds for all items presently in the store.

Consider a new item that got added as the result of a scan. Then this new item has the form $i, \theta, A \rightarrow \alpha w \bullet \beta, \eta, \zeta, j + 1$, and it derives from $i, \theta, A \rightarrow \alpha \bullet w \beta, \eta, \zeta, j$. The result follows from the fact that the latter item satisfies the induction hypothesis.

Consider a new item that got added as the result of a prediction. Then there are four cases to check. As an example, we check the first case. The new item has the form

$$j, \eta, B \rightarrow \bullet \chi, \eta', \eta'', j,$$

and it derives from an item

$$i, \theta, A \rightarrow \alpha \bullet B \beta, \eta, \zeta, j.$$

The latter item satisfies the induction hypothesis, so there are γ, δ with

$$S \xrightarrow{G^*} \gamma A \theta \delta \text{ and } \gamma \xrightarrow{G^*} w_0, \dots, w_{i-1},$$

and there is an α' with $\alpha' \in (\alpha)_{\theta/(\eta+\zeta)}$, $\alpha' \xrightarrow{G^*} w_i, \dots, w_{j-1}$.

It follows that $S \xrightarrow{G^*} \gamma \alpha' B \eta \beta' \delta$, for some $\beta' \in (\beta)_{\zeta}$, and $\epsilon_{\square} \xrightarrow{G^*} \epsilon$, so the property holds.

Consider a new item that got added as the result of a completion step. Then again there are four cases to check. As an example, we check the first case. The new item has the form $i, \theta, A \rightarrow \alpha B \bullet \beta, \zeta', \zeta'', j$, and it derives from premises:

$$i, \theta, A \rightarrow \alpha \bullet B \beta, \eta, \zeta, k \quad k, \eta, B \rightarrow \chi \bullet, \square, \square, j.$$

Invoking the induction hypothesis on the first premise we get γ, δ with

$$S \xrightarrow{G^*} \gamma A \theta \delta \text{ and } \gamma \xrightarrow{G^*} w_0, \dots, w_{i-1},$$

and $\alpha' \in (\alpha)_{\theta/(\eta+\zeta)}$ with $\alpha' \xrightarrow{G^*} w_i, \dots, w_{k-1}$.

Invoking the induction hypothesis on the second premise we get that for some $\chi' \in (\chi)_{\eta}$, $\chi' \xrightarrow{G^*} w_k, \dots, w_{j-1}$.

Together with $B \rightarrow \chi$ this gives $B \eta \xrightarrow{G^*} w_k, \dots, w_{j-1}$, and together with $\alpha' \xrightarrow{G^*} w_i, \dots, w_{k-1}$ we get:

$$\alpha' B \eta \xrightarrow{G^*} w_i, \dots, w_{j-1}.$$

It follows that the required property holds. □

Theorem 15 (Soundness) *The parsing calculus is sound.*

Proof. Apply Lemma 14 to the special case of item $0, \square, S' \rightarrow S \bullet, \square, \square, n$. □

Completeness We have to show: if $S \xrightarrow{G^*} w_0, \dots, w_{n-1}$ then the parsing calculus for G on input w_0, \dots, w_{n-1} allows us to prove this fact, i.e., the calculus yields:

$$\vdash_{w_0, \dots, w_{n-1}}^G 0, \square, S' \rightarrow S \bullet, \square, \square, n.$$

The completeness result follows from Adequacy Lemma 16.

Lemma 16 (Adequacy) *Let an indexed grammar G and an input sequence w_0, \dots, w_{n-1} be given. Suppose that the following conditions are fulfilled:*

1. $A \rightarrow \alpha\beta$ is a production of G ,
2. there are $\gamma, \delta \in (N^\bullet \cup T)^*$ with

$$S \xRightarrow{G^*} \gamma A \theta \delta \text{ and } \gamma \xRightarrow{G^*} w_0, \dots, w_{i-1},$$

3. θ has $\eta ++ \zeta$ as a suffix,
4. there is an $\alpha' \in (N^\bullet \cup T)^*$ with $\alpha' \in (\alpha)_{\theta/(\eta ++ \zeta)}$ and

$$\alpha' \xRightarrow{G^*} w_i, \dots, w_{j-1}$$

Then it follows that:

$$\vdash_{w_0, \dots, w_{n-1}}^G i, \theta, A \rightarrow \alpha \bullet \beta, \eta, \zeta, j$$

Proof. The proof is a variation on the proof of the Adequacy Lemma for Earley parsing with context free grammars (see, e.g., [AU72]).

An instance of the implication that we have to show can be characterized by a list

$$[i, \theta, A, \alpha, \beta, \eta, \zeta, j, \alpha', \gamma, \delta].$$

Let \mathcal{I} range over such instances. The rank $r(\mathcal{I})$ of an instance \mathcal{I} is given by

$$r(\mathcal{I}) = \tau_1(\mathcal{I}) + 2(j + \tau_2(\mathcal{I}) + \tau_3(\mathcal{I}))$$

where

$$\begin{aligned} \tau_1(\mathcal{I}) &= \text{the length of the shortest derivation } S \xRightarrow{G^*} \gamma A \theta \delta \\ \tau_2(\mathcal{I}) &= \text{the length of the shortest derivation } \gamma \xRightarrow{G^*} w_0, \dots, w_{i-1} \\ \tau_3(\mathcal{I}) &= \text{the length of the shortest derivation } \alpha' \xRightarrow{G^*} w_i, \dots, w_{j-1}. \end{aligned}$$

We prove the implication by induction on instance rank.

Base case. If the rank of an instance \mathcal{I} is 0, then it is easily seen that

$$\tau_1(\mathcal{I}) = j = \tau_2(\mathcal{I}) = \tau_3(\mathcal{I}) = 0.$$

Therefore,

$$\alpha = \alpha' = \gamma = \delta = \epsilon \text{ and } \theta = \eta = \zeta = [].$$

Thus A must be equal to the start symbol S , and what we need to show is that presence of rule $S \rightarrow \beta$ in the grammar implies that item

$$0, [], S \rightarrow \bullet \beta, [], [], 0$$

is in the store. This follows from the fact that this item is predicted from the axiom $0, [], S' \rightarrow \bullet S, [], [], 0$.

Induction step. Consider instance

$$\mathcal{I} = [i, \theta, A, \alpha, \beta, \eta, \zeta, j, \alpha', \gamma, \delta],$$

and assume that the implication holds for all instances with rank $< r(\mathcal{I})$. Assume that the antecedent of the implication holds for the instance. There are three cases to consider, depending on the form of α .

Case 1. Suppose \mathcal{I} has $\alpha = \chi w$ for some terminal symbol w . Since it is given that $\alpha' \in (\alpha)_{\theta/(\eta+\zeta)}$ and

$$\alpha' \xrightarrow{G^*} w_i, \dots, w_{j-1}$$

we can conclude that $w = w_{j-1}$ and that $\alpha' = \chi' w$ with $\chi' \in (\chi)_{\theta/(\eta+\zeta)}$, and therefore:

$$\chi' \xrightarrow{G^*} w_i, \dots, w_{j-2}.$$

Consider instance $[i, \theta, A, \chi, w\beta, \eta, \zeta, j-1, \alpha', \gamma, \delta]$. This instance has rank $< r(\mathcal{I})$, and it satisfies the antecedent of the induction hypothesis implication. So by induction hypothesis we get that item $i, \theta, A \rightarrow \chi \bullet w\beta, \eta, \zeta, j-1$ is in the store. Then by an application of the scan rule, item $i, \theta, A \rightarrow \alpha \bullet \beta, \eta, \zeta, j$ is also in the store.

Case 2. Suppose \mathcal{I} has $\alpha = \chi B$ for some nonterminal symbol B . From $\alpha' \in (\alpha)_{\theta/(\eta+\zeta)}$ and $\alpha = \chi B$ we can conclude that α' has the form $\chi' B_\rho$, with $\chi' \in (\chi)_{\theta/(\rho+\eta+\zeta)}$.

Then it follows from $\alpha' \xrightarrow{G^*} w_i, \dots, w_{j-1}$ that there is some k with $i \leq k \leq j$ with $\chi' \xrightarrow{G^*} w_i, \dots, w_{k-1}$ and $B_\rho \xrightarrow{G^*} w_k, \dots, w_{j-1}$. From instance

$$[i, \theta, A, \chi, B\beta, \rho, \eta + \zeta, j, \alpha', \gamma, \delta],$$

with rank $< r(\mathcal{I})$, we conclude that item

$$i, \theta, A \rightarrow \chi \bullet B\beta, \rho, \eta + \zeta, j$$

is in store. Let $B \rightarrow \mu$ be the grammar rule that is used in the first step $B \xrightarrow{G} \mu'$ (with $\mu' \in (\mu)_\rho$) of a minimum length derivation $B_\rho \xrightarrow{G^*} w_k, \dots, w_{j-1}$. Now for each $\beta' \in (\beta)_{\eta+\zeta}$, consider the instance

$$[k, \rho, B, \mu, \epsilon, [], [], j, \mu', \gamma\chi', \beta'\delta],$$

Let \mathcal{I}' be such an instance. We show that $r(\mathcal{I}') < r(\mathcal{I})$.

We have $S \xrightarrow{G^*} \gamma A \theta \delta \xrightarrow{G} \gamma \chi' B_\rho \beta' \delta$, so $\tau_1(\mathcal{I}') \leq \tau_1(\mathcal{I}) + 1$.

Let n_1 be the minimum number of steps in a derivation $\chi' \xrightarrow{G^*} w_i, \dots, w_{k-1}$, and n_2 the minimum number of steps in a derivation $B_\rho \xrightarrow{G^*} w_k, \dots, w_{j-1}$. Then $\tau_3(\mathcal{I}) = n_1 + n_2$. Since $B_\rho \xrightarrow{G} \mu' \xrightarrow{G^*} w_k, \dots, w_{j-1}$, we see that $\tau_3(\mathcal{I}') = n_2 - 1$. Observe that $\tau_2(\mathcal{I}') = \tau_2(\mathcal{I}) + n_1$. It follows that

$$\tau_2(\mathcal{I}') + \tau_3(\mathcal{I}') = \tau_2(\mathcal{I}) + n_1 + n_2 - 1 = \tau_2(\mathcal{I}) + \tau_3(\mathcal{I}) - 1.$$

Thus,

$$\tau_1(\mathcal{I}') + 2(j + \tau_2(\mathcal{I}) + \tau_3(\mathcal{I})) < r(\mathcal{I}),$$

i.e., $r(\mathcal{I}') < r(\mathcal{I})$. By the induction hypothesis we conclude that item $k, \rho, B \rightarrow \mu \bullet, [], [], j$ is in the store. An application of the completion rule yields that

$$i, \theta, A \rightarrow \chi B \bullet \beta, \eta, \zeta, j$$

is in the store too.

Case 3. Suppose \mathcal{I} has $\alpha = \epsilon$. Then $i = j$, $\theta = \eta ++ \zeta$, and $\tau_3(\mathcal{I}) = 0$. From the given $r(\mathcal{I}) > 0$ it follows that any derivation $S \xrightarrow{\mathcal{G}^*} \gamma A_\theta \delta$ has positive length. For suppose the shortest derivation $S \xrightarrow{\mathcal{G}^*} \gamma A_\theta \delta$ has length 0. Then $\tau_1(\mathcal{I}) = 0$, $\gamma = \epsilon$ and $i = 0$, and therefore $\tau_2(\mathcal{I}) = 0$, and contradiction with the given that $r(\mathcal{I}) > 0$.

So there has to be some rule $B \rightarrow \mu A \nu$ used in the penultimate step of a shortest derivation $S \xrightarrow{\mathcal{G}^*} \gamma A_\theta \delta$, for some stack ρ , i.e.,

$$S \xrightarrow{\mathcal{G}^*} \gamma' B_\rho \delta' \xrightarrow{\mathcal{G}} \gamma' \mu' A_\theta \nu' \delta',$$

with $\gamma' \mu' = \gamma$, $\nu' \delta' = \delta$, and

$$\rho = \rho'' ++ \theta ++ \rho', \mu' \in (\mu)_{\rho''}, \nu' \in (\nu)_{\rho'},$$

and $\gamma' B_\rho \delta'$ is the penultimate step in some shortest derivation $S \xrightarrow{\mathcal{G}^*} \gamma A_\theta \delta$. Now consider the following instance \mathcal{I}' :

$$[k, \rho, B, \mu, A \nu, \theta, \rho', j, \mu', \gamma', \delta'],$$

for k an integer with

$$\gamma' \xrightarrow{\mathcal{G}^*} w_0, \dots, w_{k-1} \text{ and } \mu' \xrightarrow{\mathcal{G}^*} w_k, \dots, w_{j-1}.$$

Let the smallest length of a derivation $\gamma' \xrightarrow{\mathcal{G}^*} w_0, \dots, w_{k-1}$ be n_1 and let the smallest length of a derivation $\mu' \xrightarrow{\mathcal{G}^*} w_k, \dots, w_{j-1}$ be n_2 . Then:

$$\tau_2(\mathcal{I}') = n_1, \quad \tau_3(\mathcal{I}') = n_2, \text{ and } \tau_2(\mathcal{I}) = n_1 + n_2.$$

By the choice of B, μ', ν' , $\tau_1(\mathcal{I}') = \tau_1(\mathcal{I}) - 1$. From these facts, combined with the fact that $\tau_3(\mathcal{I}) = 0$, we get that

$$r(\mathcal{I}') = \tau_1(\mathcal{I}) - 1 + 2(j + \tau_2(\mathcal{I})) = r(\mathcal{I}) - 1.$$

Since item \mathcal{I}' satisfies the antecedent of the induction implication, item $k, \rho, B \rightarrow \mu \bullet A \nu, \theta, \rho', j$ is in the store. Since $\theta = \eta ++ \zeta$ and $A \rightarrow \beta$ is a rule of the grammar, we get by an application of the prediction rule that item $j, \theta, A \rightarrow \bullet \beta, \eta, \zeta, j$ is in the store. \square

Theorem 17 (Completeness) *The parsing calculus is complete.*

Proof. Apply Lemma 16 to the special case of instance $[0, [], S', S, \epsilon, [], [], n, S, \epsilon, \epsilon]$. \square

11 Complexity Analysis

Obviously, for CF grammars the algorithm behaves as Earley's algorithm, giving worst case time complexity $O(n^3)$ for arbitrary CF grammars, and complexity $O(n^2)$ for non-ambiguous CF grammars.

To analyse the cases that involve stack manipulations, consider the general form of an item:

$$(i, \theta, A \rightarrow \alpha \bullet \beta, \eta, \zeta, j).$$

A crucial insight for the complexity analysis of parsing with such items is that no concrete instances of the stacks θ, η, ζ need to be stored. Consider the example of a rule $A \rightarrow BC_X$ that gets applied with stack XYZ . Then the possible applications are $A_{XYZ} \Rightarrow B_{XYZ}C_X$, $A_{XYZ} \Rightarrow B_{XY}C_{XZ}$, $A_{XYZ} \Rightarrow B_XC_{XYZ}$, and $A_{XYZ} \Rightarrow BC_{XXYZ}$. The only information we need to store about the stack is its length 3, plus how this length gets split over the nonterminals in the righthand side: (3, 0) or (2, 1) or (1, 2) or (0, 3). Similarly for a case of an application of $A_X \rightarrow BC_Y$. The shape of the rule makes clear that the head of the stack at the point of invocation should be X , and the different ways of applying the rule correspond with all the ways of splitting the tail of the stack; e.g., the application $A_{XYZ} \Rightarrow B_{YZ}C_Y$ would correspond to the split (2, 0). Thus we see that constraints on the shape of items can be given in terms of $|\theta|$, $|\eta|$, $|\zeta|$ instead of θ, η, ζ , as follows:

- $i \leq j$: given j , there are $j + 1$ possible choices for i ,
- $|\theta| \leq n$,
- $\eta ++ \zeta$ is a suffix of θ : there are at most $|\theta| + 1 = n + 1$ such η, ζ .

The possible choices for A, α, β depend on the grammar. Assume for given G there are k such choices. Then it follows that there are

$$\sum_{j=0}^{n-1} k(j+1)n(n+1) = \frac{1}{2}kn^2(n+1)^2$$

different items to consider. For non-ambiguous grammars we can show that the parsing algorithm can be organized in such a way that it considers each item only once, and that the time it spends on the derivation step does not depend of the length on the input:

Lemma 18 *Let G be an unambiguous grammar and let w_0, \dots, w_{n-1} be a string of terminal symbols. Then the parsing deduction process for G and w_0, \dots, w_{j-1} can be organized in such a way that item*

$$(i, \theta, A \rightarrow \alpha \bullet \beta, \eta, \zeta, j)$$

is derived at most once if $\alpha \neq \epsilon$.

Proof omitted. It follows that the parsing time for such grammars is $O(n^4)$.

Now for the general case. Let S_j be the set of all items with index j as their last component. Then, as we have seen, S_j contains at most $(j + 1)n(n + 1)$ items. Now assume that we let the

completion steps be triggered by items of the form $\alpha \bullet$ (items with completed righthand sides). Then:

- items satisfying the pattern $\alpha \bullet B\beta$ or $\alpha \bullet B_X\beta$ will trigger a prediction rule. Prediction will add at most $n + 1$ new items (the number of ways to split a stack of length n).
- items satisfying the pattern $\alpha \bullet w\beta$ will trigger the scan rule, and scanning will add a single new item,
- items satisfying the pattern $\alpha \bullet$ will trigger a completion rule, and completion will add at most $(j + 1)(n + 1)$ items. This is because derived items have the form $(i, \theta, A \rightarrow \alpha B \bullet \beta, \zeta', \zeta'', j)$, with $j + 1$ ways to pick i below j , and $n + 1$ ways to split a stack ζ of length n .

Thus, the number of items to be considered in processing S_j is $(j + 1)^2 n(n + 1)^3$, and summing over j gives a polynomial for the worst case:

$$\sum_{j=0}^{n-1} (j + 1)^2 n(n + 1)^3 = \frac{1}{2} n^2 (n + 1)^5,$$

showing that the worst case time complexity of the algorithm for arbitrary grammars is $O(n^7)$.

12 Sequentially Indexed Languages: Where do They Fit?

Sequentially indexed languages are closed under union, concatenation, Kleene star, homomorphism, inverse homomorphism and intersection with regular languages. Thus, sequentially indexed languages are a so-called fully abstract family of languages [GG67].

Theorem 19 *Sequentially indexed grammars can generate all linear indexed languages in the sense of [Gaz88].*

Proof. Let a linear indexed grammar be given. Then there is a linear indexed grammar for the same language satisfying a set of constraints corresponding to the constraints on sequentially indexed grammar rules. First, put the LIG in this format. Next, consider linear indexed grammar rules of the form

$$A[\theta] \rightarrow A_1[] \cdots A_{i-1}[] A_i[X : \theta] A_{i+1}[] \cdots A_n[].$$

Add a new nonterminal \hat{A} for each nonterminal A . Then give the above grammar rule the following two sequential indexed grammar counterparts:

$$A \rightarrow A_1 \cdots A_{(i-1)} \hat{A}_{iX} A_{i+1} \cdots A_n,$$

$$\hat{A} \rightarrow A_1 \cdots A_{(i-1)} \hat{A}_{iX} A_{i+1} \cdots A_n.$$

Linear indexed grammar rules of the form

$$A[\theta] \rightarrow A_1[] \cdots A_{i-1} A_i[\theta] A_{i+1}[] \cdots A_n[]$$

should each get two counterparts:

$$A \rightarrow A_1 \cdots A_{i-1} \hat{A}_i A_{i+1} \cdots A_n,$$

$$\hat{A} \rightarrow A_1 \cdots A_{i-1} \hat{A}_i A_{i+1} \cdots A_n.$$

Linear indexed grammar rules of the form

$$A[X : \theta] \rightarrow A_1[] \cdots A_{i-1} A_i[\theta] A_{i+1}[] \cdots A_n[]$$

should each get a single counterpart:

$$\hat{A}_X \rightarrow A_1 \cdots A_{i-1} \hat{A}_i A_{i+1} \cdots A_n.$$

This ensures that only the stacks on nonterminals of the form \hat{A} ever get popped. \square

It follows from this that linear indexed languages are included in the class of sequentially indexed languages.

Theorem 20 *Linear indexed languages are a proper subclass of sequentially indexed languages.*

Proof. Inclusion follows from the previous theorem. To see that the inclusion is proper, note that

$$\{a^n b^m c^m d^k e^k f^n \mid n > 0, n = m + k\}$$

is a sequentially indexed language. It is generated by the grammar

$$\begin{aligned} S &\rightarrow aS_X f \mid BC \\ B_Y &\rightarrow bBc \mid bc \\ C_Y &\rightarrow dCe \mid de \end{aligned}$$

It follows from the pumping lemma for linear indexed languages (proved in [VS87] for TAG which are proved equivalent to LIG in [VSW94]; see also [Kan06]) that this language is not a linear indexed language. \square

One of the referees suggested that sequentially indexed grammars might be weakly equivalent to a variant of Combinatory Categorical Grammars in which composition into unbounded categories is allowed (see [VSWJ87]). I leave this issue for future research. However this may turn out, the sequential indexed grammar format has quite a lot going for it: the simplicity of its grammar rules, its closeness to the CFG format, its closeness to the IG format, the natural correspondence between indices and traces, and last but not least the perspicuity of the parsing scheme. Issues for future research are: (i) formulating and proving a pumping lemma for sequentially indexed languages, and (ii) finding out whether the restriction on the SIG rule format affects generating power of the grammar class, i.e., whether SIGs without the restrictions on the rule format imposed in Definition 4 generate the same language class. I conjecture that generating power is not affected but so far I have not been able to find a proof.

Acknowledgement This paper was written as a follow-up to a talk at the second workshop on Lambda Calculus, Type Theory and Natural Language (Sept 12, 2005), in London. I talked on the same topic at the Signes Seminar, Université Michel de Montaigne, Bordeaux, on March 14 of that year. I thank the audiences at these occasions for valuable feedback. The presentation owes a lot to the perceptive remarks of two anonymous referees of this Journal. They pointed out, among other things, that the parsing algorithm I propose makes crucial use of a restriction on stack length, and challenged me to give a formal justification of that restriction. This is now taken care of by Theorem 11. The referees also suggested that the relationship between LIG and SIG needed further investigation. This is now taken up in the final section. Christina Unger spotted some last-minute errors with her careful proofreading. Many thanks to all of you. All remaining errors are my own. The author is grateful to the Netherlands Institute for Advanced Studies in Wassenaar (NIAS) for providing the opportunity to complete this paper as Fellow-in-Residence.

References

- [Aho68] Alfred V. Aho. Indexed grammars — an extension of context-free grammars. *Journal of the ACM*, 15(4):647–671, 1968.
- [Aho69] Alfred V. Aho. Nested stack automata. *Journal of the ACM*, 16(3):383–406, 1969.
- [AU72] A.V. Aho and J.D. Ullman. *Theory of Parsing, Translation and Compiling*, volume Volume I: Parsing. Englewood Cliffs (NJ), 1972.
- [BKPZ82] J. Bresnan, R.M. Kaplan, S. Peters, and A. Zaenen. Cross-serial dependencies in Dutch. *Linguistic Inquiry*, 1982.
- [Ear70] J. Earley. An efficient context-free parsing algorithm. *Communications of the ACM*, 13:94–102, 1970.
- [Gaz88] G. Gazdar. Applicability of indexed grammars to natural languages. In U. Reyle and C. Rohrer, editors, *Natural Language Parsing and Linguistic Theories*, pages 69–94. Reidel, Dordrecht, 1988.
- [GG67] S. Ginsburg and S.A. Greibach. Abstract families of languages. In *Eight Annual Symposium on Switching and Automata Theory*, pages 128–139. IEEE, 1967.
- [Kan06] Makoto Kanazawa. The pumping lemma for tree-adjoining languages. Note for Mathematical Linguistics Course, May 2006.
- [SSP95] S.M. Shieber, Y. Schabes, and F.C.N. Pereira. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24:3–36, 1995.
- [VS87] K. Vijay-Shanker. *A Study of Tree Adjoining Grammars*. PhD thesis, University of Pennsylvania, 1987.
- [VSW94] K. Vijay-Shanker and D.J. Weir. The equivalence of four extensions of context-free grammars. *Mathematical Systems Theory*, 27:511–546, 1994.

- [VSWJ87] K. Vijay-Shanker, D.J. Weir, and A. Joshi. Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings of the 25th Meeting of the Association for Computational Linguistics*, pages 104–111, 1987.