A redefinition of Embedded Push-Down Automata*

Miguel A. Alonso†, Eric de la Clergerie‡ and Manuel Vilares†

†Departamento de Computación, Universidad de La Coruña
Campus de Elviña s/n, 15071 La Coruña (Spain)
{alonso,vilares}@dc.fi.udc.es
‡INRIA, Domaine de Voluceau
Rocquecourt, B.P. 105, 78153 Le Chesnay (France)
Eric.De_La_Clergerie@inria.fr

Abstract

A new definition of Embedded Push-Down Automata is provided. We prove this new definition preserves the equivalence with tree adjoining languages and we provide a tabulation framework to execute any automaton in polynomial time with respect to the length of the input string.

1. Introduction

Embedded Push-Down Automata (EPDA) were defined in (Vijay-Shanker, 1988) as an extension of Push-Down Automata that accept exactly the class of Tree Adjoining Languages. They can also be seen as a level-2 automata in a progression of linear iterated pushdowns involving nested stacks (Weir, 1994).

An EPDA consists of a finite state control, an input tape and a stack made up of non-empty stacks containing stack symbols. A transition can consult the state, the input string and the top element of the top stack and then change the state, read a character of the input string and replace the top element by a finite sequence of stack elements to give a new top stack, and new stacks can be placed above and below the top stack.

EPDA can describe parsing strategies for tree adjoining grammars in which adjunctions are recognized top-down. The same kind of strategies can be described in strongly-driven 2-stack automata (de la Clergerie & Alonso Pardo, 1998) and linear indexed automata (Nederhof, 1999), which has associated tabulation frameworks allowing those automata to be executed in polynomial time with respect to the size of the input string. In this paper we propose a redefinition of EPDA in order to provide a tabulation framework for this class of automata.

2. EPDA without states

Finite-state control is not a fundamental component of push-down automata, as the current state in a configuration can be stored in the top element of the stack of the automaton (Lang, 1991). Finite-state control can also be eliminated from EPDA, obtaining a new definition that considers a EPDA as a tuple $(V_T, V_S, \Theta, \$_0, \$_f)$ where V_T is a finite set of terminal symbols, V_S is a finite set of stack symbols, $\$_0 \in V_S$ is the initial stack symbol, $\$_f \in V_S$ is the final stack symbol and Θ is a finite set of six types of transition:

SWAP: Transitions of the form $C \stackrel{a}{\longmapsto} F$ that replace the top element of the top stack while scanning a. The application of such a transition on a stack $\Upsilon \,[\![\alpha B$ returns the stack $\Upsilon \,[\![\alpha C$.

* This research was partially supported by the FEDER of EU (Grant 1FD97-0047-C04-02) and Xunta de Galicia (Grant PGIDT99XI10502B).

PUSH: Transitions of the form $C \overset{a}{\longmapsto} C\ F$ that push F onto C. The application of such a transition on a stack $\Upsilon[\alpha C$ returns the stack $\Upsilon[\alpha CF$.

POP: Transitions of the form $C\ F \overset{a}{\longmapsto} G$ that replace C and F by G. The application of such a transition on $\Upsilon[\alpha CF$ returns the stack $\Upsilon[\alpha G$.

WRAP-A: Transitions *wrap-above* of the form $C \overset{a}{\longmapsto} C, [F$ that push a new stack $[F$ on the top of the automaton stack. The application of such a transition on a stack $\Upsilon[\alpha C$ returns the stack $\Upsilon[\alpha C[F$.

WRAP-B: Transitions *wrap-below* of the form $C \overset{a}{\longmapsto} [C, F$ that store a new stack $[C$ just below the top stack, and change from C to F the top element of the top stack. The application of such a transition on a stack $\Upsilon[\alpha C$ returns the stack $\Upsilon[C[\alpha F$.

UNWRAP: Transitions of the form $C, [F \overset{a}{\longmapsto} G$ that delete the top stack $[F$ and replace the new top element by G. The application of such a transition on a stack $\Upsilon[\alpha C[F$ returns the stack $\Upsilon[\alpha G$.

where $C, F, G \in V_S$, $\Upsilon \in ([V_S^*)^*$, $\alpha \in V_S^*$, $a \in V_T \cup \{\epsilon\}$ and $[\ \notin V_S$ is a new symbol used as stack separator. It can be proved that transitions of a EPDA with states can be emulated by transitions in Θ and vice versa.

An *instantaneous configuration* is a pair (Υ, w), where Υ represents the contents of the automaton stack and w is the part of the input string that is yet to be read. A configuration (Υ, aw) derives a configuration (Υ', w), denoted $(\Upsilon, aw) \vdash (\Upsilon', w)$, if and only if there exists a transition that applied to Υ gives Υ' and scans a from the input string. We use $\vdash^*$ to denote the reflexive and transitive closure of $\vdash$. An input string is accepted by an EPDA if $([\$_0, w) \vdash^* ([\$_f, \epsilon)$. The language accepted by an EPDA is the set of $w \in V_T^*$ such that $([\$_0, w) \vdash^* ([\$_f, \epsilon)$.

3. Compiling TAG into EPDA

We consider each elementary tree γ of a TAG as formed by a set of context-free productions $\mathcal{P}(\gamma)$: a node N^γ and its g children $N_1^\gamma \ldots N_g^\gamma$ are represented by a production $N^\gamma \rightarrow N_1^\gamma \ldots N_g^\gamma$. The elements of the productions are the nodes of the tree, except for the case of elements belonging to $V_T \cup \{\varepsilon\}$ in the right-hand side of production. Those elements may have no children and can not be adjoined, so we identify such nodes labeled by a terminal with that terminal. We use $\beta \in \mathrm{adj}(N^\gamma)$ to denote that a tree β may be adjoined at node N^γ. If adjunction is not mandatory at N^γ, then $\mathbf{nil} \in \mathrm{adj}(N^\gamma)$. We consider the additional productions $\top^\alpha \rightarrow \mathbf{R}^\alpha$, $\top^\beta \rightarrow \mathbf{R}^\beta$ and $\mathbf{F}^\beta \rightarrow \bot$ for each initial tree $\alpha \in \boldsymbol{I}$ and each auxiliary tree $\beta \in \boldsymbol{A}$, where $\mathbf{R}^\alpha$ is the root node of α and $\mathbf{R}^\beta$ and $\mathbf{F}^\beta$ are the root node and foot node of β, respectively. After disabling $\top^\gamma$ and $\bot$ as adjunction nodes the generative capability of the grammar remains intact.

Figure 1 shows the generic compilation schema from TAG to EPDA, where symbols $\nabla_{r,s}^\gamma$ have been introduced to denote dotted productions. The meaning of each compilation rule is graphically shown in figure 2. This schema is parameterized by $\overrightarrow{N^\gamma}$, the information propagated top-down w.r.t. the node N^γ, and by $\overleftarrow{N^\gamma}$, the information propagated bottom-up. When the schema is used to implement a top-down strategy $\overrightarrow{N^\gamma} = N^\gamma$ and $\overleftarrow{N^\gamma} = \Box$, where $\Box$ is a fresh stack symbol. A bottom-up strategy requires $\overrightarrow{N^\gamma} = \Box$ and $\overleftarrow{N^\gamma} = N^\gamma$. For a Earley-like parsing strategy, $\overrightarrow{N^\gamma} = \overline{N^\gamma}$ and $\overleftarrow{N^\gamma} = \overline{\overline{N^\gamma}}$, where $\overline{N^\gamma}$ and $\overline{\overline{N^\gamma}}$ are used to distinguish the top-down prediction from the bottom-up propagation of a node.

We can observe in figure 1 that each stack stores pending adjunctions with respect to the node placed on the top of the stack in a top-down treatment of adjunctions: when an adjunction node

[INIT]	$\$_0 \longmapsto \$_0 \, [\nabla^{\alpha}_{0,0}$	$\alpha \in \boldsymbol{I}$
[CALL]	$\nabla^{\gamma}_{r,s} \longmapsto \nabla^{\gamma}_{r,s} \, [\overrightarrow{N^{\gamma}_{r,s+1}}$	$N^{\gamma}_{r,s+1} \notin \mathrm{spine}(\gamma),\ \mathbf{nil} \in \mathrm{adj}(N^{\gamma}_{r,s+1})$
[SCALL]	$\nabla^{\beta}_{r,s} \longmapsto [\nabla^{\beta}_{r,s},\ \ \overrightarrow{N^{\beta}_{r,s+1}}$	$N^{\beta}_{r,s+1} \in \mathrm{spine}(\beta),\ \mathbf{nil} \in \mathrm{adj}(N^{\beta}_{r,s+1})$
[SEL]	$\overrightarrow{N^{\gamma}_{r,0}} \longmapsto \nabla^{\gamma}_{r,0}$	
[TAB]	$\nabla^{\gamma}_{r,n_r} \longmapsto \overleftarrow{N^{\gamma}_{r,0}}$	
[RET]	$\nabla^{\gamma}_{r,s},\ [\overleftarrow{N^{\gamma}_{r,s+1}} \longmapsto \nabla^{\gamma}_{r,s+1}$	$N^{\gamma}_{r,s+1} \notin \mathrm{spine}(\gamma),\ \mathbf{nil} \in \mathrm{adj}(N^{\gamma}_{r,s+1})$
[SRET]	$\nabla^{\beta}_{r,s},\ [\overleftarrow{N^{\beta}_{r,s+1}} \longmapsto \nabla^{\beta}_{r,s+1}$	$N^{\gamma}_{r,s+1} \in \mathrm{spine}(\beta),\ \mathbf{nil} \in \mathrm{adj}(N^{\beta}_{r,s+1})$
[SCAN]	$\overrightarrow{N^{\gamma}_{r,0}} \overset{a}{\longmapsto} \overleftarrow{N^{\gamma}_{r,0}}$	$N^{\gamma}_{r,0} \rightarrow a$
[ACALL-a]	$\nabla^{\gamma}_{r,s} \longmapsto [\nabla^{\gamma}_{r,s},\ \ \Delta^{\gamma}_{r,s}$	$\mathrm{adj}(N^{\gamma}_{r,s+1}) \neq \{\mathbf{nil}\}$
[ACALL-b]	$\Delta^{\gamma}_{r,s} \longmapsto \Delta^{\gamma}_{r,s}\ \ \overrightarrow{\top^{\beta}}$	$\beta \in \mathrm{adj}(N^{\gamma}_{r,s+1})$
[ARET]	$\nabla^{\gamma}_{r,s},\ [\overleftarrow{\top^{\beta}} \longmapsto \nabla^{\gamma}_{r,s+1}$	$\beta \in \mathrm{adj}(N^{\gamma}_{r,s+1})$
[FCALL-a]	$\nabla^{\beta}_{f,0} \longmapsto [\nabla^{\beta}_{f,0},\ \ \bot$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta}$
[FCALL-b]	$\Delta^{\gamma}_{r,s}\ \ \bot \longmapsto \overrightarrow{N^{\gamma}_{r,s+1}}$	
[FRET]	$\nabla^{\beta}_{f,0},\ [\overleftarrow{N^{\gamma}_{r,s+1}} \longmapsto \nabla^{\beta}_{f,1}$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \ \beta \in \mathrm{adj}(N^{\gamma}_{r,s+1})$
[FINAL]	$\$_0 \, [\nabla^{\alpha}_{0,1} \longmapsto [\$_f$	$\alpha \in \boldsymbol{I}$

Figure 1: Generic compilation schema from TAG to EPDA

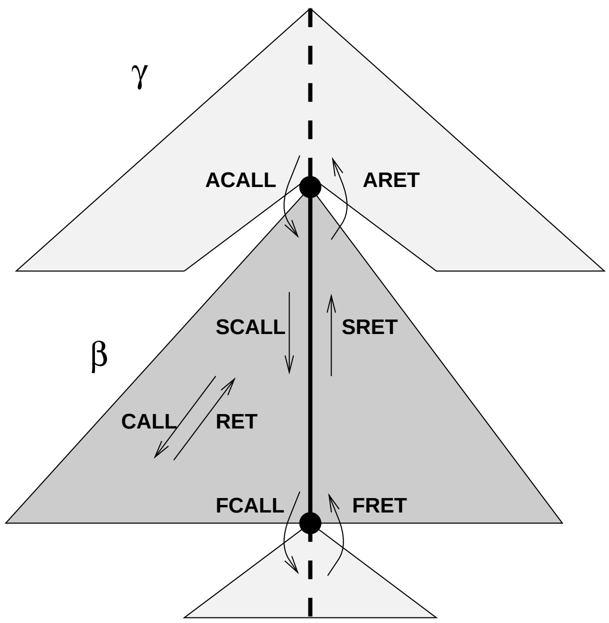

Figure 2: Meaning of compilation rules

Transition	**EPDA**	**L-LIA**
SWAP	$C \overset{a}{\longmapsto} F$	$C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ]$
PUSH	$C \overset{a}{\longmapsto} CF$	$C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ C]$
POP	$CF \overset{a}{\longmapsto} G$	$F[\circ\circ C] \overset{a}{\longmapsto} G[\circ\circ]$
WRAP-A	$C \overset{a}{\longmapsto} C, \llbracket F$	$C[\circ\circ] \overset{a}{\longmapsto} C[\circ\circ]\ F[\,]$
WRAP-B	$C \overset{a}{\longmapsto} \llbracket C, F$	$C[\circ\circ] \overset{a}{\longmapsto} C[\,]\ F[\circ\circ]$
UNWRAP	$C, \llbracket F \overset{a}{\longmapsto} G$	$C[\circ\circ]\ F[\,] \overset{a}{\longmapsto} G[\circ\circ]$
WRAP-B+PUSH	$C \overset{a}{\longmapsto} \llbracket C, X\,F$	$C[\circ\circ] \overset{a}{\longmapsto} C[\,]\ F[\circ\circ X]$
WRAP-B+POP	$X\,C \overset{a}{\longmapsto} \llbracket C, F$	$C[\circ\circ X] \overset{a}{\longmapsto} C[\,]\ F[\circ\circ]$

Figure 3: Equivalence between EPDA and L-LIA

is reached, the adjunction node is stored on the top of the stack (**[ACALL-a]**) and the traversal of the auxiliary tree is started (**[ACALL-b]**); the adjunction stack is propagated through the spine (**[SCALL]**) down to the foot node, where the traversal of the auxiliary tree is suspended to resume the traversal of the subtree rooted by the adjunction node (**[FCALL-a]**), which is eliminated of the stack (**[FCALL-b]**). To avoid confusion, we store $\Delta^{\gamma}_{r,s}$ instead of $\nabla^{\gamma}_{r,s}$ to indicate that an adjunction was started at node $N^{\gamma}_{r,s+1}$. A symbol Δ can be seen as a symbol ∇ waiting an adjunction to be completed.

4. EPDA and Left-oriented Linear Indexed Automata

Left-oriented Linear Indexed Automata (L-LIA) is a class of automata defined by Nederhof (1999) that can be used to implement parsing strategies for TAG in which adjunctions are recognized in a top-down way. Given a EPDA, the equivalent L-LIA is obtained by means of a simple change in the notations: if we consider the top element of a stack as a stack symbol, and the rest of the stack as the indices list associated to them, we obtain the correspondence shown in figure 3.

This change in notation is also useful to show that EPDA accept exactly the class of tree adjoining languages. That tree adjoining languages are accepted by EPDA is shown by the compilation schema defined previously. To prove that the languages accepted by EPDA are tree adjoined languages, we exhibit a procedure that, given an EPDA $\mathcal{A} = (V_T, V_S, \Theta, \$_0, \$_f)$, builds a linear indexed grammar (Gazdar, 1987) $\mathcal{G} = (V_T, V_N, V_I, S, P)$ that recognizes the language accepted by $\mathcal{A}$. Non-terminals in V_N are pairs $\langle A, B\rangle$, where $A, B \in V_S$, and $V_I = V_S$. Productions in P are obtained from transitions in Θ as follows:

- For each transition $C \overset{a}{\longmapsto} F$ and for each $E \in V_S$, a production $\langle C, E\rangle[\circ\circ] \to a\ \langle F, E\rangle[\circ\circ]$ is created.
- For each transition $C \overset{a}{\longmapsto} CF$ and for each $E \in V_S$, a production $\langle C, E\rangle[\circ\circ] \to a\ \langle F, E\rangle[\circ\circ C]$ is created.
- For each transition $C\ F \overset{a}{\longmapsto} G$ and for each $E \in V_S$, a production $\langle F, E\rangle[\circ\circ C] \to a\ \langle G, E\rangle[\circ\circ]$ is created.

- For each pair of transitions $C \overset{b}{\longmapsto} C, \lceil F'$ and $C, \lceil F \overset{a}{\longmapsto} G$, and for each $E \in V_S$, a production $\langle C, E\rangle[\circ\circ] \to b\ \langle F', F\rangle[\,]\ a\ \langle G, E\rangle[\circ\circ]$ is created.

- For each pair of transitions $C \overset{b}{\longmapsto} \lceil C, F'$ and $C, \lceil F \overset{a}{\longmapsto} G$, and for each $E \in V_S$, a production $\langle C, E\rangle[\circ\circ] \to b\ \langle F', F\rangle[\circ\circ]\ a\ \langle G, E\rangle[\,]$ is created.

- For each $E \in V_S$, a production $\langle E, E\rangle[\,] \to \epsilon$ is created.

The axiom of the grammar is $S = \langle \$_0, \$_f\rangle$. Applying induction in the length of derivations, we can prove that $\langle C, E\rangle[\alpha] \overset{*}{\Rightarrow} w$ if and only if $(\lceil \alpha C, w) \vdash {}^*(\lceil E, \epsilon)$.

5. Tabulation

The direct execution of EPDA may be exponential with respect to the length of the input string and may even loop. To get polynomial complexity, we must avoid duplicating computations by tabulating traces of configurations called *items*. The amount of information to keep in an item is the crucial point to determine to get efficient executions.
The tabulation of EPDA using PUSH and POP transitions without restrictions seems to be difficult. By studying the compilation schema of figure 1, we observe that the compilation rules **[ACALL-a]** and **[ACALL-b]** can be combined to form a single rule **[ACALL]** generating transitions WRAP-B+PUSH of the form $C \longmapsto \lceil C, X\ F$:

$$\textbf{[ACALL]} \quad \nabla^{\gamma}_{r,s} \longmapsto \lceil \nabla^{\gamma}_{r,s},\ \ \Delta^{\gamma}_{r,s}\ \ \top^{\beta}$$

such that $\beta \in \text{adj}(N^{\gamma}_{r,s+1})$. The **[FCALL-a]** and **[FCALL-b]** can be combined to form a single rule generating transitions WRAP-B+POP of the form $X\ C \longmapsto \lceil C, F$:

$$\textbf{[FCALL]} \quad \Delta^{\gamma}_{r,s}\ \ \nabla^{\beta}_{f,0} \longmapsto \lceil \nabla^{\beta}_{f,0},\ \ N^{\gamma}_{r,s+1}$$

such that $N^{\beta}_{f,0} = \mathbf{F}^{\beta}$ and $\beta \in \text{adj}(N^{\gamma}_{r,s+1})$.
In this section, we consider the tabulation of a subset of EPDA consisting of transitions SWAP, WRAP-A, WRAP-B, UNWRAP, WRAP-B+PUSH and WRAP-B+POP.
In order to define items and attending to the form of the transitions, we classify derivations of EPDA into the following types:

Call derivations. Correspond to the propagation of a stack by means of WRAP-B, WRAP-B+PUSH and WRAP-B+POP transitions:

$$\begin{array}{ll} & (\Upsilon\ \lceil \alpha A, a_{h+1} \dots a_n) \\ \vdash {}^* & (\Upsilon\ \lceil A\ \ \Upsilon_1\ \lceil \alpha X B, a_{i+1} \dots a_n) \\ \vdash {}^* & (\Upsilon\ \lceil A\ \ \Upsilon_1\ \lceil \alpha X C, a_{j+1} \dots a_n) \end{array}$$

where $A, B, C, X \in V_S$, $\alpha \in V_S^*$ and $\Upsilon, \Upsilon_1 \in (\lceil V_S^*)^*$. The two occurrences of α denote the same stack in the sense that α is neither consulted nor modified through the derivation. These derivations are independent of Υ and α, so they can be represented by items

$$[A, h \mid B, i, X, C, j, X \mid -, -, -, -]$$

Return derivations. Correspond to the bottom-up propagation of unitary stack by means of UNWRAP transitions:

$$\begin{array}{rl} & (\Upsilon \ [\alpha A, a_{h+1} \dots a_n) \\ \vdash^* & (\Upsilon \ [A \ \ \Upsilon_1 \ [\alpha X B, a_{i+1} \dots a_n) \\ \vdash^* & (\Upsilon \ [A \ \ \Upsilon_1 \ [B \ \ \Upsilon_2 \ [\alpha D, a_{p+1} \dots a_n) \\ \vdash^* & (\Upsilon \ [A \ \ \Upsilon_1 \ [B \ \ \Upsilon_2 \ [E, a_{q+1} \dots a_n) \\ \vdash^* & (\Upsilon \ [A \ \ \Upsilon_1 \ [C, a_{j+1} \dots a_n) \end{array}$$

where $A, B, C, D, E, X \in V_S$, $\alpha \in V_S^*$, $\Upsilon, \Upsilon_1, \Upsilon_2 \in ([V_S^*)^*$ and α is passed unaffected through derivation. These derivations are independent of Υ but not with respect to the subderivation $([\alpha D, a_{p+1} \dots a_n) \vdash^* ([E, a_{q+1} \dots a_n)$, so they are be represented in compact form by items

$$[A, h \mid B, i, X, C, j, - \mid D, p, E, q]$$

Special point derivations. When $\alpha X = \epsilon$ we have a particular case of previous derivations:

$$(\Upsilon \ [B, a_{i+1} \dots a_n) \vdash^* (\Upsilon \ [C, a_{j+1} \dots a_n)$$

where $B, C \in V_S$, and $\Upsilon \in ([V_S^*)^*$. These derivations can be represented by items

$$[-, - \mid B, i, -, C, j, - \mid -, -, -, -]$$

To combine items, we use the set of inference rules shown in figures 4 and 5. Each rule is of the form $\frac{\eta_1, \dots, \eta_k}{\eta'}$ *trans*, meaning that if all antecedents η_i are tabulated items and there exist the transitions *trans*, then the consequent item η' should be created. In order to simplify the inference rules, but without loss of generality, we have considered that scanning is only performed by SWAP transitions. The computation starts with the initial item $[-, - \mid \$_0, 0, -, \$_0, 0, - \mid -, -, -, -]$. An input string $a_1 \dots a_n$ has been recognized if the final item $[-, - \mid \$_0, 0, -, \$_f, n, - \mid -, -, -, -]$ is present. It can be proved that handling items with the inference rules is equivalent to applying the transitions on the whole stacks.
To illustrate the relation between EPDA and L-LIA, figures 4 and 5 show the transitions of both models of automata that must be considered to apply a given inference rule. Therefore, the proposed tabulated technique can be also applied to L-LIA working with transitions SWAP, WRAP-A, WRAP-B, UNWRAP, WRAP-B+PUSH and WRAP-B+POP.

6. Conclusion

Embedded Push-Down Automata have been redefined: finite-state control has been eliminated and several kinds of transition have been defined. We have also shown that the new definition preserves the equivalence with tree adjoining languages and that tabulation techniques are possible to execute these automata in polynomial time with respect to the length of the input string.

References

DE LA CLERGERIE E. & ALONSO PARDO M. (1998). A tabular interpretation of a class of 2-Stack Automata. In *COLING-ACL'98, 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Proceedings of the Conference*, volume II, p. 1333–1339, Montreal, Quebec, Canada: ACL.

Rule	EPDA transition	L-LIA transition
$\dfrac{[A,h \mid B,i,X,C,j,X \mid -,-,-,-]}{[A,h \mid B,i,X,F,k,X \mid -,-,-,-]}$ where $k=j$ if $a=\epsilon$ and $k=j+1$ if $a \in V_T$	$C \xmapsto{a} F$	$C[\circ\circ] \xmapsto{a} F[\circ\circ]$
$\dfrac{[A,h \mid B,i,X,C,j,- \mid D,p,E,q]}{[A,h \mid B,i,X,F,k,- \mid D,p,E,q]}$ where $k=j$ if $a=\epsilon$ and $k=j+1$ if $a \in V_T$	$C \xmapsto{a} F$	$C[\circ\circ] \xmapsto{a} F[\circ\circ]$
$\dfrac{[A,h \mid B,i,X,C,j,X \mid -,-,-,-]}{[-,- \mid F,j,-,F,j,- \mid -,-,-,-]}$	$C \longmapsto C, [F$	$C[\circ\circ] \longmapsto C[\circ\circ]\ F[\,]$
$\dfrac{[A,h \mid B,i,X,C,j,- \mid D,p,E,q]}{[-,- \mid F,j,-,F,j,- \mid -,-,-,-]}$	$C \longmapsto C, [F$	$C[\circ\circ] \longmapsto C[\circ\circ]\ F[\,]$
$\dfrac{[A,h \mid B,i,X,C,j,X \mid -,-,-,-]}{[A,h \mid F,j,X,F,j,X \mid -,-,-,-]}$	$C \longmapsto [C, F$	$C[\circ\circ] \longmapsto C[\,]\ F[\circ\circ]$
$\dfrac{[A,h \mid B,i,X,C,j,- \mid D,p,E,q]}{[-,- \mid F,j,-,F,j,- \mid -,-,-,-]}$	$C \longmapsto [C, F$	$C[\circ\circ] \longmapsto C[\,]\ F[\circ\circ]$
$\dfrac{[A,h \mid B,i,X,C,j,X \mid -,-,-,-]}{[C,j \mid F,j,X',F,j,X' \mid -,-,-,-]}$	$C \longmapsto [C, X'F$	$C[\circ\circ] \longmapsto C[\,]\ F[\circ\circ X']$
$\dfrac{[A,h \mid B,i,X,C,j,- \mid D,p,E,q]}{[C,j \mid F,j,X',F,j,X' \mid -,-,-,-]}$	$C \longmapsto [C, X'F$	$C[\circ\circ] \longmapsto C[\,]\ F[\circ\circ X']$
$\dfrac{\begin{array}{c}[A,h \mid B,i,X,C,j,X \mid -,-,-,-]\\ [M,m \mid N,t,X',A,h,X' \mid -,-,-,-]\end{array}}{[M,m \mid F,j,X',F,j,X' \mid -,-,-,-]}$	$XC \longmapsto [C, F$	$C[\circ\circ X] \longmapsto C[\,]\ F[\circ\circ]$
$\dfrac{\begin{array}{c}[A,h \mid B,i,X,C,j,X \mid -,-,-,-]\\ [M,m \mid N,t,X',A,h,- \mid D,p,E,q]\end{array}}{[-,- \mid F,j,-,F,j,- \mid -,-,-,-]}$	$XC \longmapsto [C, F$	$C[\circ\circ X] \longmapsto C[\,]\ F[\circ\circ]$

Figure 4: Tabulation rules

GAZDAR G. (1987). Applicability of indexed grammars to natural languages. In U. REYLE & C. ROHRER, Eds., *Natural Language Parsing and Linguistic Theories*, p. 69–94. D. Reidel Publishing Company.

LANG B. (1991). Towards a uniform formal framework for parsing. In M. TOMITA, Ed., *Current Issues in Parsing Technology*, p. 153–171. Norwell, MA, USA: Kluwer Academic Publishers.

NEDERHOF M.-J. (1999). Models of tabulation for TAG parsing. In *Proc. of the Sixth Meeting on Mathematics of Language (MOL 6)*, p. 143–158, Orlando, Florida, USA.

VIJAY-SHANKER K. (1988). *A Study of Tree Adjoining Grammars*. PhD thesis, University of Pennsylvania. Available as Technical Report MS-CIS-88-03 LINC LAB 95 of the Department of Computer and Information Science, University of Pennsylvania.

WEIR D. J. (1994). Linear iterated pushdowns. *Computational Intelligence*, **10** (4), p. 422–430.

Rule	**EPDA transition**	**L-LIA transition**
$\dfrac{\begin{array}{c}[-,- \mid F',j,-,F,k,- \mid -,-,-,-]\\ [A,h \mid B,i,X,C,j,X \mid -,-,-,-]\end{array}}{[A,h \mid B,i,X,G,k,X \mid -,-,-,-]}$	$\begin{array}{c}C \longmapsto C, [F'\\ C, [F \longmapsto G\end{array}$	$\begin{array}{c}C[\circ\circ] \longmapsto C[\circ\circ]\ F'[\,]\\ C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]\end{array}$
$\dfrac{\begin{array}{c}[-,- \mid F',j,-,F,k,- \mid -,-,-,-]\\ [A,h \mid B,i,X,C,j,- \mid D,p,E,q]\end{array}}{[A,h \mid B,i,X,G,k,- \mid D,p,E,q]}$	$\begin{array}{c}C \longmapsto C, [F'\\ C, [F \longmapsto G\end{array}$	$\begin{array}{c}C[\circ\circ] \longmapsto C[\circ\circ]\ F'[\,]\\ C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]\end{array}$
$\dfrac{\begin{array}{c}[A,h \mid F',j,X,F,k,- \mid D,p,E,q]\\ [A,h \mid B,i,X,C,j,X \mid -,-,-,-]\end{array}}{[A,h \mid B,i,X,G,k,- \mid D,p,E,q]}$	$\begin{array}{c}C \longmapsto [C, F'\\ C, [F \longmapsto G\end{array}$	$\begin{array}{c}C[\circ\circ] \longmapsto C[\,]\ F'[\circ\circ]\\ C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]\end{array}$
$\dfrac{\begin{array}{c}[-,- \mid F',j,-,F,k,- \mid -,-,-,-]\\ [A,h \mid B,i,X,C,j,- \mid D,p,E,q]\end{array}}{[A,h \mid B,i,X,G,k,- \mid D,p,E,q]}$	$\begin{array}{c}C \longmapsto [C, F'\\ C, [F \longmapsto G\end{array}$	$\begin{array}{c}C[\circ\circ] \longmapsto C[\,]\ F'[\circ\circ]\\ C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]\end{array}$
$\dfrac{\begin{array}{c}[C,j \mid F',j,X',F,k,- \mid D,p,E,q]\\ [A,h \mid B,i,X,C,j,X \mid -,-,-,-]\\ [A,h \mid D,p,X,E,q,- \mid O,u,P,v]\end{array}}{[A,h \mid B,i,X,G,k,- \mid O,u,P,v]}$	$\begin{array}{c}C \longmapsto [C, X'F'\\ C, [F \longmapsto G\end{array}$	$\begin{array}{c}C[\circ\circ] \longmapsto C[\,]\ F'[\circ\circ X']\\ C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]\end{array}$
$\dfrac{\begin{array}{c}[C,j \mid F',j,X',F,k,- \mid O,u,P,v]\\ [A,h \mid B,i,X,C,j,- \mid D,p,E,q]\\ [-,- \mid O,u,-,P,v,- \mid -,-,-,-]\end{array}}{[A,h \mid B,i,X,G,k,- \mid D,p,E,q]}$	$\begin{array}{c}C \longmapsto [C, X'F'\\ C, [F \longmapsto G\end{array}$	$\begin{array}{c}C[\circ\circ] \longmapsto C[\,]\ F'[\circ\circ X']\\ C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]\end{array}$
$\dfrac{\begin{array}{c}[M,m \mid F',j,X',F,k,- \mid D,p,E,q]\\ [A,h \mid B,i,X,C,j,X \mid -,-,-,-]\\ [M,m \mid N,t,X',A,h,X' \mid -,-,-,-]\end{array}}{[A,h \mid B,i,X,G,k,- \mid F',j,F,k]}$	$\begin{array}{c}XC \longmapsto [C, F'\\ C, [F \longmapsto G\end{array}$	$\begin{array}{c}C[\circ\circ X] \longmapsto C[\,]\ F'[\circ\circ]\\ C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]\end{array}$
$\dfrac{\begin{array}{c}[-,- \mid F',j,-,F,k,- \mid -,-,-,-]\\ [A,h \mid B,i,X,C,j,X \mid -,-,-,-]\\ [M,m \mid N,t,X',A,h,- \mid D,p,E,q]\end{array}}{[A,h \mid B,i,X,G,k,- \mid F',j,F,k]}$	$\begin{array}{c}XC \longmapsto [C, F'\\ C, [F \longmapsto G\end{array}$	$\begin{array}{c}C[\circ\circ X] \longmapsto C[\,]\ F'[\circ\circ]\\ C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]\end{array}$

Figure 5: Tabulation rules