Tabulation of Automata for Tree Adjoining Languages

Miguel A. Alonso Pardo
David Cabrero Souto
Departamento de Computación
Universidad de La Coruña
Campus de Elviña s/n
15071 La Coruña, Spain
{alonso,cabrero}@dc.fi.udc.es

Eric de la Clergerie
INRIA
Domaine de Voluceau
Rocquecourt, B.P. 105
78153 Le Chesnay Cedex, France
Eric.De_La_Clergerie@inria.fr

Abstract

We try to provide a common framework to clarify the relationships between different automata and their associated tabulation techniques for Tree Adjoning Languages, a subclass of Mildly Context-Sensitive Languages. We have chosen Logic Push-down Automata working with Linear Indexed Grammars as a starting point. Several tabulation techniques for different parsing strategies are proposed and compared with previous approaches.

1 Introduction

The class of Mildly Context-Sensitive Languages (MCSL) is placed between context-free languages and context-sensitive languages. An important subclass in MCSL is that of Tree Adjoining Languages, which can be described by several grammar formalisms which have been shown to be equivalent with respect to their weak generative capacity (Vijay-Shanker and Weir, 1994): Tree Adjoining Grammars (Joshi and Schabes, 1997), Linear Indexed Grammars (Gazdar, 1987), Head Grammars (Pollard, 1984) and Combinatory Categorial Grammars (Steedman, 1986). Several parsing algorithms have been proposed for all of them, but the design of correct and efficient parsing algorithms is a difficult task that could be simplified by providing a separation between the description of the parsing strategy and the execution of the parser. A way to do that is to define a class of automata to describe the parsing strategy, which could be executed using some tabulation technique.

Several classes of automata have been proposed for tree adjoining languages: Embedded Push-down Automata (Vijay-Shanker, 1988), Bottom-up Embedded Push-down Automata (Schabes and Vijay-Shanker, 1990), 2-Stack Automata (Becker, 1994), Strongly-driven 2–Stack Automata (de la Clergerie and Alonso Pardo, 1998), Bottom-up 2–Stack Automata (de la Clergerie et al., 1998) and Linear Indexed Automata (Nederhof, 1998b), but tabulation techniques have only been designed for the latter three classes. Moreover, the parsing strategies that can be described using each of them are different as are the tabulation techniques. In contrast, in the realm of context-free parsing and logic programming there exist standard operational devices with also standard tabulation techniques: push-down automata and Logic Push-down Automata (Lang, 1988; de la Clergerie and Lang, 1994).

In this paper we try to clarify the relationships between the different classes of automata. Following Nederhof (1998b), we have chosen to take Logic Push-Down Automata (LPDA) as a starting point due to the relationships between Linear Indexed Grammars (LIG) and Definite Clause Grammars (DCG) (Pereira and Warren, 1980).

2 Linear Indexed Grammars and Logic Push-Down Automata

Indexed Grammars (Aho, 1968) are an extension of Context-free Grammars with a stack of indices associated with each non-terminal symbol. Linear Indexed Grammars (Gazdar, 1987) are a restricted form of Indexed Grammars in which the index stack of the head non-terminal of each production can be inherited by at most one body non-terminal (the *dependent child*) while the other stacks must have a bounded stack size. Formally, a LIG is a tuple (V_T, V_N, V_I, P, S), where V_T is a finite set of terminals, V_N a finite set of non-terminals, V_I is a finite set of indices, $S \in V_N$ is the start symbol and P is a finite set of productions. Following Gazdar (1987) we consider productions in which at most one element can be pushed on or popped from a stack of indices:

$$\begin{array}{rcl} A_{r,0}[\circ\circ\gamma] & \rightarrow & A_{r,1}[\,]\ \ldots A_{r,i-1}[\,] \\ & & A_{r,i}[\circ\circ\gamma'] \\ & & A_{r,i+1}[\,]\ \ldots A_{r,n_r}[\,] \end{array}$$

$$A_{r,0}[\,] \rightarrow a$$

where $A_j \in V_N$, $a \in V_T^*$ and $\gamma, \gamma' \in V_I \cup \{\epsilon\}$ and for each production, either γ or γ' or both must be ϵ. In any derivation the stack associated to the start symbol must be empty: $S[\,]$.

LIG can be considered as a special case of DCG. For example, a LIG production

$$A[\circ\circ\gamma] \rightarrow B[\,]\ C[\circ\circ]\ D[\,]$$

can be written as a DCG production

```
big_a([gamma|X]) --> big_b([]),
                     big_c(X),
                     big_d([]).
```

and a LIG production

$$A[\,] \rightarrow a$$

can be written as

```
big_a([]) --> "a".
```

Given a parse forest for LIG as defined by Boullier (1995), a path starting from a non-dependent child of a production and following the dependent children is called a *spine*. The index stacks involved along a spine refers to the same index stack, not to copies.

Logic Push-Down Automata, which can be used as an operational device for DCGs parsing, follow the structure of classical push-down automata but stack symbols are replaced by logic atoms on some sets P of predicates, F of functions and X of variable symbols. Transitions are applied to stacks modulo a unification process with some of the topmost elements, with the resulting substitution applied to the whole stack. Formally, a LPDA is defined by a tuple $(P, F, X, \$_0, \$_f, \Theta)$, where $\$_0$ is the initial stack element, $\$_f$ is the final stack element and Θ is a finite set of transitions of the form

SWAP $(B \overset{a}{\mapsto} C)(\xi A) = (\xi C)\sigma$ where $\sigma = \mathrm{mgu}(A, B)$.

PUSH $(B \overset{a}{\mapsto} BC)(\xi A) = (\xi AC)\sigma$ where $\sigma = \mathrm{mgu}(A, B)$.

POP $(DB \overset{a}{\mapsto} C)(\xi EA) = (\xi C)\sigma$ where $\sigma = \mathrm{mgu}(\langle E, A\rangle, \langle D, B\rangle)$.

where a is in $V_T \cup \{\epsilon\}$, A, B, C, D, and E are stacks elements, and ξ is a stack (growing rightward), and $\mathrm{mgu}(x, y)$ is the most general unifier between x and y. A *configuration* for a LPDA is a pair (ξ, w), where ξ denotes the content of the stack and w the input string to be read. The derivation relation between configurations is denoted by $\vdash$ and its transitive and reflexive closure is denoted by $\overset{*}{\vdash}$.

Given a DCG and a parsing strategy, a LPDA can be defined by means of a set of transitions describing the computations that could be applied to the grammar using the parsing strategy. If we restrict the input grammars to those with the form of LIG, we obtain a subclass of LPDA that we call Restricted LPDA and that may serve as operational devices for LIG.

Table 1 shows a generic compilation schema transforming a LIG into a set of transitions of a Restricted LPDA having $\$_0[\,]$ as initial element and $\overleftarrow{S}[\,]$ as final element, where $\overrightarrow{x}$ (resp. $\overleftarrow{x}$) denotes the information predicted (resp. propagated) with respect to x, which can be a non-terminal, a index or an index stack.

We have used the notation of linear indexed grammars as a shorthand of Prolog notation. So, the index stack associated to each non-terminal represents its only argument, where $[\,]$ stands for the empty list and $[\circ\circ\gamma]$ for the list $[\gamma|X]$ where $\circ\circ$ plays the role of a variable X. With respect to the compilation rules, **[INIT]** transitions are used to start the computations, **[CALL]** transitions to call a grammar element which is not a dependent child, **[SCALL]** (*spine call*) to call a dependent child, **[SEL]** to select a production, **[PUB]** to finish the parsing of a production, **[RET]** to continue the parsing process after a non-dependent child has been recognized, **[SRET]** (*spine ret*) to continue the parsing process after a dependent child has been recognized and **[SCAN]** to recognize the terminals in the input string.

Table 2 shows the parsing strategies obtained

[INIT]	$\$_0[\circ\circ] \mapsto \$_0[\circ\circ]\ \nabla_{0,0}[\,]$	
[CALL]	$\nabla_{r,s}[\circ\circ] \mapsto \nabla_{r,s}[\circ\circ]\ \overrightarrow{A_{r,s+1}}[\,]$	$A_{r,0}[\circ\circ\gamma] \rightarrow \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SCALL]	$\nabla_{r,s}[\overrightarrow{\circ\circ}\,\overrightarrow{\gamma}] \mapsto \nabla_{r,s}[\overrightarrow{\circ\circ}\,\overrightarrow{\gamma}]\ \overrightarrow{A_{r,s+1}}[\overrightarrow{\circ\circ}\,\overrightarrow{\gamma'}]$	$A_{r,0}[\circ\circ\gamma] \rightarrow \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SEL]	$\overrightarrow{A_{r,0}}[\circ\circ] \mapsto \nabla_{r,0}[\circ\circ]$	$r \neq 0$
[PUB]	$\nabla_{r,n_r}[\circ\circ] \mapsto \overleftarrow{A_{r,0}}[\circ\circ]$	
[RET]	$\nabla_{r,s}[\circ\circ]\ \overleftarrow{A_{r,s+1}}[\,] \mapsto \nabla_{r,s+1}[\circ\circ]$	$A_{r,0}[\circ\circ\gamma] \rightarrow \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SRET]	$\nabla_{r,s}[\overrightarrow{\circ\circ}\,\overrightarrow{\gamma}]\ \overleftarrow{A_{r,s+1}}[\overleftarrow{\circ\circ}\,\overleftarrow{\gamma'}] \mapsto \nabla_{r,s+1}[\overleftarrow{\circ\circ}\,\overleftarrow{\gamma}]$	$A_{r,0}[\circ\circ\gamma] \rightarrow \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SCAN]	$\overrightarrow{A_{r,0}}[\,] \overset{a}{\mapsto} \overleftarrow{A_{r,0}}[\,]$	$A_{r,0}[\,] \rightarrow a$

Table 1: Generic compilation schema

CF-strategy	indices-strategy	$\overrightarrow{A_{r,s+1}}$	$\overrightarrow{\gamma}$	$\overrightarrow{\circ\circ}$	$\overleftarrow{A_{r,s+1}}$	$\overleftarrow{\gamma}$	$\overleftarrow{\circ\circ}$
BU	BU	$\perp$	ϵ	ϵ	$A_{r,s+1}$	γ	$\circ\circ$
Earley		$\overline{A_{r,s+1}}$	ϵ	ϵ	$\overline{\overline{A_{r,s+1}}}$	γ	$\circ\circ$
TD		$A_{r,s+1}$	ϵ	ϵ	$\perp$	γ	$\circ\circ$
BU	Earley	$\perp$	γ	$\circ\circ$	$A_{r,s+1}$	γ	$\circ\circ$
Earley		$\overline{A_{r,s+1}}$	γ	$\circ\circ$	$\overline{\overline{A_{r,s+1}}}$	γ	$\circ\circ$
TD		$A_{r,s+1}$	γ	$\circ\circ$	$\perp$	γ	$\circ\circ$
BU	TD	$\perp$	γ	$\circ\circ$	$A_{r,s+1}$	ϵ	ϵ
Earley		$\overline{A_{r,s+1}}$	γ	$\circ\circ$	$\overline{\overline{A_{r,s+1}}}$	ϵ	ϵ
TD		$A_{r,s+1}$	γ	$\circ\circ$	$\perp$	ϵ	ϵ

Table 2: Parsing strategies obtained by different instantiations of $\overrightarrow{x}$ and $\overleftarrow{x}$

by different instantiations of $\overrightarrow{x}$ and $\overleftarrow{x}$ in the previous compilation schema. Parsing strategies are specified using a pair of strategies controlling the flow of information, the first one (CF-strategy) dealing with non-terminals while the other (indices-strategy) deals with the indices. In both cases, BU stands for bottom-up, TD for top-down and Earley for mixed strategies. As is usual in LPDA, when a non-terminal A is predicted and propagated we differentiate the two cases using $\overline{A}$ to indicate the prediction and $\overline{\overline{A}}$ to indicate the propagation, although the information transmitted is the same in both cases. It is interesting to remark that strategies in which $\overrightarrow{\circ\circ}$ and $\overleftarrow{\circ\circ}$ are both different from ϵ must include an additional restriction to **[SRET]** transitions indicating that $\overrightarrow{\circ\circ}$ and $\overleftarrow{\circ\circ}$ must unify, i.e. the contents of both stacks must be the same although the stacks themselves are different, as one has been constructed from the empty stack by **[SCALL]** transitions throughout the spine (the *call phase* or descendent phase of the computation of a spine) while the other was constructed from the empty stack by **[SRET]** transitions throughout the spine (the *return phase* or ascendent phase of the computation of a spine).

Standard tabulation techniques for LPDA can be applied to Restricted LPDA, but LIGs usually do not have the properties guaranteeing termination in DCG, such as off-line parsability (Pereira and Warren, 1983) or depth-boundness (Haas, 1989). However, it is possible to design specific tabulation techniques with polynomial complexity with respect to the length of the input string for Restricted LPDA based on the property defined by Vijay-Shanker and Weir (1993), that we call *context-freeness property of LIG*, establishing that if

$$A[\gamma] \stackrel{*}{\Rightarrow} uB[\;]w$$

where $u, w \in V_T^*$, $A, B \in V_N$, $\gamma \in V_I \cup \{\epsilon\}$ and $B[\;]$ is the dependent descendant of $A[\gamma]$, then for each $\Upsilon_1, \Upsilon_2 \in (V_N[V_I^*] \cup V_T)^*$and $\beta \in V_I^*$ we have

$$\Upsilon_1 A[\beta\gamma]\Upsilon_2 \stackrel{*}{\Rightarrow} \Upsilon_1 uB[\beta]w\Upsilon_2$$

In addition, if $B[\beta] \stackrel{*}{\Rightarrow} v$, where $v \in V_T^*$, then $\Upsilon_1 A[\beta\gamma]\Upsilon_2 \stackrel{*}{\Rightarrow} \Upsilon_1 uB[\beta]w\Upsilon_2 \stackrel{*}{\Rightarrow} \Upsilon_1 uvw\Upsilon_2$.

3 Tabulation for $*$-BU strategies

By $*$-BU we denote those strategies which do not predict information with respect to the indices, regardless of the CF-strategy. As an example, table 3 shows the compilation schema corresponding to the BU-BU strategy which generates a Restricted LPDA having $\$_0[\;]$ as initial element and $S[\;]$ as final element. The **[SCALL]** and **[SCAN]** transitions are slightly different from those defined for the generic strategies. This is only for convenience, as the strategy forces the stacks involved in the former kind of transitions to be empty. The correspondence between kinds of transition and compilation rules is shown in table 4.

The key step needed to design a tabulation technique is to determine the information we need to trace with respect to a given derivation. That information is stored into *items*. For Restricted LPDA we consider items having the form [*head* | *tail*] storing a set of non-terminals with the positions of the input string corresponding to the moment when they were pushed, and in some cases the index in the top of the index stack associated to some of the non-terminals. The *head* is used to store the information needed to determine if a transition could be applied and the *tail* is used to store the information needed to guarantee the consistency of the index stacks when transitions are applied.

In the case of $*$-BU strategies, we consider two different types of derivations that can yield a configuration $(\xi\; B[\alpha]\; C[\beta\gamma], a_{j+1}\ldots a_n)$:

Call derivations. Correspond to configurations with $\beta\gamma = \epsilon$:

$$\begin{array}{l}(\xi\; B[\alpha], a_{i+1}\ldots a_n)\\ \stackrel{*}{\vdash} (\xi\; B[\alpha]\; C[\;], a_{j+1}\ldots a_n)\end{array}$$

This kind of derivations can be tabulated using *call items* of the form

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

Due to context-freeness property of LIG, derivations are independent of the values taken by ξ and α and so it is not necessary to store information about these elements.

Return derivations. Correspond to configurations with $\beta \in V_I^*$ and $\gamma \in V_I$:

$$\begin{array}{l}(\xi\; B[\;], a_{i+1}\ldots a_n)\\ \stackrel{*}{\vdash}_{d_1} (\xi\; B[\;]\;\xi_1\; D[\;], a_{p+1}\ldots a_n)\\ \stackrel{*}{\vdash}_{d_2} (\xi\; B[\;]\;\xi_1\; D[\;]\; E[\beta], a_{q+1}\ldots a_n)\\ \stackrel{*}{\vdash}_{d_3} (\xi\; B[\;]\; C[\beta\gamma], a_{j+1}\ldots a_n)\end{array}$$

where $\gamma \in V_I$, $\beta \in V_I^*$. The two occurrences of β denote the same list in the sense that it is passed on unaffected through d_3. This kind of configuration can be tabulated using *return items* of the form

$$[B, i, C, j, \gamma \mid D, p, E, q]$$

To combine items, we use inference rules similar to those used by grammatical deduction systems described by Shieber et al. (1995). Each rule $\frac{\text{ants}}{\text{consq}}\text{trans}$ consists of a list *ants* of antecedent items and a consequent item *consq*, meaning that if all antecedents η_i are present and there exists a transition *trans* then the consequent item should be generated. The rules for $*$-BU strategies are shown in table 5, where $k = j$ if $a = \epsilon$ and $k = j + 1$ if $a = a_{j+1}$. These

[INIT]	$\$_0[\circ\circ] \mapsto \$_0[\circ\circ]\ \nabla_{0,0}[\,]$	
[CALL]	$\nabla_{r,s}[\circ\circ] \mapsto \nabla_{r,s}[\circ\circ]\ \ \bot\,[\,]$	
[SCALL]	$\nabla_{r,s}[\circ\circ] \mapsto \nabla_{r,s}[\circ\circ]\ \ \bot\,[\,]$	
[SEL]	$\bot\,[\circ\circ] \mapsto \nabla_{r,0}[\circ\circ]$	$r \neq 0$
[PUB]	$\nabla_{r,n_r}[\circ\circ] \mapsto A_{r,0}[\circ\circ]$	
[RET]	$\nabla_{r,s}[\circ\circ]\ \ A_{r,s+1}[\,] \mapsto \nabla_{r,s+1}[\circ\circ]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SRET]	$\nabla_{r,s}[\,]\ \ A_{r,s+1}[\circ\circ\gamma'] \mapsto \nabla_{r,s+1}[\circ\circ\gamma]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SCAN]	$\bot\,[\circ\circ] \stackrel{a}{\mapsto}\ A_{r,0}[\circ\circ]$	$A_{r,0}[\,] \to a$

Table 3: Compilation schema corresponding to the BU-BU strategy

Transition	**Compilation rule**
$C[\circ\circ] \stackrel{a}{\mapsto} C[\circ\circ]\ F[\,]$	**[INIT] [CALL] [SCALL]**
$C[\circ\circ] \stackrel{a}{\mapsto} F[\circ\circ]$	**[SEL] [PUB] [SCAN]**
$B[\circ\circ]\ C[\,] \stackrel{a}{\mapsto} F[\circ\circ]$	**[RET]**
$B[\,]\ C[\circ\circ\gamma] \stackrel{a}{\mapsto} F[\circ\circ\gamma']$	**[SRET]**

Table 4: Correspondence between types of transition and compilation rules for $*$-BU strategies

rules, which are very similar to those proposed by Nederhof (1998b), give a time complexity $\mathcal{O}(n^6)$, using partial application to decrease the complexity of the last rule from $\mathcal{O}(n^7)$ to $\mathcal{O}(n^6)$, and a space complexity $\mathcal{O}(n^4)$ with respect to the length n of the input string.

4 Tabulation for $*$-Earley strategies

By $*$-Earley we denote those strategies which predict and propagate information with respect to the indices, regardless of the CF-strategy. As an example, the compilation schema corresponding to the Earley-Earley strategy is shown in table 6. The Restricted LPDA generated has $\$_0[\,]$ as initial element and $\overline{\overline{S}}[\,]$ as final element. The types of transition corresponding to each compilation rule are shown in table 7.

To design a tabulation technique for this kind of strategies, we need to consider three different types of derivations:

Call derivations. Correspond to the transmission of the index stack during the call phase:

$$\begin{array}{l} (\xi\ A[\delta], a_{h+1} \dots a_n) \\ \overset{*}{\vdash}_{d_1} (\xi\ A[\delta]\ \xi_1\ B[\delta\alpha], a_{i+1} \dots a_n) \\ \overset{*}{\vdash}_{d_2} (\xi\ A[\delta]\ \xi_1\ B[\delta\alpha]\ C[\delta\gamma], a_{j+1} \dots a_n) \end{array}$$

where $\gamma \in V_I$, $\delta, \alpha \in V_I^*$, $0 \leq |\alpha| \leq 2$ and the index stack δ is passed unaffected through derivations. If $\delta\alpha{=}\delta$, then d_1 is a empty derivation and $A = B$. This kind of derivation is tabulated by means of *call items* of the form

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

Return derivations. Correspond to the transmission of the index stack during the

$$\frac{[B,i,C,j,\gamma \mid D,p,E,q]}{[C,j,F,k,- \mid -,-,-,-]} \quad C[\circ\circ] \overset{a}{\mapsto} C[\circ\circ]\ F[\,]$$

$$\frac{[B,i,C,j,\gamma \mid D,p,E,q]}{[B,i,F,k,\gamma \mid D,p,E,q]} \quad C[\circ\circ] \overset{a}{\mapsto} F[\circ\circ]$$

$$\frac{\begin{array}{l}[B,i,C,j,- \mid -,-,-,-] \\ [M,m,B,i,\gamma \mid D,p,E,q]\end{array}}{[M,m,F,k,\gamma \mid D,p,E,q]} \quad B[\circ\circ]\ C[\,] \overset{a}{\mapsto} F[\circ\circ]$$

$$\frac{\begin{array}{l}[B,i,C,j,\gamma \mid D,p,E,q] \\ [M,m,B,i,- \mid -,-,-,-]\end{array}}{[M,m,F,k,\gamma \mid D,p,E,q]} \quad B[\,]\ C[\circ\circ] \overset{a}{\mapsto} F[\circ\circ]$$

$$\frac{\begin{array}{l}[B,i,C,j,\gamma \mid D,p,E,q] \\ [M,m,B,i,- \mid -,-,-,-]\end{array}}{[M,m,F,k,\gamma' \mid B,i,C,j]} \quad B[\,]\ C[\circ\circ] \overset{a}{\mapsto} F[\circ\circ\gamma']$$

$$\frac{\begin{array}{l}[B,i,C,j,\gamma \mid D,p,E,q] \\ [M,m,B,i,- \mid -,-,-,-] \\ [D,p,E,q,\gamma' \mid O,u,P,v]\end{array}}{[M,m,F,k,\gamma' \mid O,u,P,v]} \quad B[\,]\ C[\circ\circ\gamma] \overset{a}{\mapsto} F[\circ\circ]$$

Table 5: Combination rules for $*$-BU strategies

[INIT]	$\$_0[\circ\circ] \mapsto \$_0[\circ\circ]\ \nabla_{0,0}[\,]$	
[CALL]	$\nabla_{r,s}[\circ\circ] \mapsto \nabla_{r,s}[\circ\circ]\ \overline{A_{r,s+1}}[\,]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SCALL]	$\nabla_{r,s}[\circ\circ\gamma] \mapsto \nabla_{r,s}[\circ\circ\gamma]\ \overline{A_{r,s+1}}[\circ\circ\gamma']$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SEL]	$\overline{A_{r,s+1}}[\circ\circ] \mapsto \nabla_{r,0}[\circ\circ]$	$r \neq 0$
[PUB]	$\nabla_{r,n_r}[\circ\circ] \mapsto \overline{\overline{A_{r,0}}}[\circ\circ]$	
[RET]	$\nabla_{r,s}[\circ\circ]\ \overline{\overline{A_{r,s+1}}}[\,] \mapsto \nabla_{r,s+1}[\circ\circ]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SRET]	$\nabla_{r,s}[\circ\circ\gamma]\ \overline{\overline{A_{r,s+1}}}[\circ\circ\gamma'] \mapsto \nabla_{r,s+1}[\circ\circ\gamma]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SCAN]	$\overline{A_{r,0}}[\,] \overset{a}{\mapsto} \overline{\overline{A_{r,0}}}[\,]$	$A_{r,0}[\,] \to a$

Table 6: Compilation schema corresponding to the Earley-Earley strategy

Transition	Compilation rule
$C[\circ\circ] \overset{a}{\mapsto} C[\circ\circ]\ F[\,]$	**[INIT] [CALL]**
$C[\circ\circ\gamma] \overset{a}{\mapsto} C[\circ\circ\gamma]\ F[\circ\circ\gamma']$	**[SCALL]**
$C[\circ\circ] \overset{a}{\mapsto} F[\circ\circ]$	**[SEL] [PUB]**
$B[\circ\circ]\ C[\,] \overset{a}{\mapsto} F[\circ\circ]$	**[RET]**
$B[\circ\circ_1\gamma]\ C[\circ\circ_2\gamma'] \overset{a}{\mapsto} F[\circ\circ_2\gamma]$	**[SRET]**
$C[\,] \overset{a}{\mapsto} F[\,]$	**[SCAN]**

Table 7: Correspondence between types of transition and compilation rules for $*$-Earley strategies

return phase:

$$\begin{aligned}
&(\xi\ A[\delta], a_{h+1} \ldots a_n)\\
&\overset{*}{\vdash}_{d_1} (\xi\ A[\delta]\ \xi_1\ B[\delta\alpha], a_{i+1} \ldots a_n)\\
&\overset{*}{\vdash}_{d_2} (\xi\ A[\delta]\ \xi_1\ B[\delta\alpha]\ \xi_2\ D[\delta\gamma], a_{p+1} \ldots a_n)\\
&\overset{*}{\vdash}_{d_3} (\xi\ A[\delta]\ \xi_1\ B[\delta\alpha]\ \xi_2\ D[\delta\gamma]\ E[\beta], a_{q+1} \ldots a_n)\\
&\overset{*}{\vdash}_{d_4} (\xi\ A[\delta]\ \xi_1\ B[\delta\alpha]\ C[\beta\gamma], a_{j+1} \ldots a_n)
\end{aligned}$$

where $\gamma \in V_I$, $\alpha, \beta, \delta \in V_I^*$, $|\delta| = |\beta|$, $0 \leq |\alpha| \leq 2$. We have that $\delta = \delta_1 \ldots \delta_z$ and $\beta = \beta_1 \ldots \beta_z$ where $\delta_i, \beta_i \in V_I$. It is mandatory that $\forall i, \delta_i = \beta_i$, i.e. the contents of both stacks are the same but the stacks are different: one has been constructed during the call phase, the other during the return phase. The *return items* that represent this kind of derivations are:

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

Special point derivations. Correspond to the creation or termination of a spine:

$$\begin{aligned}
&(\xi_2\ B[\delta], a_{i+1} \ldots a_n)\\
&\overset{*}{\vdash}_{d} (\xi_2\ B[\delta]\ C[\,], a_{j+1} \ldots a_n)
\end{aligned}$$

and are represented in the tabular framework by *special point items* of the form

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

These derivations are similar to context-free derivations, as actually each spine is independent of the other ones. This kind of derivation was not distinguishable from call derivations in the case of $*$-BU strategies.

The two leading elements A and h of items should in theory be placed in the tail of the items, as they are used to check the consistency of the index stacks when a transition is applied. Instead, they are placed in front of the head to emphasize that A was pushed onto the stack before the other elements in the item. It is also interesting to remark that these items are similar to those proposed by Nederhof (1997) for the Earley-like parsing algorithm for TAG preserving the valid prefix property, which also store information about five points in a derivation. The rules combining items are shown in table 8, where $k = j$ if $a = \epsilon$ and $k = j + 1$ if $a = a_{j+1}$. These rules give a time complexity $\mathcal{O}(n^7)$ but applying the technique described in (de la Clergerie and Alonso Pardo, 1998; Nederhof, 1997) the rule corresponding to the transitions $B[\circ\circ_1]\ C[\circ\circ_2\gamma'] \overset{a}{\mapsto} F[\circ\circ_2]$ can be decomposed into the following two rules

$$\frac{\begin{array}{c}[B, i \mid B, i, C, j, \gamma' \mid D, p, E, q]\\ [M, m \mid D, p, E, q, \gamma \mid O, u, P, v]\end{array}}{[[B, i, C, j, \gamma' \mid O, u, P, v]]}$$

$$\frac{\begin{array}{c}[[B, i, C, j, \gamma' \mid O, u, P, v]]\\ [M, m \mid, N, t, B, i, \gamma \mid -, -, -, -]\\ [M, m \mid D, p, E, q, \gamma \mid O, u, P, v]\end{array}}{[M, m \mid N, t, F, k, \gamma \mid O, u, P, v]}$$

where $[[B, i, C, j, \gamma' \mid O, u, P, v]]$ is an intermediate pseudo-item, obtaining a final time complexity $\mathcal{O}(n^6)$. The space complexity is $\mathcal{O}(n^5)$.

$$\frac{[A,h \mid B,i,C,j,\gamma \mid D,p,E,q]}{[-,- \mid C,j,F,k,- \mid -,-,-,-]} \quad C[\circ\circ] \overset{a}{\mapsto} C[\circ\circ]\ F[\,]$$

$$\frac{[A,h \mid B,i,C,j,\gamma \mid -,-,-,-]}{[A,h \mid C,j,F,k,\gamma \mid -,-,-,-]} \quad C[\circ\circ] \overset{a}{\mapsto} C[\circ\circ]\ F[\circ\circ]$$

$$\frac{[A,h \mid B,i,C,j,\gamma \mid -,-,-,-]}{[C,j \mid C,j,F,k,\gamma' \mid -,-,-,-]} \quad C[\circ\circ] \overset{a}{\mapsto} C[\circ\circ]\ F[\circ\circ\gamma']$$

$$\frac{\begin{array}{c}[A,h \mid B,i,C,j,\gamma \mid -,-,-,-]\\ [M,m \mid N,t,A,h,\gamma' \mid -,-,-,-]\end{array}}{[M,m \mid C,j,F,k,\gamma' \mid -,-,-,-]} \quad C[\circ\circ\gamma] \overset{a}{\mapsto} C[\circ\circ\gamma]\ F[\circ\circ]$$

$$\frac{[A,h \mid B,i,C,j,\gamma \mid D,p,E,q]}{[A,h \mid B,i,F,k,\gamma \mid D,p,E,q]} \quad C[\circ\circ] \overset{a}{\mapsto} F[\circ\circ]$$

$$\frac{\begin{array}{c}[-,- \mid B,i,C,j,- \mid -,-,-,-]\\ [M,m \mid N,t,B,i,\gamma \mid D,p,E,q]\end{array}}{[M,m \mid N,t,F,k,\gamma \mid D,p,E,q]} \quad B[\circ\circ]\ C[\,] \overset{a}{\mapsto} F[\circ\circ]$$

$$\frac{\begin{array}{c}[A,h \mid B,i,C,j,\gamma \mid D,p,E,q]\\ [A,h \mid M,m,B,i,\gamma \mid -,-,-]\end{array}}{[A,h \mid M,m,F,k,\gamma \mid D,p,E,q]} \quad B[\circ\circ_1]\ C[\circ\circ_2] \overset{a}{\mapsto} F[\circ\circ_2]$$

$$\frac{\begin{array}{c}[A,h \mid B,i,C,j,\gamma' \mid D,p,E,q]\\ [A,h \mid N,t,M,m,\eta \mid -,-,-,-]\\ [M,m \mid O,u,B,i,\gamma \mid -,-,-,-]\end{array}}{[M,m \mid O,u,F,k,\gamma \mid B,i,C,j]} \quad B[\circ\circ_1\gamma]\ C[\circ\circ_2] \overset{a}{\mapsto} F[\circ\circ_2\gamma]$$

$$\frac{\begin{array}{c}[B,i \mid B,i,C,j,\gamma' \mid D,p,E,q]\\ [M,m \mid, N,t,B,i,\gamma \mid -,-,-]\\ [M,m \mid D,p,E,q,\gamma \mid O,u,P,v]\end{array}}{[M,m \mid N,t,F,k,\gamma \mid O,u,P,v]} \quad B[\circ\circ_1]\ C[\circ\circ_2\gamma'] \overset{a}{\mapsto} F[\circ\circ_2]$$

$$\frac{[-,- \mid B,i,C,j,- \mid -,-,-,-]}{[-,- \mid C,j,F,k,- \mid -,-,-,-]} \quad C[\,] \overset{a}{\mapsto} F[\,]$$

Table 8: Combining rules ofr $*$-Earley strategies

5 Tabulation for $*$-TD strategies

By $*$-TD we denote those strategies which predict information with respect to the indices but do not propagate any information about them during the return phase. As an example, table 9 shows the compilation schema corresponding to the TD-TD strategy, where **[SCAN]** transitions are slightly different from those defined for the generic strategies for convenience, as the strategy forces the stacks involved in that kind of transitions to be empty. The resulting Restricted LPDA has $\$_0[\ ]$ as initial element and $\nabla_{0,n_0}[\ ]$ as final element. The transition types corresponding to each compilation rule are listed in Table 10.

The tabulation technique for these strategies is based on three types of derivations, similar to those described for $*$-Earley strategies, except for the emptiness of index stacks during return phases. To denote items, we need to introduce a mark $\otimes \in \{\nearrow, \rightarrow, \searrow, \models\}$: this mark is used as a "guide" to indicate what kind of checking must be performed during the return phase of the strategy due to the lack of information about the index stacks during the return. More precisely, $\nearrow$ indicates that an index was pushed on the stack during the call phase, $\searrow$ indicates a pop, $\rightarrow$ indicates no change and $\models$ indicates the creation of an empty index stack.

For this kind of strategies, we need to consider three different types of derivations in order to design a tabulation technique:

Call derivations. Correspond to the prediction of the index stack during the call phase:

$$\begin{array}{l}(\xi\ A[\delta], a_{h+1}\dots a_n)\\ \overset{*}{\vdash}_{d_1} (\xi\ A[\delta]\ \xi_1\ B[\delta\alpha], a_{i+1}\dots a_n)\\ \overset{*}{\vdash}_{d_2} (\xi\ A[\delta]\ \xi_1\ B[\delta\alpha]\ C[\delta\gamma], a_{j+1}\dots a_n)\end{array}$$

where $\gamma \in V_I$, $\delta, \alpha \in V_I^*$ and $0 \le |\alpha| \le 2$. They are represented by *call items* of the form

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

where the value of $\otimes$ is equals to $\nearrow$ if and only if $\alpha = \epsilon$; to $\rightarrow$ if and only if $\alpha = \gamma$ and it is equals to $\searrow$ if and only if $\alpha = \gamma\gamma'$ and $\gamma' \in V_I$.

Return derivations. Correspond to the transmission of a empty index stack during the return phase:

$$\begin{array}{l}(\xi\ A[\delta], a_{h+1}\dots a_n)\\ \overset{*}{\vdash}_{d_1} (\xi\ A[\delta]\ \xi_1\ B[\delta\alpha], a_{i+1}\dots a_n)\\ \overset{*}{\vdash}_{d_2} (\xi\ A[\delta]\ \xi_1\ B[\delta\alpha]\ \xi_2\ D[\delta\gamma], a_{p+1}\dots a_n)\\ \overset{*}{\vdash}_{d_3} (\xi\ A[\delta]\ \xi_1\ B[\delta\alpha]\ \xi_2\ D[\delta\gamma]\ E[\,], a_{q+1}\dots a_n)\\ \overset{*}{\vdash}_{d_4} (\xi\ A[\delta]\ \xi_1\ B[\delta\alpha]\ C[\,], a_{j+1}\dots a_n)\end{array}$$

where $\gamma \in V_I$, $\alpha, \delta \in V_I^*$ and $0 \le |\alpha| \le 2$. They are represented by *return items* of the form

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

Special point derivations. Correspond to the creation or termination of a spine:

$$\begin{array}{l}(\xi_2\ B[\delta], a_{i+1}\dots a_n)\\ \overset{*}{\vdash}_{d} (\xi_2\ B[\delta]\ C[\,], a_{j+1}\dots a_n)\end{array}$$

and are represented by *special point items* of the form

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

Items are combined using the set of combining rules shown in table 11, very similar to the rules introduced for $*$-Earley strategies, where $k = j$ if $a = \epsilon$ and $k = j + 1$ if $a = a_{j+1}$. As in the case of $*$-Earley strategies, time complexity is $\mathcal{O}(n^7)$ but can be reduced to $\mathcal{O}(n^6)$ decomposing the last rule into two rules. Space complexity is $\mathcal{O}(n^5)$.

6 Relations with tabulation techniques for other automata

6.1 SD–2SA

Strongly-Driven 2-Stack Automata (de la Clergerie and Alonso Pardo, 1998) are an extension of push-down automata working on a pair of asymmetric stacks, a master stack **MS** and an auxiliary stack **AS**. These stacks are partitioned into sessions. Computations in each session are performed in one of two modes *write* **w** and *erase* **e**. A session starts in mode write and switches at some point to mode erase. In mode write (resp. erase), no element can be popped from

[INIT]	$\$_0[\circ\circ] \mapsto \$_0[\circ\circ]\ \nabla_{0,0}[\,]$	
[CALL]	$\nabla_{r,s}[\circ\circ] \mapsto \nabla_{r,s}[\circ\circ]\ \ A_{r,s+1}[\,]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SCALL]	$\nabla_{r,s}[\circ\circ\gamma] \mapsto \nabla_{r,s}[\circ\circ\gamma]\ \ A_{r,s+1}[\circ\circ\gamma']$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SEL]	$A_{r,0}[\circ\circ] \mapsto \nabla_{r,0}[\circ\circ]$	$r \neq 0$
[PUB]	$\nabla_{r,n_r}[\circ\circ] \mapsto \perp [\circ\circ]$	$r \neq 0$
[RET]	$\nabla_{r,s}[\circ\circ]\ \ \perp [\,] \mapsto \nabla_{r,s+1}[\circ\circ]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SRET]	$\nabla_{r,s}[\circ\circ]\ \ \perp [\,] \mapsto \nabla_{r,s+1}[\,]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SCAN]	$A_{r,0}[\circ\circ] \overset{a}{\mapsto} \perp [\circ\circ]$	$A_{r,0}[\,] \to a$

Table 9: Compilation schema corresponding to the TD-TD strategy

Transition	Compilation rule
$C[\circ\circ] \overset{a}{\mapsto} C[\circ\circ]\ F[\,]$	**[INIT] [CALL]**
$C[\circ\circ\gamma] \overset{a}{\mapsto} C[\circ\circ\gamma]\ F[\circ\circ\gamma']$	**[SCALL]**
$C[\circ\circ] \overset{a}{\mapsto} F[\circ\circ]$	**[SEL] [PUB] [SCAN]**
$B[\circ\circ]\ C[\,] \overset{a}{\mapsto} F[\circ\circ]$	**[RET]**
$B[\circ\circ]\ C[\,] \overset{a}{\mapsto} F[\,]$	**[SRET]**

Table 10: Correspondence between types of transition and compilation rules for $*$-TD strategies

(resp. pushed to) the master stack. Switching back from **e** to **w** is not allowed. This requirement means that a same session stack is never used twice by "descendants" of an element in **MS**. Exiting a session is only possible when reaching back, with an empty session stack and in mode erase, the **MS** element that initiated the session. Each pushing on **MS** done in write mode leaves a mark in **MS** about the action that took place on **AS**: $\nearrow$ for a push, $\searrow$ for a pop and $\to$ if any action has taken place. When a new session is created, the mark $\models$ is used. The popping of the marks in erase mode will guide which action should take place on **AS**, the erasing actions faithfully retracing the writing actions.

To design the tabulation technique for SD–2SA, two kinds of derivations were considered. *Context-free derivations* include call derivations and special points derivations and are represented by context-free items

$$\langle A, h\rangle\langle B, i\rangle \otimes \langle C, j, \gamma\rangle m$$

with $\otimes \in \{\nearrow, \searrow, \to, \models\}, m \in \{\mathbf{w}, \mathbf{e}\}$. *Escaped context-free derivations* correspond to return derivations and are represented by escaped context-free items of the form

$$\langle A, h\rangle\langle B, i\rangle \otimes [\langle D, p, \eta\rangle\langle E, q\rangle]\langle C, j, \gamma\rangle\mathbf{e}$$

The tabulation technique for SD–2SA looks like a combination of the tabular techniques described for $*$-Earley and $*$-TD strategies. This is not surprising because top-down and Earley strategies can be described using SD–2SA. It was noted in (de la Clergerie and Alonso Pardo, 1998) that modes are often not explicitly needed. This is the case for Restricted LPDA: write and erase modes corresponds to call and

$$\frac{[A,h \mid B,i,\otimes,C,j,\gamma \mid D,p,\eta,E,q]}{[-,- \mid C,j,\models,F,k,- \mid -,-,-,-,-]} \quad C[\circ\circ] \overset{a}{\mapsto} C[\circ\circ]\ F[\,]$$

$$\frac{[A,h \mid B,i,\otimes,C,j,\gamma \mid -,-,-,-,-]}{[A,h \mid C,j,\rightarrow,F,k,\gamma \mid -,-,-,-,-]} \quad C[\circ\circ] \overset{a}{\mapsto} C[\circ\circ]\ F[\circ\circ]$$

$$\frac{[A,h \mid B,i,\otimes,C,j,\gamma \mid -,-,-,-,-]}{[C,j \mid C,j,\nearrow,F,k,\gamma' \mid -,-,-,-,-]} \quad C[\circ\circ] \overset{a}{\mapsto} C[\circ\circ]\ F[\circ\circ\gamma']$$

$$\frac{\begin{array}{c}[A,h \mid B,i,\otimes_1,C,j,\gamma \mid -,-,-,-,-]\\ [M,m \mid N,t,\otimes_2,A,h,\gamma' \mid -,-,-,-,-]\end{array}}{[M,m \mid C,j,\searrow,F,k,\gamma' \mid -,-,-,-,-]} \quad C[\circ\circ\gamma] \overset{a}{\mapsto} C[\circ\circ\gamma]\ F[\circ\circ]$$

$$\frac{[A,h \mid B,i,\otimes,C,j,\gamma \mid D,p,\eta,E,q]}{[A,h \mid B,i,\otimes,F,k,\gamma \mid D,p,\eta,E,q]} \quad C[\circ\circ] \overset{a}{\mapsto} F[\circ\circ]$$

$$\frac{\begin{array}{c}[-,- \mid B,i,\models,C,j,- \mid -,-,-,-,-]\\ [M,m \mid N,t,\otimes,B,i,\gamma \mid -,-,-,-,-]\end{array}}{[M,m \mid N,t,\otimes,F,k,\gamma \mid -,-,-,-,-]} \quad B[\circ\circ]\ C[\,] \overset{a}{\mapsto} F[\circ\circ]$$

$$\frac{\begin{array}{c}[A,h \mid B,i,\rightarrow,C,j,- \mid D,p,\eta,E,q]\\ [A,h \mid M,m,\otimes,B,i,\eta \mid -,-,-,-,-]\end{array}}{[A,h \mid M,m,\rightarrow,F,k,- \mid D,p,\eta,E,q]} \quad B[\circ\circ]\ C[\,] \overset{a}{\mapsto} F[\,]$$

$$\frac{\begin{array}{c}[A,h \mid B,i,\searrow,C,j,- \mid D,p,\eta,E,q]\\ [A,h \mid N,t,\otimes_1,M,m,\eta \mid -,-,-,-,-]\\ [M,m \mid O,u,\otimes_2,B,i,\gamma \mid -,-,-,-,-]\end{array}}{[M,m \mid O,u,\otimes_2 F,k,- \mid B,i,\gamma,C,j]} \quad B[\circ\circ]\ C[\,] \overset{a}{\mapsto} F[\,]$$

$$\frac{\begin{array}{c}[B,i \mid B,i,\nearrow,C,j,- \mid D,p,\eta,E,q]\\ [M,m \mid ,N,t,\otimes,B,i,\gamma \mid -,-,-,-]\\ [M,m \mid D,p,\searrow,E,q,\gamma'' \mid O,u,\eta',P,v]\end{array}}{[M,m \mid N,t,\otimes,F,k,- \mid O,u,\eta',P,v]} \quad B[\circ\circ]\ C[\,] \overset{a}{\mapsto} F[\,]$$

Table 11: Combining rules for $*$-TD strategies

return phases, which can be distinguished by the form of the non-terminals. A session corresponds to the evolution of the index stack through a spine. With respect to the correspondence between items for $*$-Earley strategies and items for SD–2SA:

- A call item

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

corresponds to a SD–2SA context-free item

$$\langle A,h\rangle\langle B,i\rangle \otimes \langle C,j,\gamma\rangle \mathbf{w}$$

where $\otimes$ indicates the operation performed to have γ as the top of the index stack.

- A return item

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

corresponds to a escaped context-free item

$$\langle A, h\rangle\langle B, i\rangle \otimes [\langle D, p, \gamma\rangle\langle E, q\rangle]\langle C, j, \gamma\rangle\mathbf{e}$$

where $\otimes$ indicates the operation performed during the call phase to have γ as the top of the index stack.

- A special point item

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

corresponds to a context-free item

$$\langle B, i\rangle\langle B, i\rangle{\models}\langle C, j, \models^0\rangle m$$

where o indicates the mode of the automaton when the current session was started, $m = \mathbf{w}$ if the item represents the creation of a new spine and $m = \mathbf{e}$ if the item represents the termination of a spine.

A similar correspondence exists between items for $*$-TD strategies and SD–2SA items:

- A call item

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

corresponds to a SD–2SA context-free item

$$\langle A, h\rangle\langle B, i\rangle \otimes \langle C, j, \gamma\rangle\mathbf{w}$$

- A return item

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

corresponds to a escaped context-free item

$$\langle A, h\rangle\langle B, i\rangle \otimes [\langle D, p, \eta\rangle\langle E, q\rangle]\langle C, j, \bot\rangle\mathbf{e}$$

- A special point item

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

corresponds to a context-free item

$$\langle B, i\rangle\langle B, i\rangle{\models}\langle C, j, \models^0\rangle m$$

as the items for $*$-Earley strategies.

The correspondence between combination rules for $*$-Earley and $*$-TD strategies in Restricted LPDA and the rules combining items in SD–2SA is shown in table 12.

6.2 BU–2SA

Bottom-up 2-Stack Automata (de la Clergerie et al., 1998) are a projection of SD–2SA requiring the emptiness of the auxiliary stack during computations in mode write. Only two marks are needed: $\models$ indicates a new session and $\triangleright$ indicates a push onto the master stack. With respect to the tabulation technique, this type of automata also considers *context-free derivations*, corresponding to call configurations and represented by context-free items of the form

$$\langle B, i\rangle \otimes \langle C, j, \models^o\rangle m$$

where $\otimes \in \{\triangleright, \models\}$, $m \in \{\mathbf{w}, \mathbf{e}\}$ and $o \in \{\mathbf{w}, \mathbf{e}\}$ is the mode of the automaton when the current session was started, and *escaped context-free derivations*, corresponding to return derivations and represented by items of the form

$$\langle B, i\rangle \otimes [\langle D, p\rangle\langle E, q\rangle]\langle C, j, \gamma\rangle\mathbf{e}$$

The tabulation technique proposed for $*$-BU strategies is similar to the proposal for BU–2SA but for the absence of modes and marks. Modes are not needed because the call and return phases are differentiated by the form of the involved non-terminal. The mark $\triangleright$ is actually redundant in BU–2SA, as every push in write mode involves a mark of this type, and is only considered to provide a true projection from SD–2SA. The mark $\models$ is not necessary because modes are not considered and sessions are implicitly represented as the propagation of the index stack through a spine. Table 13 shows the correspondence between combination rules for $*$-BU strategies and rules combining items in BU–2SA.

6.3 LIA

Linear Indexed Automata (Nederhof, 1998b) are an extension of push-down automata in which stack symbols are replaced by elements in $V_N[V_I^*]$, the set of transitions having the form

1. $C[\circ\circ] \overset{a}{\mapsto} C[\circ\circ]\ F[\,]$
2. $C[\circ\circ\gamma] \overset{a}{\mapsto} F[\circ\circ\gamma']$
3. $B[\circ\circ\gamma]\ C[\,] \overset{a}{\mapsto} F[\circ\circ\gamma']$
4. $B[\,]\ C[\circ\circ\gamma] \overset{a}{\mapsto} F[\circ\circ\gamma']$

Restricted LPDA	**SD–2SA**
$C[\circ\circ] \overset{a}{\mapsto} F[\circ\circ]$	**[SWAP]** without mode change
$C[\,] \overset{a}{\mapsto} F[\,]$	**[SWAP]** with mode change
$C[\circ\circ] \overset{a}{\mapsto} C[\circ\circ]\ F[\,]$	[$\models$–**WRITE**]
$B[\circ\circ]\ C[\,] \overset{a}{\mapsto} F[\circ\circ]$	[$\models$–**ERASE**]
$C[\circ\circ] \overset{a}{\mapsto} C[\circ\circ]\ F[\circ\circ]$	[$\rightarrow$–**WRITE**]
$C[\circ\circ\gamma] \overset{a}{\mapsto} C[\circ\circ\gamma]\ F[\circ\circ]$	[$\searrow$–**WRITE**]
$C[\circ\circ] \overset{a}{\mapsto} C[\circ\circ]\ F[\circ\circ\gamma']$	[$\nearrow$–**WRITE**]
$B[\circ\circ_1]\ C[\circ\circ_2] \overset{a}{\mapsto} F[\circ\circ_2]$	[$\rightarrow$–**ERASE**]
$B[\circ\circ_1]\ C[\circ\circ_2\gamma'] \overset{a}{\mapsto} F[\circ\circ_2]$	[$\searrow$–**ERASE**]
$B[\circ\circ_1\gamma]\ C[\circ\circ_2] \overset{a}{\mapsto} F[\circ\circ_2\gamma]$	[$\nearrow$–**ERASE**]

Table 12: Correspondence between Restricted LPDA and SD–2SA

Restricted LPDA	**BU–2SA**
$C[\circ\circ] \overset{a}{\mapsto} F[\circ\circ]$	[**SWAP1**] [**SWAP2**]
$C[\circ\circ] \overset{a}{\mapsto} C[\circ\circ]\ F[\,]$	[$\models$–**WRITE**] [$\triangleright$–**WRITE**]
$B[\circ\circ]\ C[\,] \overset{a}{\mapsto} F[\circ\circ]$	[$\models$–**ERASE**]
$B[\,]\ C[\circ\circ] \overset{a}{\mapsto} F[\circ\circ]$	[$\rightarrow$–**ERASE**]
$B[\,]\ C[\circ\circ] \overset{a}{\mapsto} F[\circ\circ\gamma']$	[$\searrow$–**ERASE**]
$B[\,]\ C[\circ\circ\gamma] \overset{a}{\mapsto} F[\circ\circ]$	[$\nearrow$–**ERASE**]

Table 13: Correspondence between Restricted LPDA and BU–2SA

with either γ or γ' (or both) equal to ϵ. The tabulation technique for this class of automata consider items

$$((B, C, j, i), (\gamma, D, E, p, q))$$

which are identical to items used for tabulation of $*$-BU strategies. In fact, there is an implicit distinction in (Nederhof, 1998b) between items with the second tuple equal to $(-, -, -, -, -)$ and items having that tuple instantiated. The former correspond to call items in $*$-BU strategies, the latter correspond to return items.

Transitions of type 1 and 4 are the same in both classes of automata (see table 4). Transitions of type 2 and 3 are present in $*$-BU strategies in a restrictive form mandating γ and γ' be equal to ϵ. The recognition power of both class of automata being the same, linear indexed automata can be used to implement strategies that are not possible to implement using the transitions for $*$-BU strategies. As an example, shift-reduce parsing algorithms, such as LR (Nederhof, 1998a), are not definable without transitions of type 4 (and therefore they can not be implemented in BU–2SA). By including the following combination rules, we obtain a tabulation framework for $*$-BU strategies in Restricted LPDA which is identical to that of LIA:

$$\frac{[B,i,C,j,\gamma \mid D,p,E,q]}{[B,i,F,k,\gamma' \mid B,i,C,j]} \quad C[\circ\circ] \overset{a}{\mapsto} F[\circ\circ\gamma']$$

$$\frac{\begin{array}{c}[B,i,C,j,\gamma \mid D,p,E,q] \\ [D,p,E,q,\gamma' \mid O,u,P,v]\end{array}}{[C,j,F,k,\gamma' \mid O,u,P,v]} \quad C[\circ\circ\gamma] \overset{a}{\mapsto} F[\circ\circ]$$

$$\frac{\begin{array}{c}[B,i,C,j,- \mid -,-,-,-] \\ [M,m,B,i,\gamma \mid D,p,E,q]\end{array}}{[M,m,F,k,\gamma \mid B,i,C,j]} \quad B[\circ\circ]\ C[\,] \overset{a}{\mapsto} F[\circ\circ\gamma']$$

$$\frac{\begin{array}{c}[B,i,C,j,- \mid -,-,-,-] \\ [M,m,B,i,\gamma \mid D,p,E,q] \\ [D,p,E,q,\gamma' \mid O,u,P,v]\end{array}}{[M,m,F,k,\gamma' \mid O,u,P,v]} \quad B[\circ\circ\gamma]\ C[\,] \overset{a}{\mapsto} F[\circ\circ]$$

6.4 A dual version of LIA

Linear indexed automata recognize the index stacks in a bottom-up manner. Nederhof (1998b) conjectures the existence of a dual stack automata for a top-down recognition of the index stacks, with the following set of transitions:

1. $B[\circ\circ]\ C[\,] \overset{a}{\mapsto} F[\circ\circ]$
2. $C[\circ\circ\gamma] \overset{a}{\mapsto} F[\circ\circ\gamma']$
3. $B[\circ\circ\gamma] \overset{a}{\mapsto} C[\circ\circ\gamma']\ F[\,]$
4. $B[\circ\circ\gamma] \overset{a}{\mapsto} C[\,]\ F[\circ\circ\gamma']$

where either γ or γ' (or both) equal to ϵ. Designing a tabulation technique for this class of automata is difficult because a chain of transitions of type 4 can produce a gap of unbounded size in the stack during the transmission of the dependent stack. To avoid this problem, we propose a slightly different set of transitions:

1. $B[\circ\circ]\ C[\,] \overset{a}{\mapsto} F[\circ\circ]$
2. $C[\circ\circ] \overset{a}{\mapsto} F[\circ\circ]$
3. $C[\circ\circ] \overset{a}{\mapsto} C[\circ\circ]\ F[\,]$
4. $C[\circ\circ\gamma] \overset{a}{\mapsto} C[\circ\circ\gamma]\ F[\circ\circ\gamma']$
5. $B[\circ\circ]\ C[\,] \overset{a}{\mapsto} F[\,]$

where transitions of type 4 transmit the index stack without creating a gap. The new transitions of type 5 are the alternative to those transitions of type 1 expecting an empty stack to be transmitted from B to F. This new class of automata can be tabulated using the technique developed for $*$-TD strategies.

7 Conclusion

We have studied the tabulation of automata for tree adjoining languages. We have taken LPDA and LIG as a starting point, and, for each family of parsing strategies, we have applied the following steps:

1. Description of the parsing strategies for LIG in LPDA.
2. Identification of the types of transition involved.
3. Design of a tabulation technique based on those types of transitions.

The resulting tabulation techniques have been compared with previous approaches. As a result we have found the technique for $*$-BU strategies to be identical to the technique designed for BU–2SA and being a subset of those designed for LIA (but with a possible extension to be equivalent to LIA). Techniques developed for $*$-Earley and $*$-TD strategies together allow us to obtain a better understanding of the complex tabulation technique available for SD–2SA. There is no tabulation technique for pure top-down automata for tree adjoining languages currently available but techniques developed for $*$-TD strategies provide the basis for a new model of automata, a dual version of LIA, which could also serve as a basis for the design of a top-down projection of SD–2SA.

8 Acknowledgments

This work has been partially supported by FEDER of European Union (1FD97-0047-C04-02) and Xunta de Galicia (XUGA20402B97).

References

Alfred V. Aho. 1968. Indexed grammars — an extension of context-free grammars. *Journal of the Association for Computer Machinery*, 15(4):647–671, October.

Tilman Becker. 1994. A new automaton model for TAGs: 2-SA. *Computational Intelligence*, 10(4):422–430.

Pierre Boullier. 1995. Yet another $\mathcal{O}(n^6)$ recognition algorithm for mildly context-sensitive languages. In *Proc. of the Fourth International Workshop on Parsing Technologies*, pages 34–47. Extended version as INRIA Rapport de Recherche 2730.

Eric de la Clergerie and Miguel A. Alonso Pardo. 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, pages 1333–1339, Montreal, Quebec, Canada, August. ACL.

Eric de la Clergerie and Bernard Lang. 1994. LPDA: Another look at tabulation in logic programming. In Van Hentenryck, editor, *Proc. of the 11th International Conference on Logic Programming (ICLP'94)*, pages 470–486. MIT Press, June.

Eric de la Clergerie, Miguel A. Alonso Pardo, and David Cabrero Souto. 1998. A tabular interpretation of bottom-up automata for TAG. In *Proc. of Fourth International Workshop on Tree-Adjoining Grammars and Related Frameworks (TAG+4)*, pages 42–45, Philadelphia, PA, USA, August.

Gerald Gazdar. 1987. Applicability of indexed grammars to natural languages. In U. Reyle and C. Rohrer, editors, *Natural Language Parsing and Linguistic Theories*, pages 69–94. D. Reidel Publishing Company.

Andrew Haas. 1989. A parsing algorithm for unification grammar. *Computational Linguistics*, 15(4):219–232.

Aravind K. Joshi and Yves Schabes. 1997. Tree-adjoining grammars. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages. Vol 3: Beyond Words*, chapter 2, pages 69–123. Springer-Verlag, Berlin/Heidelberg/New York.

Bernard Lang. 1988. Complete evaluation of Horn Clauses, an automata theoretic approach. Rapport de Recherche 913, INRIA, Rocquencourt, France, November.

Mark-Jan Nederhof. 1997. Solving the correct-prefix property for TAGs. In T. Becker and H.-V. Krieger, editors, *Proc. of the Fifth Meeting on Mathematics of Language*, pages 124–130, Schloss Dagstuhl, Saarbruecken, Germany, August.

Mark-Jan Nederhof. 1998a. An alternative LR algorithm for TAGs. 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, pages 946–952, Montreal, Quebec, Canada, August. ACL.

Mark-Jan Nederhof. 1998b. Linear indexed automata and tabulation of TAG parsing. In *Proc. of First Workshop on Tabulation in Parsing and Deduction (TAPD'98)*, pages 1–9, Paris, France, April.

Fernando C. N. Pereira and David H. D. Warren. 1980. Definite Clause Grammars for language analysis — a survey of the formalism and a comparison with Augmented Transition Networks. *Artificial Intelligence*, 13:231–278.

Fernando C. N. Pereira and David H. D. Warren. 1983. Parsing as deduction. In *Proc. of the 21st Annual Meeting of the Association for Computational Linguistics*, pages 137–144. ACL, June.

C. Pollard. 1984. *Generalized Phrase Structure Grammars, Head Grammars and Natural Language*. Ph.D. thesis, Stanford University.

Yves Schabes and K. Vijay-Shanker. 1990. Deterministic left to right parsing of tree adjoining languages. In *Proc. of 28th Annual Meeting of the Association for Computational Linguistics*, pages 276–283, Oittsburgh, Pennsylvania, USA, June. ACL.

Stuart M. Shieber, Yves Schabes, and Fernando C. N. Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24(1&2):3–36, July-August.

M. Steedman. 1986. Combinators and grammars. In R. Oehrle, E. Bach, and D Wheeler, editors, *Categorial Grammars and Natural Language Structures*, pages 417–442. Foris, Dordrecht.

K. Vijay-Shanker and David J. Weir. 1993. Parsing some constrained grammar formalisms. *Computational Linguistics*, 19(4):591–636.

K. Vijay-Shanker and David J. Weir. 1994. The equivalence of four extensions of context-free grammars. *Mathematical Systems Theory*, 27:511–545.

K. Vijay-Shanker. 1988. *A Study of Tree Adjoining Grammars*. Ph.D. thesis, University of Pennsylvania, January. Available as Technical Report MS-CIS-88-03 LINC LAB 95 of the Department of Computer and Information Science, University of Pennsylvania.