A tabular interpretation of a class of 2-Stack Automata

Eric Villemonte de la Clergerie
INRIA - Rocquencourt - B.P. 105
78153 Le Chesnay Cedex, FRANCE
Eric.De_La_Clergerie@inria.fr

Miguel Alonso Pardo
Universidad de La Coruña
Campus de Elviña s/n
15071 La Coruña, SPAIN
alonso@dc.fi.udc.es

Abstract

The paper presents a tabular interpretation for a kind of 2-Stack Automata. These automata may be used to describe various parsing strategies, ranging from purely top-down to purely bottom-up, for LIGs and TAGs. The tabular interpretation ensures, for all strategies, a time complexity in $O(n^6)$ and space complexity in $O(n^5)$ where n is the length of the input string.

Introduction

2-Stack automata [2SA] have been identified as possible operational devices to describe parsing strategies for Linear Indexed Grammars [LIG] or Tree Adjoining Grammars [TAG] (mirroring the traditional use of Push-Down Automata [PDA] for Context-Free Grammars [CFG]). Different variants of 2SA (or not so distant Embedded Push-Down Automata) have been proposed, some to describe top-down strategies (Vijay-Shanker, 1988; Becker, 1994), some to describe bottom-up strategies (Rambow, 1994; Nederhof, 1998; Alonso Pardo et al., 1997), but none (that we know) that are able to describe both kinds of strategies.

The same dichotomy also exists in the different tabular algorithms that has been proposed for specific parsing strategies with complexity ranging from $O(n^6)$ for bottom-up strategies to $O(n^9)$ for prefix-valid top-down strategies (with the exception of a $O(n^6)$ tabular interpretation of a prefix-valid hybrid strategy (Nederhof, 1997)). It must also be noted that the different tabular algorithms may be difficult to understand and it is often unclear to know if the algorithms still hold for different strategies.

This paper overcomes these problems by (a) introducing *strongly-driven 2SA* [SD-2SA] that may be used to describe parsing strategies for TAGs and LIGs, ranging from purely top-down to purely bottom-up, and (b) presenting a tabular interpretation of these automata in time complexity $O(n^6)$ and space complexity $O(n^5)$.

The tabular interpretation follows the principles of Dynamic Programming: the derivations are broken into elementary sub-derivations that may (a) be combined in different contexts to retrieve all possible derivations and (b) be represented in a compact way by *items*, allowing tabulation.

The strongly-driven 2SA are introduced and motivated in Section 1. We illustrate in Sections 2 and 3 their power by describing several parsing strategies for LIGs and TAGs. Items are presented in Section 4. Section 5 lists the rules to combine items and transitions and establishes correctness theorems.

1 Strongly-driven 2-Stack Automata

2SA are natural extensions of Push-Down Automata working on a pair of stacks. However, it is known that unrestricted 2SA have the power of a Turing Machine. The remedy is to consider asymmetric stacks, one being the Master Stack **MS** where most of the work is done and the other being the Auxiliary Stack **AS** mainly used for restricted "bookkeeping".

The following remarks are intended to give an idea of the restrictions we want to enforce. The first ones are rather standard and may be found under different forms in the literature. The last one justifies the qualification of "strongly-driven" for our automata.

[Session] **AS** should actually be seen as a stack of *session stacks*, each one being associated to a *session*. Only the topmost session stack may be consulted or modified. This idea is closely related to the notion of *Embedded Push-Down Automata* (Rambow, 1994, 96–102).

[Linearity] A session starts in mode write **w** and switches at some point in mode erase **e**. In mode **w** (resp. **e**), no element can be popped from (resp. pushed to) the master stack **MS**. Switching back from **e** to **w** is not allowed. This requirement is related to linearity because it means that a same session stack is never used twice by "descendants" of an element in **MS**.

[Soft Session Exit] 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.

[Driving] Each pushing on **MS** done in write mode leaves some mark in **MS** about the action that

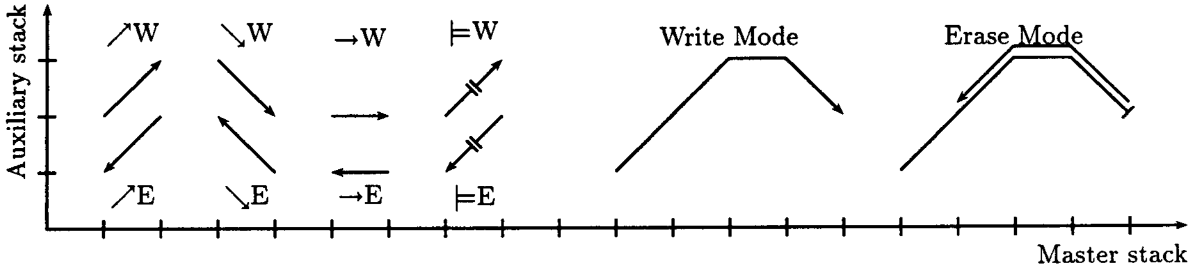

Figure 1: Representation of transitions and derivations

took place on the session stack. The popping of this mark (in erase mode) will guide which action should take place on the session stack. In other words, we want the erasing actions to faithfully retrace the writing actions.

Formally, a SD-2SA $\mathcal{A}$ is specified by a tuple $(\Sigma, \mathcal{M}, \mathcal{X}, \$_0, \$_f, \Theta)$ where Σ denotes the finite set of terminals, $\mathcal{M}$ the finite set of master stack elements and $\mathcal{X}$ the finite set of auxiliary stack elements. The **init** symbol $\$_0$ and **final** symbol $\$_f$ are distinguished elements of $\mathcal{M}$. Θ is a finite set of transitions.

The master stack **MS** is a word in $(\mathcal{D}\mathcal{M})^\star$ where $\mathcal{D}$ denotes the set $\{\nearrow, \searrow, \rightarrow, \models\}$ of **action marks** used to remember which action (w.r.t. the auxiliary stack **AS**) takes place when pushing the next master stack element. The empty master stack is noted ϵ and a non-empty master stack $\delta_1 A_1 \ldots \delta_n A_n$ where A_n denotes the topmost element.

The meaning of the action marks is:

- $\nearrow$ Pushing of an element on **AS**.
- $\searrow$ Popping of the topmost element of **AS**.
- $\rightarrow$ No action on **AS**.
- $\models$ Creation of a new session (with a new empty session stack on **AS**).

The auxiliary stack **AS** is a word of $(\mathcal{K}\mathcal{X}^\star)^\star$ where $\mathcal{K} = \{\models^{\mathbf{w}}, \models^{\mathbf{e}}\}$ is a set of two elements used to delimit session stacks in **AS**. Delimiter $\models^{\mathbf{w}}$ (resp. $\models^{\mathbf{e}}$) is used to start a new session from a session which is in writing (resp. erasing) mode. The empty auxiliary stack is noted ϵ.

Given some input string $x_1 \ldots x_f \in \Sigma^\star$, a configuration of $\mathcal{A}$ is a tuple (m, u, Ξ, ξ) where $m \in \{\mathbf{w}, \mathbf{e}\}$ denotes a mode (writing or erasing), u a string position in $[0, f]$, Ξ the master stack and ξ the auxiliary stack. Modes are ordered by $\mathbf{w} \prec \mathbf{e}$ to capture the fact that no switching from $\mathbf{e}$ to $\mathbf{w}$ is possible. The initial configuration of $\mathcal{A}$ is $(\mathbf{w}, 0, \models\$_0, \models^{\mathbf{w}})$ and the final one $(\mathbf{e}, f, \models\$_f, \models^{\mathbf{w}})$.

A transition is given as a pair $(p, \Xi, \xi) \overset{z}{\longmapsto} (q, \Theta, \theta)$ where p, q are modes (or, with some abuse, variables ranging over modes), z in $\Sigma^\star$, Ξ and Θ suffixes of master stacks in $\mathcal{M}(\mathcal{D}\mathcal{M})^\star$, and ξ, θ suffixes of auxiliary stacks in $\mathcal{X}^\star(\mathcal{K}\mathcal{X}^\star)^\star = (\mathcal{X} \cup \mathcal{K})^\star$. Such a transition applies on any configuration $(p, u, \Psi\Xi, \psi\xi)$ such that $x_{u+1} \ldots x_v = z$ and returns $(q, v, \Psi\Theta, \psi\theta)$.

We restrict the kind of allowed transitions:

SWAP $(p, A, \xi) \overset{z}{\longmapsto} (q, B, \xi)$ with $p \preceq q$ and either $\xi \in \mathcal{K}$ ("session bottom check") or $\xi = \epsilon$ ("no **AS** consultation") .

$\nearrow$**-WRITE** $(\mathbf{w}, A, \epsilon) \overset{z}{\longmapsto} (\mathbf{w}, A{\nearrow}B, b)$

$\nearrow$**-ERASE** $(\mathbf{e}, A{\nearrow}B, a) \overset{z}{\longmapsto} (\mathbf{e}, D, \epsilon)$

$\rightarrow$**-WRITE** $(\mathbf{w}, A, \epsilon) \overset{z}{\longmapsto} (\mathbf{w}, A{\rightarrow}B, \epsilon)$

$\rightarrow$**-ERASE** $(\mathbf{e}, A{\rightarrow}B, \epsilon) \overset{z}{\longmapsto} (\mathbf{e}, C, \epsilon)$

$\models$**-WRITE** $(m, A, \epsilon) \overset{z}{\longmapsto} (\mathbf{w}, A{\models}B, \models^m)$

$\models$**-ERASE** $(\mathbf{e}, A{\models}B, \models^m) \overset{z}{\longmapsto} (m, C, \epsilon)$

$\searrow$**-WRITE** $(\mathbf{w}, A, a) \overset{z}{\longmapsto} (\mathbf{w}, A{\searrow}B, \epsilon)$

$\searrow$**-ERASE** $(\mathbf{e}, A{\searrow}B, \epsilon) \overset{z}{\longmapsto} (\mathbf{e}, C, c)$

Figure 1 graphically outlines the different kinds of transitions using a 2D representation where the X-axis (Y-axis) is related to the master (resp. auxiliary) stack. Figure 1 also shows the two forms of derivations we encounter (during a same session).

2 Using 2SA to parse LIGs

Indexed Grammars (Aho, 1968) are an extension of Context-free Grammars in which a stack of indices is associated with each non-terminal symbol. Linear Indexed Grammars (Gazdar, 1987) are a restricted form of Indexed Grammars in which the index stack of at most one body non-terminal (the *child*) is related with the stack of the head non-terminal (the *father*). The other stacks of the production must have a bounded stack size.

Formally, a LIG G is a 5-tuple (V_T, V_N, S, V_I, P) where V_T is a finite set of terminals, V_N is a finite set of non-terminals, $S \in V_N$ is the start symbol, V_I is a finite set of indices 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:

[Terminal] $A_{k,0}[\,] \rightarrow a_k$ where $a_k \in V_T \cup \{\epsilon\}$,

[POP] $A_{k,0}[\circ\circ] \rightarrow A_{k,1}[\,] \ldots A_{k,d}[\circ\circ\gamma] \ldots A_{k,n_k}[\,]$

[PUSH] $A_{k,0}[\circ\circ\gamma] \rightarrow A_{k,1}[\,] \ldots A_{k,d}[\circ\circ] \ldots A_{k,n_k}[\,]$

[HOR] $A_{k,0}[\circ\circ] \rightarrow A_{k,1}[\,] \ldots A_{k,d}[\circ\circ] \ldots A_{k,n_k}[\,]$

To each production k of type PUSH, POP or HOR, we associate a characteristic tuple $t(k) = (d, \delta, \alpha, \beta)$ where d is the position of the child and the other arguments given by the following table:

Type	δ	α	β
PUSH	$\nearrow$	ϵ	γ
POP	$\searrow$	γ	ϵ
HOR	$\rightarrow$	ϵ	ϵ

We introduce symbols $\nabla_{k,i}$ as a shortcut for dotted productions $[A_{k,0} \rightarrow A_{k,1} \ldots A_{k,i} \bullet A_{k,i+1} \ldots A_{k,n_k}]$.

In order to design a broad class of parsing strategies ranging from pure top-down to pure bottom-up, we parameterize the automaton to be presented by a call projection $\rightarrow$ from $\mathcal{V}$ to $\mathcal{V}^{call}$ and a return projection $\leftarrow$ from $\mathcal{V}$ to $\mathcal{V}^{ret}$ where $\mathcal{V} = V_N \cup V_I$ and $\mathcal{V}^{call}$ and $\mathcal{V}^{ret}$ are two sets of elements. We require $\mathcal{V}^{call} \cap \mathcal{V}^{ret} = \emptyset$ and $(\rightarrow, \leftarrow)$ to be invertible, i.e $\forall X, Y \in \mathcal{V},\ (\overrightarrow{X}, \overleftarrow{X}) = (\overrightarrow{Y}, \overleftarrow{Y}) \ \Rightarrow\ X = Y$

The projections extend to sequences by taking $\overrightarrow{X_1 \ldots X_n} = \overrightarrow{X_1} \ldots \overrightarrow{X_n}$ and $\overrightarrow{\epsilon} = \epsilon$ (similarly for $\leftarrow$).

Given a LIG G and a choice of projections, we define the 2SA $\mathcal{A}(G, \rightarrow, \leftarrow) = (V_T, \mathcal{M}, \mathcal{X}, \overrightarrow{S}, \overleftarrow{S}, \Theta)$ where $\mathcal{M} = \{\nabla_{k,i}\} \cup \overrightarrow{V_T} \cup \overleftarrow{V_T}$, $\mathcal{X} = \overrightarrow{V_I} \cup \overleftarrow{V_I}$, and whose transitions are built using the following rules.

- Call/Return of a non child

CALL : $(m, \nabla_{k,i}, \epsilon) \longmapsto (\mathbf{w}, \nabla_{k,i} {\models} \overrightarrow{A_{k,i+1}}, \models^m)$

RET : $(\mathbf{e}, \nabla_{k,i} {\models} \overleftarrow{A_{k,i+1}}, \models^m) \longmapsto (m, \nabla_{k,i+1}, \epsilon)$

- Call/Return of a child for $t(k) = (i+1, \delta, \alpha, \beta)$.

CALL(δ) : $(\mathbf{w}, \nabla_{k,i}, \overrightarrow{\alpha}) \longmapsto (\mathbf{w}, \nabla_{k,i} \delta \overrightarrow{A_{k,i+1}}, \overrightarrow{\beta})$

RET(δ) : $(\mathbf{e}, \nabla_{k,i} \delta \overleftarrow{A_{k,i+1}}, \overleftarrow{\beta}) \longmapsto (\mathbf{e}, \nabla_{k,i+1}, \overleftarrow{\alpha})$

- Production Selection
  SEL : $(\mathbf{w}, \overrightarrow{A_{k,0}}, \epsilon) \longmapsto (\mathbf{w}, \nabla_{k,0}, \epsilon)$
- Production Publishing
  PUB : $(\mathbf{e}, \nabla_{k,n_k}, \epsilon) \longmapsto (\mathbf{e}, \overleftarrow{A_{k,0}}, \epsilon)$
- Scanning (for terminal productions)
  SCAN : $(\mathbf{w}, \overrightarrow{A_{k,0}}, \models^m) \xmapsto{a_k} (\mathbf{e}, \overleftarrow{A_{k,0}}, \models^m)$

The reader may easily check that $\mathcal{A}(G, \rightarrow, \leftarrow)$ recognizes $L(G)$. The choice of the **call** and **return** elements for the **MS** ($\overrightarrow{A_{k,i}}$ and $\overleftarrow{A_{k,i}}$) and the **AS** ($\overrightarrow{\gamma}$ and $\overleftarrow{\gamma}$) defines a parsing strategy, by controlling how information flow between the phases of prediction and propagation. The following table lists the choices corresponding to the main parsing strategies (but others are definable).

Strategy	$\overrightarrow{A}$	$\overleftarrow{A}$	$\overrightarrow{\gamma}$	$\overleftarrow{\gamma}$
Top-Down	A	$\perp$	γ	$\perp$
Bottom-Up	$\perp$	A'	$\perp$	γ
Earley	A	A'	γ	γ'

It is also worth to note that the description of $\mathcal{A}(G, \rightarrow, \leftarrow)$ could be simplified. Indeed, for every configuration (m, u, Ξ, ξ) derivable with $\mathcal{A}(G, \rightarrow, \leftarrow)$, we can show that $\Xi = \models \nabla_{k_1,i_1} \delta_1 \ldots \nabla_{k_n,i_n} \delta_n X$, and that δ_l only depends on ∇_{k_l,i_l}. That means that we could use a master stack without action marks, these marks being implicitly given by the elements $\nabla_{k,i}$.

3 Using 2SA to parse TAGs

Tree Adjoining Grammars are a extension of CFG introduced by Joshi in (Joshi, 1987) that use trees instead of productions as primary representing structure. Formally, a TAG is a 5-tuple $\mathcal{G} = (V_N, V_T, S, I, A)$, where V_N is a finite set of non-terminal symbols, V_T a finite set of terminal symbols, S the axiom of the grammar, I a finite set of *initial trees* and A a finite set of *auxiliary trees*. $I \cup A$ is the set of *elementary trees*. Internal nodes are labeled by non-terminals and leaf nodes by terminals or ϵ, except for exactly one leaf per auxiliary tree (the *foot*) which is labeled by the same non-terminal used as label of its root node.

New trees are derived by *adjoining*: let be α a tree containing a node ν labeled by A and let be β an auxiliary tree whose root and foot nodes are also labeled by A. Then, the adjoining of β at the *adjunction node* ν is obtained by excising the subtree α_ν of α with root ν, attaching β to ν and attaching the excised subtree to the foot of β (See Fig. 2).

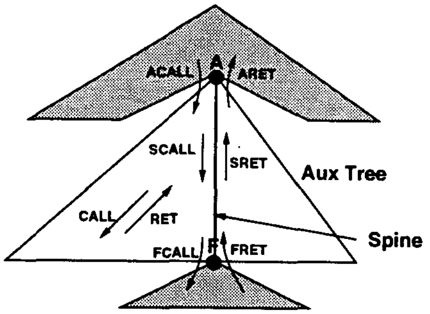

Figure 2: Traversal of an adjunction

An elementary tree α may be represented by a set $P(\alpha)$ of context free productions, each one being either of the form

- $\nu_{k,0} \rightarrow \nu_{k,1} \ldots \nu_{k,n_k}$, where $\nu_{k,0}$ denotes some non-leaf node k of α and $\nu_{k,i}$ the i^{th} son of k.

- $\nu_{k,0} \longrightarrow a_k$, where $\nu_{k,0}$ denotes some leaf node k of α with terminal label a_k.

As done for LIGs, we introduce symbols $\nabla_{k,i}$ to denote dotted productions and consider projections $\overrightarrow{\ }$ and $\overleftarrow{\ }$ to define the parameterized 2SA $\mathcal{A}(G, \overrightarrow{\ }, \overleftarrow{\ }) = (V_T, \mathcal{M}, \mathcal{M}, \overrightarrow{\nu_{0,0}}, \overleftarrow{\nu_{0,0}}, \Theta)$ where $\mathcal{M} = \{\nabla^{\alpha}_{k,i}\} \cup \{\overrightarrow{\nu^{\alpha}_{k,i}}\} \cup \{\overleftarrow{\nu^{\alpha}_{k,i}}\}$. The transitions are given by the following rules (and illustrated in Figure 2).

- Call / Return for a node not on a spine. The call starts a new session, exited at return.

CALL : $(m, \nabla_{k,i}, \epsilon) \longmapsto (\mathbf{w}, \nabla_{k,i}{\models}\overrightarrow{\nu_{k,i+1}}, \models^m)$
RET : $(\mathbf{e}, \nabla_{k,i}{\models}\overleftarrow{\nu_{k,i+1}}, \models^m) \longmapsto (m, \nabla_{k,i+1}, \epsilon)$

- Call / Return for a node $\nu_{k,i+1}$ on a spine. The adjunction stack is propagated un-modified along the spine.

SCALL : $(\mathbf{w}, \nabla_{k,i}, \epsilon) \longmapsto (\mathbf{w}, \nabla_{k,i}{\rightarrow}\overrightarrow{\nu_{k,i+1}}, \epsilon)$
SRET : $(\mathbf{e}, \nabla_{k,i}{\rightarrow}\overleftarrow{\nu_{k,i+1}}, \epsilon) \longmapsto (\mathbf{e}, \nabla_{k,i+1}, \epsilon)$

- Call / Return for an adjunction on node $\nu_{k,0}$. The computation is diverted to parse some acceptable auxiliary tree β (with root node r_β), and a continuation point is stored on the auxiliary stack.

ACALL : $(\mathbf{w}, \overrightarrow{\nu_{k,0}}, \epsilon) \longmapsto (\mathbf{w}, \overrightarrow{\nu_{k,0}}{\nearrow}\overrightarrow{r_\beta}, \nabla_{k,0})$
ARET : $(\mathbf{e}, \overrightarrow{\nu_{k,0}}{\nearrow}\overleftarrow{r_\beta}, \nabla_{k,n_k}) \longmapsto (\mathbf{e}, \overleftarrow{\nu_{k,0}}, \epsilon)$

- Call / Return for a foot node f_β. The continuation stored by the adjunction is used to parse the excised subtree.

FCALL : $(\mathbf{w}, \overrightarrow{f_\beta}, A) \longmapsto (\mathbf{w}, \overrightarrow{f_\beta}{\searrow}A, \epsilon)$
FRET : $(\mathbf{e}, \overrightarrow{f_\beta}{\searrow}A, \epsilon) \longmapsto (\mathbf{e}, \overleftarrow{f_\beta}, A)$

Note: These two transitions use a variable A over $\mathcal{M}$. This is a slight extension of 2SA that preserves correctness and complexity.

- Production Selection

SEL : $(\mathbf{w}, \overrightarrow{\nu_{k,0}}, \epsilon) \longmapsto (\mathbf{w}, \nabla_{k,0}, \epsilon)$

- Production Publishing

PUB : $(m, \nabla_{k,n_k}, \epsilon) \longmapsto (\mathbf{e}, \overleftarrow{\nu_{k,0}}, \epsilon)$

- Scanning

SCAN : $(\mathbf{w}, \overrightarrow{\nu_{k,0}}, \models^m) \xmapsto{a_k} (\mathbf{e}, \overleftarrow{\nu_{k,0}}, \models^m)$

Different parsing strategies can be obtained by choosing the **call** ($\overrightarrow{\nu_{k,i}}$) and **return** ($\overleftarrow{\nu_{k,i}}$) elements:

Strategy	$\overrightarrow{\nu}$	$\overleftarrow{\nu}$
prefix-valid Top-Down	ν	$\bot$
Bottom-Up	$\bot$	ν'
prefix-valid Earley	ν	ν'

Non prefix-valid variants of the top-down and Earley-like strategies can also be defined, by taking $\overrightarrow{r_\beta} = \bot$ and $\overleftarrow{r_\beta} = r_\beta$ for every root node r_β of an auxiliary tree β (the projections being unmodified on the other elements). In other words, we get a full prediction on the context-free backbone of G but no prediction on the adjunctions.

4 Items

We identify two kinds of elementary derivations, namely **Context-Free** [CF] and **escaped Context-Free** [xCF] derivations, respectively represented by CF and xCF items. An item keeps the pertinent information relative to a derivation, which allows to apply the sequence of transitions associated with the derivation in different contexts.

Before presenting these items, we introduce the following classification about derivations.

A derivation $(p, u, \Xi A, \xi) \overset{\star}{\underset{d_1}{\vdash}} (q, v, \Theta, \theta)$ is said **rightward** if no element of Ξ is accessed (even for consultation) during the derivation and if A is only consulted. Then ΞA is a prefix of Θ.

Similarly, a derivation $(p, u, \Xi, \xi) \overset{\star}{\underset{d}{\vdash}} (q, v, \Theta, \theta)$ is said **upward** if no element of ξ is accessed (even for consultation). Then ξ is a prefix of θ.

We also note $w[q/p]$ the prefix substitution of p by q for all words w, p, q on some vocabulary such that p is prefix of w.

4.1 Context-Free Derivations

A **Context-Free** [CF] derivation only depends on the topmost element A of the initial stack **MS**. That means that no element of the initial **AS** and no element of **MS** below element A is needed:

$$(o, u, \Xi A, \xi) \overset{\star}{\underset{d_1}{\vdash}} (\mathbf{w}, v, \Theta B, \theta) \overset{\star}{\underset{d_2}{\vdash}} (m, w, \Theta B \delta C, \xi c)$$

where

- d_1 and $d_1 d_2$ are both rightward and upward.
- d_2 is rightward.
- either ($\delta \neq \models$, $o = \mathbf{w}$, and $c \in \mathcal{X}$) or ($\delta = \models$, and $c = \models^o$).

For such a derivation, we have:

Proposition 4.1 *For all prefix stacks Ξ', ξ',*

$$\begin{aligned} (o, u, \Xi' A, \xi') & \overset{\star}{\underset{d_1}{\vdash}} (\mathbf{w}, v, \Theta' B, \theta') \\ & \overset{\star}{\underset{d_2}{\vdash}} (m, w, \Theta' B \delta C, \xi' c) \end{aligned}$$

where $\Theta' = \Theta[\Xi'/\Xi]$ and $\theta' = \theta[\xi'/\xi]$.

The proposition suggests representing the CF derivation by a CF item of the form

$$\boldsymbol{A}\boldsymbol{B}\delta\bar{\boldsymbol{C}}m$$

where $\boldsymbol{A} = \langle u, A \rangle$ and $\boldsymbol{B} = \langle v, B \rangle$ are **micro** configurations and $\bar{\boldsymbol{C}} = \langle w, C, c \rangle$ a **mini** configuration.

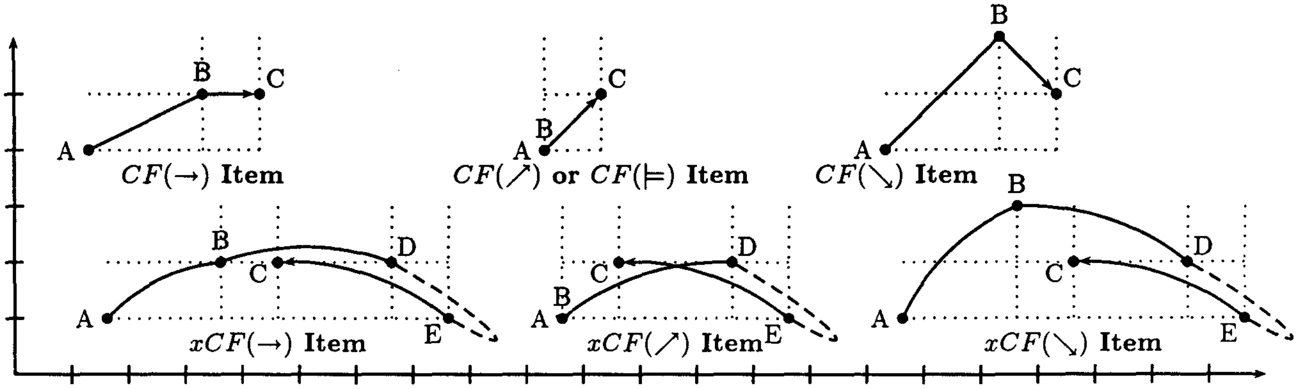

Figure 3: Items Shapes

4.2 Escaped Context-Free Derivations

An **escaped Context-Free** [xCF] derivation is almost a CF derivation, except for an *escape* subderivation that accesses deep elements of **AS**.

$$\begin{array}{rcl}(\mathbf{w},u,\Xi A,\xi) & \overset{\star}{\underset{d_1}{\vdash}} & (\mathbf{w},v,\Theta B,\theta)\\ & \overset{\star}{\underset{d_2}{\vdash}} & (\mathbf{w},s,\Phi D,\xi d)\\ & \overset{\star}{\underset{dx}{\vdash}} & (\mathbf{e},t,\Phi D{\searrow}E,\phi)\\ & \overset{\star}{\underset{d_3}{\vdash}} & (\mathbf{e},w,\Theta B\delta C,\phi c)\end{array}$$

where

- d_1 and $d_1 d_2$ are both rightward and upward.
- d_2 and dx are rightward.
- d_3 is upward.
- $\delta \neq \models$ and $d, c \in \mathcal{X}$.

Proposition 4.2 *For all prefix stacks Ξ' and ξ', stack ϕ', and rightward derivation*

$$(\mathbf{w},s,\Phi' D,\xi' d)\overset{\star}{\underset{dx'}{\vdash}}(\mathbf{e},t,\Phi' D{\searrow}E,\phi')$$

where $\Phi' = \Phi[\Xi'/\Xi]$, we have

$$\begin{array}{rcl}(\mathbf{w},u,\Xi' A,\xi') & \overset{\star}{\underset{d_1}{\vdash}} & (\mathbf{w},v,\Theta[\Xi'/\Xi]B,\theta[\xi'/\xi])\\ & \overset{\star}{\underset{d_2}{\vdash}} & (\mathbf{w},s,\Phi[\Xi'/\Xi]D,\xi' d)\\ & \overset{\star}{\underset{dx'}{\vdash}} & (\mathbf{e},t,\Phi[\Xi'/\Xi]D{\searrow}E,\phi')\\ & \overset{\star}{\underset{d_3}{\vdash}} & (\mathbf{e},w,\Theta[\Xi'/\Xi]B\delta C,\phi' c)\end{array}$$

The proposition suggests representing the xCF derivation by an xCF item of the form

$$\boldsymbol{A}\boldsymbol{B}\delta[\bar{\boldsymbol{D}}\boldsymbol{E}]\bar{\boldsymbol{C}}\mathbf{e}$$

where $\boldsymbol{A} = \langle u, A\rangle$, $\boldsymbol{B} = \langle v, B\rangle$, $\bar{\boldsymbol{D}} = \langle s, D, d\rangle$, $\boldsymbol{E} = \langle t, E\rangle$ and $\bar{\boldsymbol{C}} = \langle w, C, c\rangle$.

In order to homogenize notations, we also use the alternate notation $\boldsymbol{A}\boldsymbol{B}\delta[\diamond\diamond]\bar{\boldsymbol{C}}m$ to represent CF item $\boldsymbol{A}\boldsymbol{B}\delta\bar{\boldsymbol{C}}m$, introducing a dummy symbol $\diamond$.

The specific forms taken by CF and xCF items for the different actions δ are outlined in Figure 3.

5 Combining items and transitions

We provide the rules to combine items and transitions in order to retrieve all possible 2SA derivations.

These rules do not explicit the scanning constraints and suppose that the string z may be read between positions w and k of the input string. They use holes $\star$ to denote slots that not need be consulted. For any mini configuration $\bar{\boldsymbol{A}} = \langle u, A, a\rangle$, we note $\bar{\boldsymbol{A}}^{\circ} = \langle u, A\rangle$ its micro projection.

[$\rightarrow$–**WRITE**] $\tau = (\mathbf{w}, C, \epsilon) \overset{z}{\longmapsto} (\mathbf{w}, C{\rightarrow}F, \epsilon)$

$$\boldsymbol{A}{\star\star}[\diamond\diamond]\bar{\boldsymbol{C}}\mathbf{w} \overset{\tau}{\Longrightarrow} \boldsymbol{A}\bar{\boldsymbol{C}}^{\circ}{\rightarrow}[\diamond\diamond]\bar{\boldsymbol{F}}\mathbf{w} \qquad (1)$$

where $\bar{\boldsymbol{C}} = \langle w, C, c\rangle$, and $\bar{\boldsymbol{F}} = \langle k, F, c\rangle$.

[$\nearrow$–**WRITE**] $\tau = (\mathbf{w}, C, \epsilon) \overset{z}{\longmapsto} (\mathbf{w}, C{\nearrow}F, f)$

$$\boldsymbol{A}{\star\star}[\diamond\diamond]\bar{\boldsymbol{C}}\mathbf{w} \overset{\tau}{\Longrightarrow} \bar{\boldsymbol{C}}^{\circ}\bar{\boldsymbol{C}}^{\circ}{\nearrow}[\diamond\diamond]\bar{\boldsymbol{F}}\mathbf{w} \qquad (2)$$

where $\bar{\boldsymbol{C}} = \langle w, C, c\rangle$, and $\bar{\boldsymbol{F}} = \langle k, F, f\rangle$.

[$\models$–**WRITE**] $\tau = (m, C, \epsilon) \overset{z}{\longmapsto} (\mathbf{w}, C{\models}F, \models^{m})$

$$\boldsymbol{A}{\star\star}[\diamond\diamond]\bar{\boldsymbol{C}}m \overset{\tau}{\Longrightarrow} \bar{\boldsymbol{C}}^{\circ}\bar{\boldsymbol{C}}^{\circ}{\models}[\diamond\diamond]\bar{\boldsymbol{F}}\mathbf{w} \qquad (3)$$

where $\bar{\boldsymbol{C}} = \langle w, C, c\rangle$, and $\bar{\boldsymbol{F}} = \langle k, F, \models^{m}\rangle$.

[$\searrow$–**WRITE**] $\tau = (\mathbf{w}, C, c) \overset{z}{\longmapsto} (\mathbf{w}, C{\searrow}F, \epsilon)$

$$\left.\begin{array}{l}\bar{\boldsymbol{A}}^{\circ}{\star\star}[\diamond\diamond]\bar{\boldsymbol{C}}\mathbf{w}\\ \boldsymbol{M}{\star\star}[\diamond\diamond]\bar{\boldsymbol{A}}\mathbf{w}\end{array}\right\}\overset{\tau}{\Longrightarrow} \boldsymbol{M}\bar{\boldsymbol{C}}^{\circ}{\searrow}[\diamond\diamond]\bar{\boldsymbol{F}}\mathbf{w} \qquad (4)$$

where $\bar{\boldsymbol{C}} = \langle w, C, c\rangle$, $\bar{\boldsymbol{A}} = \langle u, A, a\rangle$, and $\bar{\boldsymbol{F}} = \langle k, F, a\rangle$.

[$\rightarrow$–**ERASE**] $\tau = (\mathbf{e}, B{\rightarrow}C, \epsilon) \overset{z}{\longmapsto} (\mathbf{e}, F, \epsilon)$

$$\left.\begin{array}{l}\bar{\boldsymbol{A}}^{\circ}\bar{\boldsymbol{B}}^{\circ}{\rightarrow}[\bar{\boldsymbol{D}}\boldsymbol{E}]\bar{\boldsymbol{C}}\mathbf{e}\\ \bar{\boldsymbol{A}}^{\circ}\boldsymbol{M}\lambda[\diamond\diamond]\bar{\boldsymbol{B}}\mathbf{w}\end{array}\right\}\overset{\tau}{\Longrightarrow} \bar{\boldsymbol{A}}^{\circ}\boldsymbol{M}\lambda[\bar{\boldsymbol{D}}\boldsymbol{E}]\bar{\boldsymbol{F}}\mathbf{e} \qquad (5)$$

where $\bar{\boldsymbol{C}} = \langle w, C, c\rangle$, $\bar{\boldsymbol{B}} = \langle v, B, b\rangle$, $\bar{\boldsymbol{F}} = \langle k, F, c\rangle$, and (when $\bar{\boldsymbol{D}} \neq \diamond$) $\bar{\boldsymbol{D}} = \langle s, D, b\rangle$.

[$\searrow$–**ERASE**] $\tau = (\mathbf{e}, B\searrow C, \epsilon) \overset{z}{\longmapsto} (\mathbf{e}, F, f)$

$$\left.\begin{array}{l} \bar{\boldsymbol{A}}^{\circ}\bar{\boldsymbol{B}}^{\circ}\searrow[\bar{\boldsymbol{D}}\star]\bar{\boldsymbol{C}}\mathbf{e}\\ \bar{\boldsymbol{A}}^{\circ}\star\lambda[\diamond\diamond]\bar{\boldsymbol{M}}\mathbf{w}\\ \bar{\boldsymbol{M}}^{\circ}\boldsymbol{O}\mu[\diamond\diamond]\bar{\boldsymbol{B}}\mathbf{w}\end{array}\right\}\overset{\tau}{\Longrightarrow}\bar{\boldsymbol{M}}^{\circ}\boldsymbol{O}\mu[\bar{\boldsymbol{B}}\bar{\boldsymbol{C}}^{\circ}]\bar{\boldsymbol{F}}\mathbf{e} \tag{6}$$

where $\bar{\boldsymbol{C}} = \langle w, C, c\rangle$, $\bar{\boldsymbol{B}} = \langle v, B, b\rangle$, $\bar{\boldsymbol{M}} = \langle l, M, m\rangle$, $\bar{\boldsymbol{F}} = \langle k, F, f\rangle$, and (when $\bar{\boldsymbol{D}} \neq \diamond$) $\bar{\boldsymbol{D}} = \langle \star, \star, m\rangle$.

[$\models$–**ERASE**] $\tau = (\mathbf{e}, B\models C, \models^{m}) \overset{z}{\longmapsto} (m, F, \epsilon)$

$$\left.\begin{array}{l} \bar{\boldsymbol{B}}^{\circ}\boldsymbol{B}\models[\diamond\diamond]\bar{\boldsymbol{C}}\mathbf{e}\\ \boldsymbol{M}\boldsymbol{N}\lambda[\bar{\boldsymbol{D}}\boldsymbol{E}]\bar{\boldsymbol{B}}m\end{array}\right\}\overset{\tau}{\Longrightarrow}\boldsymbol{M}\boldsymbol{N}\lambda[\bar{\boldsymbol{D}}\boldsymbol{E}]\bar{\boldsymbol{F}}m \tag{7}$$

where $\bar{\boldsymbol{C}} = \langle w, C, \models^{m}\rangle$, $\bar{\boldsymbol{B}} = \langle v, B, b\rangle$, and $\bar{\boldsymbol{F}} = \langle k, F, a\rangle$

[$\nearrow$–**ERASE**] $\tau = (\mathbf{e}, B\nearrow C, c) \overset{z}{\longmapsto} (\mathbf{e}, F, \epsilon)$

$$\left.\begin{array}{l} \bar{\boldsymbol{B}}^{\circ}\bar{\boldsymbol{B}}^{\circ}\nearrow[\diamond\diamond]\bar{\boldsymbol{C}}\mathbf{e}\\ \boldsymbol{M}\boldsymbol{N}\lambda[\diamond\diamond]\bar{\boldsymbol{B}}\mathbf{w}\end{array}\right\}\overset{\tau}{\Longrightarrow}\boldsymbol{M}\boldsymbol{N}\lambda[\diamond\diamond]\bar{\boldsymbol{F}}\mathbf{e} \tag{8}$$

where $\bar{\boldsymbol{C}} = \langle w, C, c\rangle$, $\bar{\boldsymbol{B}} = \langle v, B, b\rangle$, and $\bar{\boldsymbol{F}} = \langle k, F, b\rangle$

$$\left.\begin{array}{l} \bar{\boldsymbol{B}}^{\circ}\bar{\boldsymbol{B}}^{\circ}\nearrow[\bar{\boldsymbol{D}}\bar{\boldsymbol{E}}^{\circ}]\bar{\boldsymbol{C}}\mathbf{e}\\ \boldsymbol{M}\boldsymbol{N}\lambda[\diamond\diamond]\bar{\boldsymbol{B}}\mathbf{w}\\ \boldsymbol{M}\bar{\boldsymbol{D}}^{\circ}\searrow[\bar{\boldsymbol{O}}\boldsymbol{P}]\bar{\boldsymbol{E}}\mathbf{e}\end{array}\right\}\overset{\tau}{\Longrightarrow}\boldsymbol{M}\boldsymbol{N}\lambda[\bar{\boldsymbol{O}}\boldsymbol{P}]\bar{\boldsymbol{F}}\mathbf{e} \tag{9}$$

where $\bar{\boldsymbol{C}} = \langle w, C, c\rangle$, $\bar{\boldsymbol{B}} = \langle v, B, b\rangle$, $\bar{\boldsymbol{F}} = \langle k, F, b\rangle$, and (when $\bar{\boldsymbol{O}} \neq \diamond$) $\bar{\boldsymbol{O}} = \langle l, O, b\rangle$.

[**SWAP**] $\tau = (p, C, \xi) \overset{z}{\longmapsto} (q, F, \xi)$

$$\boldsymbol{A}\boldsymbol{B}\delta[\bar{\boldsymbol{D}}\boldsymbol{E}]\bar{\boldsymbol{C}}m \overset{\tau}{\Longrightarrow} \boldsymbol{A}\boldsymbol{B}\delta[\bar{\boldsymbol{D}}\boldsymbol{E}]\bar{\boldsymbol{F}}m \tag{10}$$

where $\bar{\boldsymbol{C}} = \langle w, C, c\rangle$, $\bar{\boldsymbol{F}} = \langle k, F, c\rangle$, and either $c = \xi = \models^{\circ}$ or $\xi = \epsilon$.

The best way to apprehend these rules is to visualize them graphically as done for the two most complex ones (Rules 6 and 9) in Figures 4 and 5.

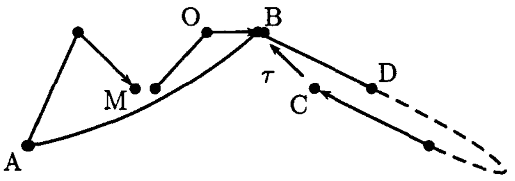

Figure 4: Application of Rule 6

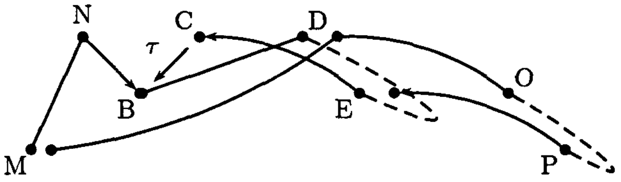

Figure 5: Application of Rule 9

5.1 Reducing the complexity

An analysis of the time complexity to apply each rule gives us polynomial complexities $O(n^{u})$ with $u \leq 6$ except for Rule 9 where $u = 8$. However, by adapting an idea from (Nederhof, 1997), we replace Rule 9 by the alternate and equivalent Rule 11.

$$\left.\begin{array}{l} \bar{\boldsymbol{B}}^{\circ}\star\nearrow[\bar{\boldsymbol{D}}\bar{\boldsymbol{E}}^{\circ}]\bar{\boldsymbol{C}}\mathbf{e}\\ \star\bar{\boldsymbol{D}}^{\circ}\searrow[\bar{\boldsymbol{O}}\boldsymbol{P}]\bar{\boldsymbol{E}}\mathbf{e}\\ \boldsymbol{M}\boldsymbol{N}\lambda[\diamond\diamond]\bar{\boldsymbol{B}}\mathbf{w}\\ \boldsymbol{M}\star\searrow[\bar{\boldsymbol{O}}\boldsymbol{P}]\star\mathbf{e}\end{array}\right\}\overset{\tau}{\Longrightarrow}\boldsymbol{M}\boldsymbol{N}\lambda[\bar{\boldsymbol{O}}\boldsymbol{P}]\bar{\boldsymbol{F}}\mathbf{e} \tag{11}$$

where $\bar{\boldsymbol{C}} = \langle w, C, c\rangle$, $\bar{\boldsymbol{B}} = \langle v, B, b\rangle$, $\bar{\boldsymbol{F}} = \langle k, F, b\rangle$, and (when $\bar{\boldsymbol{O}} \neq \diamond$) $\bar{\boldsymbol{O}} = \langle l, O, b\rangle$.

Rule 11 has same complexity than Rule 9, but may actually be split into two rules of lesser complexity $O(n^{6})$, introducing an intermediary pseudo-item $\boldsymbol{B}\boldsymbol{B}\nearrow[[\bar{\boldsymbol{O}}\boldsymbol{P}]]\bar{\boldsymbol{C}}\mathbf{e}$ (intuitively assimilable to a "deeply escaped" CF derivation).

Rule 12 collects these pseudo-items (independently from any transition) while Rule 13 combines them with items (given a $\nearrow$–ERASE transition τ).

$$\left.\begin{array}{l} \boldsymbol{B}\boldsymbol{B}\nearrow[\bar{\boldsymbol{D}}\bar{\boldsymbol{E}}^{\circ}]\bar{\boldsymbol{C}}\mathbf{e}\\ \star\bar{\boldsymbol{D}}^{\circ}\searrow[\bar{\boldsymbol{O}}\boldsymbol{P}]\bar{\boldsymbol{E}}\mathbf{e}\end{array}\right\}\Longrightarrow\boldsymbol{B}\boldsymbol{B}\nearrow[[\bar{\boldsymbol{O}}\boldsymbol{P}]]\bar{\boldsymbol{C}}\mathbf{e} \tag{12}$$

$$\left.\begin{array}{l} \bar{\boldsymbol{B}}^{\circ}\bar{\boldsymbol{B}}^{\circ}\nearrow[[\bar{\boldsymbol{O}}\boldsymbol{P}]]\bar{\boldsymbol{C}}\mathbf{e}\\ \boldsymbol{M}\boldsymbol{N}\lambda[\diamond\diamond]\bar{\boldsymbol{B}}\mathbf{w}\\ \boldsymbol{M}\star\searrow[\bar{\boldsymbol{O}}\boldsymbol{P}]\star\mathbf{e}\end{array}\right\}\overset{\tau}{\Longrightarrow}\boldsymbol{M}\boldsymbol{N}\lambda[\bar{\boldsymbol{O}}\boldsymbol{P}]\bar{\boldsymbol{F}}\mathbf{e} \tag{13}$$

where $\bar{\boldsymbol{C}} = \langle w, C, c\rangle$, $\bar{\boldsymbol{B}} = \langle v, B, b\rangle$, $\bar{\boldsymbol{F}} = \langle k, F, b\rangle$, and (when $\bar{\boldsymbol{O}} \neq \diamond$) $\bar{\boldsymbol{O}} = \langle l, O, b\rangle$.

Theorem 5.1 *The worst time complexity of the application rules (1,2,3,4,5,6,7,8,10,12,13) is* $O(n^{6})$ *where* n *is the length of the input string. The worst space complexity is* $O(n^{5})$.

5.2 Correctness results

Two main theorems establish the correctness of derivable items w.r.t. derivable configurations.

A derivable item is either the **initial item** or an item resulting from the application of a combination rules on derivable items. The initial item $\langle 0, \epsilon\rangle\langle 0, \epsilon\rangle\models[\diamond\diamond]\left\langle 0, \$_0, \models^{\mathbf{w}}\right\rangle\mathbf{w}$ stands for the virtual derivation step $(\mathbf{w}, 0, \epsilon, \epsilon) \vdash (\mathbf{w}, 0, \models\$_0, \models^{\mathbf{w}})$.

Theorem 5.2 (Soundness) *For every derivable item* $\mathcal{I} = \boldsymbol{A}\boldsymbol{B}\delta[\bar{\boldsymbol{D}}\boldsymbol{E}]\bar{\boldsymbol{C}}m$, *there exists a derivation on configurations*

$$\langle 0, \epsilon\rangle \mathrel{\vdash\!\frac{\star}{D}} \mathcal{U} \mathrel{\vdash\!\frac{\star}{d}} \mathcal{V}$$

such that $\mathcal{U} \mathrel{\vdash\!\frac{\star}{d}} \mathcal{V}$ *is a CF or xCF derivation representable by* I.

Proof: By induction on the item derivation length and by case analysis. ∎

Theorem 5.3 (Completeness) *For all derivable configuration $(m, w, \Xi C, \xi c)$, there exists a derivable item $\boldsymbol{AB}\delta[\bar{\boldsymbol{D}}\boldsymbol{E}]\bar{\boldsymbol{C}}m$ such that $\bar{\boldsymbol{C}} = \langle w, C, c\rangle$.*

Proof: By induction on the configuration derivation length and by case analysis of the different application rules. We also need the following "Extraction Lemma". ∎

Proposition 5.1 *From any derivation*

$$\langle 0, \epsilon\rangle \overset{\star}{\underset{D}{\vdash}} (m, w, \Xi C, \xi c)$$

may be extracted a suffix CF or xCF sub-derivation $\mathcal{U} \overset{\star}{\underset{d}{\vdash}} (m, w, \Xi C, \xi c)$ for some configuration $\mathcal{U}$.

5.3 Illustration

In the context of TAG parsing (Sect. 3), we can provide some intuition of the items that are built with $\mathcal{A}(G, \overrightarrow{\ }, \overleftarrow{\ })$, using some characteristic points encountered during the traversal of an adjunction (Fig. 6).

	after CALL	before RET
on ADJ	$A_1 A_1 \nearrow [\infty] R_1 \mathbf{w}$	$A_1 A_1 \nearrow [F_1 A_4] R_2 \mathbf{e}$
on SPINE	$A_1 S_1 \rightarrow [\infty] F_1 \mathbf{w}$	$A_1 S_1 \rightarrow [F_1 A_4] F_2 \mathbf{e}$
on FOOT	$B_1 F_1 \searrow [\infty] A_3 \mathbf{w}$	$B_1 F_1 \searrow [G_1 B_4] A_4 \mathbf{e}$

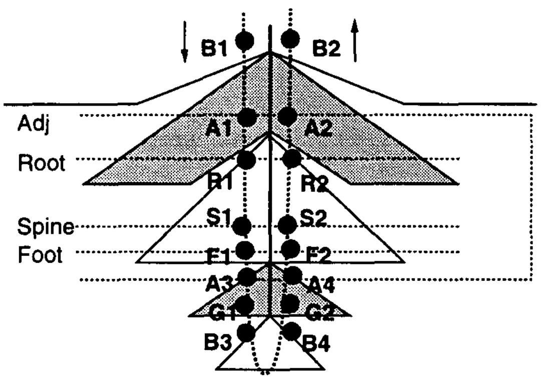

Figure 6: Adjunction and Items

6 Conclusion

This paper unifies different results about TAGs and LIGs in an uniform setting and illustrates the advantages of a clear distinction between the use of an operational device and the evaluation of this device. The operational device (here SD-2SA) helps us to focus on the description of parsing strategies (for LIGs and TAGs), while, independently, we design an efficient evaluation mechanism for this device (here tabular interpretation with complexity $O(n^6)$).

Besides illustrating a methodology, we believe our approach also opens new axes of research.

For instance, even if the tabular interpretation we have presented has (we believe) the best possible complexity, it is still possible (using techniques outside the scope of this paper, (Barthélemy and Villemonte de la Clergerie, 1996)) to improve its efficiency by refining what information should be kept in each kind of items (hence increasing computation sharing and reducing the number of items).

To handle TAGs or LIGs with attributes, we also plan to extend SD-2SA to deal with first-order terms (rather than just symbols) using unification to apply transitions and subsumption to check items.

References

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

Miguel Angel Alonso Pardo, Eric de la Clergerie, and Manuel Vilares Ferro. 1997. Automata-based parsing in dynamic programming for Linear Indexed Grammars. In A. S. Narin'yani, editor, *Proc. of DIALOGUE'97 Computational Linguistics and its Applications International Workshop*, pages 22–27, Moscow, Russia, June.

F. P. Barthélemy and E. Villemonte de la Clergerie. 1996. Information flow in tabular interpretations for generalized push-down automata. To appear in journal of TCS.

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

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.

Aravind K. Joshi. 1987. An introduction to tree adjoining grammars. In Alexis Manaster-Ramer, editor, *Mathematics of Language*, pages 87–115. John Benjamins Publishing Co., Amsterdam/Philadelphia.

Mark-Jan Nederhof. 1997. Solving the correct-prefix property for TAGs. In T. Becker and H.-V. Krieger, editors, *Proc. of MOL'97*, pages 124–130, Schloss Dagstuhl, Germany, August.

Mark-Jan Nederhof. 1998. 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.

Owen Rambow. 1994. *Formal and Computational Aspects of Natural Language Syntax.* Ph.D. thesis, University of Pennsylvania.

K. Vijay-Shanker. 1988. *A Study of Tree Adjoining Grammars.* Ph.D. thesis, University of Pennsylvania, January.