A Tabular Interpretation of Bottom-up Automata for TAG

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

Miguel A. Alonso Pardo
Depto. de Computación
Universidad de La Coruña
Campus de Elviña s/n
15071 La Coruña
Spain
alonso@dc.fi.udc.es

David Cabrero Souto
Centro Ramón Piñeiro para a
Investigación en Humanidades
Estrada Santiago-Noia km 3
15896 Santiago de Compostela
Spain
dcabrero@cirp.es

Abstract

We present a tabular interpretation for a class of 2–Stack Automata that may be used to describe bottom-up parsing strategies for TAGs. The results are also useful for tabulating other existing bottom-up automata models for this kind of languages.

1 Introduction

Several extensions of push-down automata has been proposed as operational devices for describing parsing strategies for TAGs. Embedded Push-Down Automata [EPDA] (Vijay-Shanker, 1988) and 2–Stack Automata [2–SA] (Becker, 1994) are suitable operational devices for top-down strategies. For bottom-up strategies, Bottom-up EPDA [BEPDA] (Schabes and Vijay-Shanker, 1990; Rambow, 1994) and Linear Indexed Automata [LIA] (Nederhof, 1998) have been proposed.

We classify parsing strategies for TAGs w.r.t. the way adjoining is recognized and regardless of how elementary trees are traversed. In *Top-Down* strategies, the auxiliary tree to be adjoined is predicted once the adjoining node has been reached. Examples are the Earley-like parsing algorithms which preserve the correct prefix property (Nederhof, 1997). Conversely, in *Bottom-Up* strategies, adjoining is considered only when a candidate auxiliary tree has been completely traversed. Examples are the popular CYK-like (Vijay-Shanker and Joshi, 1985) and Earley-like parsing algorithms without the valid prefix property (Schabes, 1991).

A TAG parser must handle elementary tree traversing as well as adjoining processing and keep some information about these two kinds of task. Then, a 2-stack automata is adequate to implement parsing algorithms for TAG.

Polynomial time complexity can be lost for a non deterministic grammar if redundant computations are not discarded using some kind of dynamic programming (tabular) techniques. For the above mentioned automata models, systematic tabulation is only available for LIA.

The automata model proposed in this paper for bottom-up parsing strategies presents the following characteristics: separation of the tree traversal and adjunction information by using two stacks; systematic tabulation, achieving $\mathcal{O}(n^6)$ time complexity and $\mathcal{O}(n^4)$ space complexity; and results comparable with existing tabular algorithms for TAGs.

2 (Strongly-driven) bottom-up 2–Stack Automata

Strongly Driven 2–Stack Automata [SD 2–SA] has been introduced in (de la Clergerie and Alonso Pardo, 1998) to describe arbitrary parsing strategies for TAGs. They work on 2 stacks with some restrictions added to make them equivalent, w.r.t. the recognized languages, to the class of tree adjoining languages.

A SD 2–SA uses the **Master Stack MS** to drive the evaluation and the **Auxiliary Stack AS** for restricted bookkeeping. Actually, **AS** should be considered as a stack of stacks, each of them representing a **session**. Typically, in TAG parsing, a session contains a sequence of adjunctions done along the spines of auxiliary trees. A session starts in mode **w** (write) where pop action are forbidden on **MS** and switches at some point to mode **e** (erase) where push actions are forbidden on **MS**. The actions on **AS** in mode **e** should faithfully retrace the actions done in mode **w**. Exiting a session is only possible when reaching back (in **e** mode) the **MS** element that initiated the session and when the session stack on **AS** is empty.

The bottom-up "projection" of SD 2–SA, henceforth BU 2–SA, imposes an additional restriction: **AS** must remain empty in mode **w**. That means that adjunction can be only recognized when a complete auxiliary tree has been constructed. The different behaviors of SD 2–SA and BU 2–SA are obvious when comparing the shape of derivations as illustrated in Fig. 1, where the axis display the stack sizes.

More formally, a BU 2–SA $\mathcal{A}$ is specified by a 6-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 $\$_f$

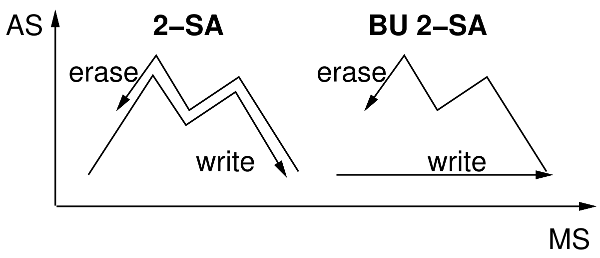

Figure 1: Derivation shapes for SD and BU 2–SA

are distinguished elements of $\mathcal{M}$. Θ is a finite set of transitions.

MS is a word in $(\mathcal{D}\mathcal{M})^\star$ where $\mathcal{D}$ denotes the set $\{\rhd, \models\}$ of **action marks**, projection of the larger action mark set $\{\nearrow, \rightarrow, \searrow, \models\}$ used for SD-2SA. Pushing an element on **MS** is either marked with $\models$ if a "new session" starts at the same time, or by $\rhd$ otherwise.

AS is a word of $(\mathcal{K}\mathcal{X}^\star)^\star$ where symbols in $\mathcal{K} = \{\models^{\mathbf{w}}, \models^{\mathbf{e}}\}$ are used to delimit session stacks and remember the mode of the previous session.

Given some input string $x_1 \dots x_n \in \Sigma^\star$, a configuration of $\mathcal{A}$ is a tuple (m, i, Ξ, ξ) where $m \in \{\mathbf{w}, \mathbf{e}\}$ denotes the current mode, i the current string position in $[0, n]$, Ξ the master stack and ξ the auxiliary stack. The initial configuration of $\mathcal{A}$ is $(\mathbf{w}, 0, \models\$_0, \models^{\mathbf{w}})$ and the final one $(\mathbf{e}, n, \models\$_f, \models^{\mathbf{w}})$.

A transition τ is represented by a pair $(m, \Xi, \xi) \overset{z}{\longmapsto} (m', \Theta, \theta)$ where $m, m' \in \{\mathbf{w}, \mathbf{e}\}$, z in $\Sigma^\star$, Ξ and Θ are suffixes of master stacks in $\mathcal{M}(\mathcal{D}\mathcal{M})^\star$, and ξ, θ suffixes of auxiliary stacks in $(\mathcal{X} \cup \mathcal{K})^\star$. We denote $(m, i, \Psi\Xi, \psi\xi) \vdash (m', j, \Psi\Theta, \psi\theta)$ a valid derivation step using τ with $z = x_{i+1} \dots x_i$, and by $\overset{\star}{\vdash}$ the reflexive and transitive closure of $\vdash$. A string $a_1 \dots x_n$ is accepted by $\mathcal{A}$ if $(\mathbf{w}, 0, \models\$_0, \models^{\mathbf{w}}) \overset{\star}{\vdash} (\mathbf{e}, n, \models\$_f, \models^{\mathbf{w}})$.

For BU 2–SA, we consider the following kinds of transitions (which enforce that the **AS** topmost session remains empty in **w** mode), namely **SWAP** to change the top element of the **MS**; $\models$–**WRITE** and $\models$–**ERASE** to start and end sessions; and $\rhd$–**WRITE** and δ–**ERASE** ($\delta \in \{\nearrow, \rightarrow, \searrow\}$) to push to and pop from **MS** while acting on **AS**:

SWAP1 $(p, A, \epsilon) \overset{z}{\longmapsto} (p, B, \epsilon)$

SWAP2 $(\mathbf{w}, A, \models^o) \overset{z}{\longmapsto} (\mathbf{e}, B, \models^o)$

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

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

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

δ–**ERASE** $(\mathbf{e}, A{\rhd}B, c) \overset{z}{\longmapsto} (\mathbf{e}, C, c')$ with
($\delta = \rightarrow$ and $c = c' = \epsilon$) or ($\delta = \nearrow$ and $c = \epsilon$) or ($\delta = \searrow$ and $c' = \epsilon$).

3 TAG parsing with BU 2–SA

We present a BU 2–SA that simulates a Earley-like parsing algorithm without the valid-prefix property (Schabes, 1991). The automata performs full prediction on the context-free backbone but no prediction on the adjunctions during the descent phase.

Each elementary tree is represented by a set of context free productions of the form $\nu_{k,0} \rightarrow \nu_{k,1} \dots \nu_{k,n_k}$, where $\nu_{k,0}$ denotes some non-leaf node k and $\nu_{k,i}$ the i^{th} son of k, and a set of terminal productions $\nu_{k,0} \rightarrow a_k$, where $\nu_{k,0}$ denotes some leaf node k with terminal label a_k.

The 6-tuple $(V_T, \mathcal{M}, \mathcal{X}, \nu_{0,0}, \nu_{0,0}', \Theta)$ defines the automata $\mathcal{A}$, with $\mathcal{M} = \{\nabla^\alpha_{k,i}\} \cup \{\nu^\alpha_{k,i}\} \cup \{{\nu^\alpha_{k,i}}'\}$ and $\mathcal{X} = \{\nabla^\alpha_{k,i}\}$, where symbols $\nabla_{k,i}$ denote dotted productions and $\nu^\alpha_{k,i}$ (resp. ${\nu^\alpha_{k,i}}'$) denote the prediction (resp. successful recognition) of a node. The transitions are given by the following rules:

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

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

- 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. At return we check if the subtree attached to the foot node of β corresponds to the subtree rooted by $\nu_{k,0}$.

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

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

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

- Call / Return for a foot node f_β. A candidate adjunction node for β is predicted. At return we remember what node was considered.

FCALL : $(\mathbf{w}, f_\beta, \epsilon) \longmapsto (\mathbf{w}, f_\beta{\rhd}\nabla_{k,0}, \epsilon)$
FRET : $(\mathbf{e}, f_\beta{\rhd}\nabla_{k,n_k}, \epsilon) \longmapsto (\mathbf{e}, {f_\beta}', \nabla_{k,n_k})$

- Production Selection
  SEL : $(\mathbf{w}, \nu_{k,0}, \epsilon) \longmapsto (\mathbf{w}, \nabla_{k,0}, \epsilon)$
- Production Publishing
  PUB : $(m, \nabla_{k,n_k}, \epsilon) \longmapsto (\mathbf{e}, {\nu_{k,0}}', \epsilon)$
- Scanning
  SCAN : $(\mathbf{w}, \nu_{k,0}, \models^m) \overset{a_k}{\longmapsto} (\mathbf{e}, {\nu_{k,0}}', \models^m)$

4 Tabulation

In a tabular framework, items store essential information about characteristics "points" of elementary derivations. Tabulation of SD 2–SA (de la Clergerie and Alonso Pardo, 1998), that achieves

$\mathcal{O}(n^6)$ time and $\mathcal{O}(n^5)$ space complexity, needs two kinds of items, namely 3-point Context-Free [CF] items and 5-point escaped Context-free [XCF] items. Each point is either a *mini configuration* $\langle i, A, a\rangle$ or a *micro configuration* $\langle i, A\rangle$ that stores some relevant information about a configuration, namely the position i in the input string, the top **MS** element A, and optionally the top **AS** element a. The uppermost curve of Fig. 2 illustrates a 3-point CF item $[\langle h, A, -\rangle, \langle i, B, -\rangle, \delta, \langle j, C, c\rangle]$, also denoted $\boldsymbol{B}\delta\bar{\boldsymbol{C}}\mathbf{w}$ where $\boldsymbol{A}$ and $\boldsymbol{B}$ are micro configurations and $\bar{\boldsymbol{C}}$ is a mini configuration. The uppermost curve of Fig. 3 illustrates a 5-point XCF item $[\langle h, A, -\rangle, \langle i, B, -\rangle, \delta, \langle p, D, d\rangle, \langle q, E, -\rangle, \langle j, C, c\rangle]$, also denoted $\boldsymbol{AB}\delta[\bar{\boldsymbol{D}}\boldsymbol{E}]\bar{\boldsymbol{C}}\mathbf{e}$ where $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{E}$ (resp. $\bar{\boldsymbol{D}}$, $\bar{\boldsymbol{C}}$) are micro (resp. mini) configurations.

BU 2–SA restrictions imply that **AS** remains empty in **w** mode, so the points $\boldsymbol{A}$, $\boldsymbol{B}$ and $\bar{\boldsymbol{C}}$ of a CF item and the points $\boldsymbol{A}$, $\boldsymbol{B}$ and $\bar{\boldsymbol{D}}$ of a XCF item are "projected" w.r.t. the top element of the **AS**. Furthermore, it may be shown that point A is actually redundant and can be discarded. The bottom curve of Fig. 2 illustrates a BU 2–SA CF item $[\langle i, B, -\rangle, \triangleright, \langle j, C, c\rangle]$, also denoted as $\boldsymbol{B}\triangleright\bar{\boldsymbol{C}}\mathbf{w}$ The bottom curve of Fig. 3 illustrates a BU 2–SA XCF item $[\langle i, B, -\rangle, \triangleright, \langle p, D, \models^o\rangle, \langle q, E, -\rangle, \langle j, C, c\rangle]$, also denoted as $\boldsymbol{B}\triangleright[\boldsymbol{DE}]\bar{\boldsymbol{C}}\mathbf{e}$. In both figures, the projection is materialized by the dashed arrows.

Formally, we identify two kinds of items for BU 2-SA, associated to two different kinds of derivations:

Bottom-up CF [buCF]
items correspond to context-free derivations that depend only on the topmost element of **MS**

$$\text{or}\quad \begin{array}{l}(\mathbf{w}, i, \Xi B, \xi\models^o)\overset{\star}{\vdash}(m, j, \Xi B\triangleright C, \xi\models^o)\\ (o, i, \Xi B, \xi)\overset{\star}{\vdash}(\mathbf{w}, j, \Xi B\models C, \xi\models^o)\end{array}$$

and are denoted by $\boldsymbol{B}\delta\bar{\boldsymbol{C}}m$, where $\boldsymbol{B} = \langle i, B\rangle$, $\bar{\boldsymbol{C}} = \langle j, C, \models^o\rangle$, and $\delta \in \mathcal{D}$.

Bottom-up Escaped CF [buXCF] items correspond to escaped context-free derivations of the form:

$$\begin{array}{rcl}(\mathbf{w}, i, \Xi B, \xi\models^o) & \overset{\star}{\vdash} & (\mathbf{w}, p, \Xi\Phi D, \xi\models^o)\\ & \overset{\star}{\vdash} & (\mathbf{e}, q, \Xi\Phi D\triangleright E, \xi\models^o\phi)\\ & \overset{\star}{\vdash} & (\mathbf{e}, j, \Xi B\triangleright C, \xi\models^o\phi c)\end{array}$$

and are denoted by $\boldsymbol{B}\triangleright[\boldsymbol{DE}]\bar{\boldsymbol{C}}\mathbf{e}$, where $\boldsymbol{B} = \langle i, B\rangle$, $\boldsymbol{D} = \langle p, D\rangle$, $\boldsymbol{E} = \langle q, E\rangle$, $\bar{\boldsymbol{C}} = \langle j, C, c\rangle$.

A set of rules combines items and transitions in order to retrieve all possible derivations. Due to space limitations, we only describe the most complex rule (see Fig. 4), used to apply a transition $\tau = (\mathbf{e}, B\triangleright C, c) \overset{z}{\longmapsto} (\mathbf{e}, F, \epsilon)$, omitting the scanning constraint z on the input string:

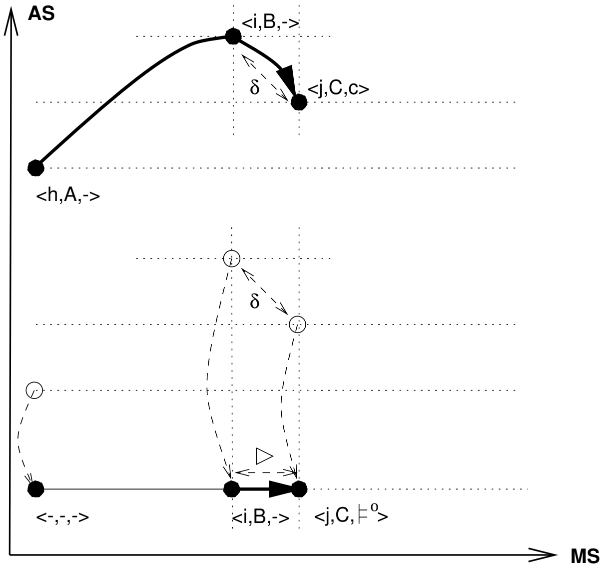

Figure 2: CF items for SD 2–SA and BU 2–SA

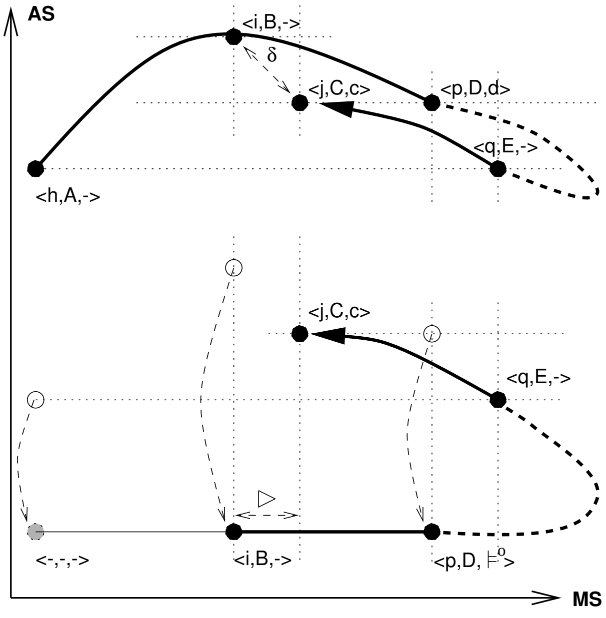

Figure 3: XCF items for SD 2–SA and BU 2–SA

$$\left.\begin{array}{l}\bar{\boldsymbol{B}}^\circ\triangleright[\boldsymbol{D}\bar{\boldsymbol{E}}^\circ]\bar{\boldsymbol{C}}\mathbf{e}\\ \boldsymbol{N}\delta\bar{\boldsymbol{B}}\mathbf{w}\\ \boldsymbol{D}\triangleright[\boldsymbol{OP}]\bar{\boldsymbol{E}}\mathbf{e}\end{array}\right\}\overset{\tau}{\Longrightarrow}\boldsymbol{N}\delta[\boldsymbol{OP}]\bar{\boldsymbol{F}}\mathbf{e} \qquad (1)$$

where $\bar{\boldsymbol{C}} = \langle j, C, c\rangle$, $\boldsymbol{B} = \langle i, B, \models^o\rangle$, $\bar{\boldsymbol{F}} = \langle k, F, b\rangle$, and $\bar{\boldsymbol{B}}^\circ = \langle i, B\rangle$ the projection of $\bar{\boldsymbol{B}}$ to a micro configuration.

The time complexity of this rule is $\mathcal{O}(n^7)$ but may be reduced to $\mathcal{O}(n^6)$ by partially applying the rule on the first two items to build an intermediary structure where $\boldsymbol{B}$ is discarded.

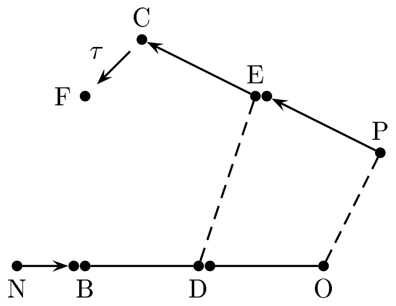

Figure 4: Application of Rule 1

Space complexity of the tabular technique for BU 2–SA is obviously $\mathcal{O}(n^4)$ as at most 4 indices are stored in buXCF items.

5 Related work

Our tabular interpretation may be used to reinterpret other existing tabular algorithms for TAGs, based on some automata model or not.

Linear Indexed Automata [LIA] (Nederhof, 1998) is the only other automata model we are aware of that has an associated tabular algorithm. This algorithm considers items $((B, C, i, j), (\diamond, \square, \square, 0, 0))$ corresponding to buCF items $\boldsymbol{B}\delta\boldsymbol{C}m$, as well as items $((B, C, i, j), (c, D, E, p, q))$ corresponding to buXCF items $\boldsymbol{B}\triangleright[\boldsymbol{DE}]\bar{\boldsymbol{C}}\mathbf{e}$. Because LIAs work on a stack of stacks, the empty stack markers we use are useless, the $\models$ mark being implicit when the second part of an item is equal to $(\diamond, \square, \square, 0, 0)$.

If we now consider the tabular algorithm of (Vijay-Shanker and Weir, 1994), which is not based on an automata model, we find that, using their terminology, our buXCF items $\boldsymbol{B}\triangleright[\boldsymbol{DE}]\bar{\boldsymbol{C}}\mathbf{e}$ correspond to a *head* $\boldsymbol{B}\bar{\boldsymbol{C}}$ with a *terminator pointer* $[\boldsymbol{DE}]$ and buCF items to a head, without terminator pointer.

In both cases, marks and modes ($\mathbf{w}$ and $\mathbf{e}$) are absent from the proposed items, but one may show that they are actually implicitly present. They may be also be discarded from our items when considering specific parsing strategies, but are needed if one wishes to exploit the full potentiality of BU-2SA, for instance for more complex parsing strategies.

6 Conclusion

Bottom-up 2–SA may be seen as the projection of a subclass of strongly-driven 2–SA, specialized to describe parsing strategies for TAG where adjunction is recognized in a bottom-up way (i.e. when being in mode erase). A tabular interpretation of BU 2–SA is straightforwardly derived by "projecting" the tabular interpretation for SD 2–SA. So, a buXCF item $\boldsymbol{B}\triangleright[\boldsymbol{DE}]\bar{\boldsymbol{C}}\mathbf{e}$ is the projection of a XCF item $\boldsymbol{AB}\delta[\bar{\boldsymbol{D}}\boldsymbol{E}]\bar{\boldsymbol{C}}\mathbf{e}$ and a buCF item $\boldsymbol{B}\delta\boldsymbol{C}m$ is the projection of a CF item $\boldsymbol{AB}\delta\boldsymbol{C}m$. For SD 2–SA, $\boldsymbol{A}$ is needed to handle popping on $\mathbf{AS}$ in $\mathbf{w}$ mode, but it may be safely removed for BU 2–SA because of the extra condition on the emptiness of $\mathbf{AS}$ in $\mathbf{w}$ mode. While the worst case time complexity remains $\mathcal{O}(n^6)$, the worst case space complexity decreases from $\mathcal{O}(n^5)$ for 2–SA to $\mathcal{O}(n^4)$ for BU 2–SA. Of course, the drawback is the violation of the valid-prefix property and it remains to investigate whether or not this is a good thing for TAG grammars used in Natural Language Processing.

7 Acknowledgements

This work has been partially supported by the European Union (1FD97-0047-C04-02), Government of Spain (HF97-223) and Xunta de Galicia (XUGA10505B96 and XUGA20402B97).

References

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

Eric de la Clergerie and Miguel A. Alonso Pardo. 1998. A tabular interpretation of a class of 2-Stack Automata. In *Proc. COLING/ACl'98*, Montreal, Canada, August.

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. 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 Languege Syntax.* Ph.D. thesis, University of Pennsylvania.

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.

Yves Schabes. 1991. The valid prefix property and left to right parsing of tree-adjoining grammar. In *Proc. of II International Workshop on Parsing Technologies, IWPT'91*, pages 21–30, Cancún, Mexico.

K. Vijay-Shanker and Aravind K. Joshi. 1985. Some computational properties of tree adjoining grammars. In *23rd Annual Meeting of the Association for Computational Linguistics*, pages 82–93, Chicago, IL, USA, July.

K. Vijay-Shanker and David J. Weir. 1994. Parsing some constrained grammar formalisms. *Computational Linguistics*, 19(4):591–636.

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