Bidirectional Push Down Automata

Miguel A. Alonso[1], Víctor J. Díaz[2], and Manuel Vilares[1,3]

[1] Departamento de Computación, Universidade da Coruña
Campus de Elviña s/n, 15071 La Coruña, Spain
{alonso,vilares}@udc.es
http://www.grupocole.org
[2] Departamento de Lenguajes y Sistemas Informáticos, Universidad de Sevilla
Avda. Reina Mercedes s/n, 41012 Sevilla (Spain)
vjdiaz@lsi.us.es
[3] Escuela Superior de Ingeniería Informática, Universidade de Vigo
Campus As Lagoas s/n, 32004 Orense, Spain
vilares@ei.uvigo.es

Abstract. We define a new model of automata for the description of bidirectional parsing strategies for context-free grammars and a tabulation mechanism that allow them to be executed in polynomial time. This new model of automata provides a modular way of defining bidirectional parsers, separating the description of a strategy from its execution.

1 Introduction

The task of designing correct and efficient parsing algorithms can be simplified by separating the definition of the parsing strategy from its tabular execution. This can be accomplished through the use of automata: the actual parsing strategy can be described by means of the construction of a non-deterministic pushdown automaton, and tabulation is introduced by means of some generic mechanism such as memoization. The construction of parsers in this way allows more straightforward proofs of correctness and makes parsing strategies easier to understand and implement.

This approach has been successfully applied to the design of parsing algorithms for context-free grammars that read the input string left-to-right [5, 6]. In this article, we define new models of push-down automata which can start reading the input string in any position, spanning to the left and to the right to include substrings which were themselves read in the same bidirectional way. Tabulation techniques are provided in order to execute efficiently these automata.

This article is outlined as follows. Section 2 introduces push-down automata. A bidirectional extension of push-down automata is presented in section 3. Two different tabular frameworks to execute efficiently bidirectional automata are defined in Section 5. These frameworks are applied to predictive and bottom-up head-corner parsing algorithms.

2 Push-Down Automata

A Context-Free Grammar (CFG) is a tuple (V_N, V_T, S, P), where V_N is a finite set of non-terminal symbols, V_T a finite set of terminal symbols, $S \in V_N$ is the axiom of the grammar, and P is a finite set of productions (rewriting rules) of the form $A \to \gamma$ with $A \in V_N$ and $\gamma \in (V_T \cup V_N)^*$. Push-Down Automata (PDA) are the operational devices for parsing CFG. Following [3], we define a PDA as a tuple $(V_T, V_S, \Theta, \$_0, \$_f)$ where V_T is a finite set of terminal symbols, V_S is a finite set of stack symbols, $\$_0 \in V_S$ is the initial stack symbol, $\$_f \in V_S$ is the final stack symbol and Θ is a finite set of SWAP, PUSH and POP transitions. A configuration of a PDA is usually defined as a pair $(\xi, a_l \dots a_n)$, where $\xi \in V_S^*$ is the stack attained and $a_l \dots a_n$ the part of the input string $a_1 \dots a_n$ to be read. We consider an alternative and equivalent definition of configuration in which the position l is stored in the top element of ξ. Thus, a configuration is given by the contents of $\hat{\xi}$, a stack of pairs in $V_S \times \mathbb{N}$. The initial configuration is $(\$_0, 0)$. Other configurations are attained by applying transitions as follows:

- The application of a SWAP transition of the form $C \overset{a}{\longmapsto} F$ to a configuration $\hat{\xi}(C, l)$ yields a configuration $\hat{\xi}(F, l + |a|)$ as a result of replacing C by F and scanning the terminal $a = a_{l+1}$ or the empty string $a = \epsilon$.
- The application of a PUSH transition $C \longmapsto C\,F$ to a configuration $\hat{\xi}(C, l)$ yields a configuration $\hat{\xi}(C, l)(F, l)$ as a result of pushing F onto C.
- The application of a POP transition of the form $CF \longmapsto G$ to a configuration $\hat{\xi}(C, l)(F, m)$ yields a configuration $\hat{\xi}(G, m)$ as a result of popping C and F, which are replaced by G.

where $C, F, G \in V_S$ and $a \in V_T \cup \{\epsilon\}$. An input string $w = a_1 \dots a_n$ is succesfully recognized by a PDA if the final configuration $(\$_0, 0)(\$_f, n)$ is attained.

Only SWAP transitions can scan elements from the input string. This is not a limitation, as a scanning push transition $C \overset{a}{\longmapsto} C\,F$ could be emulated by the consecutive application of two transitions $C \longmapsto C\,F'$ and $F' \overset{a}{\longmapsto} F$, while a scanning pop transition $C\ F \overset{a}{\longmapsto} G$ could be emulated by $C\ F \longmapsto G'$ and $G' \overset{a}{\longmapsto} G$, where F' and G' are fresh stack symbols.

We call transitions SWAP, PUSH and POP *r-transitions* as they can only read the input string from the left to the right. Thus, push-down automata can only be used to implement unidirectional parsing strategies that read the input string in the same way. As an example of the kind of parsers that can be implemented, a compilation schema[1] of a context-free grammar into a push-down automaton implementing the Earley's parsing strategy [4] is derived. In the resulting automaton, V_T is the set of terminals of the source grammar, V_S is the union of $\{\$_0, \$_f\}$ and a set of dotted productions[2], the initial element $\$_0$ is used to start computations, the final element $\$_f$ is $(S \to \delta\bullet)$ and Θ contains the

[1] A compilation schema is a set of rules indicating how to construct an automaton according to a given grammar and parsing strategy.

[2] Dotted productions $A \to \alpha \bullet \beta$ are used to indicate that the part α of the production has been recognized.

$$\textbf{[INIT]} \quad \$_0 \overset{\epsilon}{\longmapsto} \$_0 \ (S \to \bullet\alpha)$$

$$\textbf{[PRED]} \ (A \to \alpha \bullet B\beta) \longmapsto (A \to \alpha \bullet B\beta)\ (B \to \bullet\gamma)$$

$$\textbf{[SCAN]} \ (A \to \alpha \bullet a\beta) \overset{a}{\longmapsto} (A \to \alpha a \bullet \beta)$$

$$\textbf{[COMP]} \ (A \to \alpha \bullet B\beta)\ (B \to \gamma\bullet) \longmapsto (A \to \alpha B \bullet \beta)$$

Fig. 1. Compilation schema for the Earley's strategy

$$\frac{[B, j; C, l] \quad C \overset{a}{\longmapsto} F}{[B, j; F, l + |a|] \quad a = a_{l+1} \text{ or } a = \epsilon} \qquad \frac{[C, j, l] \quad C \overset{a}{\longmapsto} F}{[F, j, l + |a|] \quad a = a_{l+1} \text{ or } a = \epsilon}$$

$$\frac{[B, j; C, l]}{[C, l; F, l]} \ C \longmapsto C\,F \qquad \frac{[C, j, l]}{[F, l, l]} \ C \longmapsto C\,F$$

$$\frac{[C, l; F, m] \quad [B, j; C, l]}{[B, j; G, m]} \ C\,F \longmapsto G \qquad \frac{[F, l, m] \quad [C, j, l]}{[G, j, m]} \ C\,F \longmapsto G$$

Fig. 2. Inference rules for PDA with S^2 items (left-hand) and S^1 items (right-hand)

set of transitions derived by the compilation rules shown in Fig. 1. A **[INIT]** transition is in charge of starting the parsing process. **[LPRED]** transitions predict a non-terminal B placed just after the dot in a given production. Once a production having this non-terminal in its left-hand side has been completely recognized, the dot is advanced by the application of a transition **[COMP]**. Terminals are recognized by **[SCAN]** transitions.

The direct execution of PDA may be exponential with respect to the length of the input string and may even loop. To get polynomial complexity, we must avoid duplicating computations by tabulating traces of configurations called *items*. The amount of information to keep in an item is the crucial point to determine to get efficient executions. Following [3] we know that S^2 items, storing the two elements placed on the top of the configuration stack, can be used to design tabular interpretations which are sound and complete for any parsing strategy.

New items are derived from existing items by means of inference rules of the form $\frac{antecedents}{consequent}$ *conditions* similar to those used in grammatical deduction systems [8], meaning that if all antecedents are present and conditions are satisfied then the consequent item should be generated. Conditions usually refer to transitions of the automaton and to terminals from the input string. The set of inference rules for S^2 items is shown in Fig. 2. Computations start with the item $[\Box, 0; \$_0, 0]$ and finish with the item $[\$_0, 0; \$_f, n]$.

$$\textbf{[INIT]}\ \frac{[\$_0, 0, 0]}{[S \rightarrow \bullet\alpha, 0, 0]}$$

$$\textbf{[PRED]}\ \frac{[A \rightarrow \alpha \bullet B\beta, j, l]}{[B \rightarrow \bullet\gamma, l, l]}$$

$$\textbf{[SCAN]}\ \frac{\begin{array}{c}[A \rightarrow \alpha \bullet a\beta, j, l] \\ [a, l, l+1]\end{array}}{[A \rightarrow \alpha a \bullet \beta, j, l+1]}$$

$$\textbf{[COMP]}\ \frac{\begin{array}{c}[B \rightarrow \gamma\bullet, l, m] \\ [A \rightarrow \alpha \bullet B\beta, j, l]\end{array}}{[A \rightarrow \alpha B \bullet \beta, j, m]}$$

Fig. 3. Deduction steps for the Earley's algorithm

From [3] we also know that if the results of the non-deterministic computations are constrained only by bottom-up propagation of computed facts (e.g. bottom-up and Earley strategies, but not pure top-down strategies) S^1 items storing only the top element of the configuration stack can be used to derive a sound and complete tabular interpretation. The set of inference rules for S^1 items is shown in Fig. 2.[3] Computations start with the item $[\$_0, 0, 0]$ and finish with $[\$_f, 0, n]$. Fig. 3 shows the set of deduction steps corresponding to the parsing schema[4] Earley obtained by applying these inference rules to an automaton resulting from the compilation schema shown in Fig. 1. The worst-case time complexity for the S^2 and S^1 inference rules and for the Earley schema is $\mathcal{O}(n^3)$.

Although PDA can only be used to describe unidirectional strategies, their tabulation technique can be extended to read the input string left-to-right, right-to-left and bidirectionally [7]. This kind of bidirectional tabulation makes possible to implement robust parsers by means of PDA but it does not make possible to specify bidirectional parsing strategies due to PDA transitions does not provide any way of controlling the direction of the parsing process.

[3] Although the items involved in these inference rules are usually called S^1 items, actually they are not true S^1 items but $S^{1+\epsilon}$ items due to each item $[C, j, l]$ stores the top element (C, l) of the configuration stack plus the position j of the second element (B, j) [2].

[4] In brief, a parsing schema is a deductive parsing system where inference rules are called *deduction steps* and conditions on the existence of a given terminal a_{l+1} are represented by means of special antecedent items of the form $[a, l, l+1]$ called *hypothesis*.

3 Bidirectional Push-Down Automata

Bidirectional parsing strategies can start computations at any position of the input string and can span to the right and to the left to include substrings which were scanned in a bidirectional way by some subcomputations. As a first step towards the definition of a Bidirectional Push-Down Automata (BPDA), we must adapt configurations in order to be able to represent the discontinuos recognition of the input string. Thus, configurations of a BPDA will be given by the contents of Ξ, a stack of triples in $V_S \times \mathbb{N} \times \mathbb{N}$. The initial configuration is $(\$_0, 0, 0)$. Other configurations are attained by applying transitions as follows:

- The application of a SWAP_R transition of the form $C \overset{a}{\longmapsto}_R F$ to a configuration $\Xi(C, k, l)$ yields a configuration $\Xi(F, k, l+|a|)$ as a result of replacing C by F and scanning the terminal $a = a_{l+1}$ or the empty string $a = \epsilon$ to the right of the substring spanned by C.
- The application of a SWAP_L transition of the form $C \overset{a}{\longmapsto}_L F$ to a configuration $\Xi(C, k, l)$ yields a configuration $\Xi(F, k-|a|, l)$ as a result of replacing C by F and scanning the terminal $a = a_k$ or the empty string $a = \epsilon$ to the left of the substring spanned by C.
- The application of a PUSH_R transition $C \longmapsto_R C\,F$ to a configuration $\Xi(C, k, l)$ yields a configuration $\Xi(C, k, l)(F, l, l)$. It is expected that F will span a substring inmediatly to the right of the substring spanned by C.
- The application of a PUSH_L transition $C \longmapsto_L C\,F$ to a configuration $\Xi(C, k, l)$ yields a configuration $\Xi(C, k, l)(F, k, k)$. It is expected that F will span a substring inmediatly to the left of the substring spanned by C.
- The application of a PUSH_U transition of the form $C \overset{a}{\longmapsto}_U C\,F$ to a configuration $\Xi(C, k, l)$ yields a configuration $\Xi(C, k, l)(F, m, m + |a|)$ as a result of pushing F onto C and scanning the terminal $a = a_{m+1}$ or the empty string $a = \epsilon$. PUSH_U transitions are *undirected* in the sense that a is not necessarily adjacent to the substring spanned by C.
- The application of a POP_R transition of the form $CF \longmapsto_R G$ to a configuration $\Xi(C, k, l)(F, l, m)$ yields a configuration $\Xi(G, k, m)$. The substring spanned by F is adjacent to the right of the substring spanned by C.
- The application of a POP_L transition of the form $CF \longmapsto_L G$ to a configuration $\Xi(C, k, l)(F, m, k)$ yields a configuration $\Xi(G, m, l)$. The substring spanned by F is adjacent to the left of the substring spanned by C.

An input string $a_1 \ldots a_n$ is succesfully recognized by a BPDA if the final configuration $(\$_0, 0, 0)(\$_f, 0, n)$ is attained. SWAP_R, PUSH_R and POP_R transitions are the r-transitions corresponding to unidirectional PDA. SWAP_L, PUSH_L and POP_L transitions are *l-transitions* that advance "to the left" in the reading of the input string. However, the union of r-transitions and l-transitions is not sufficient to implement bidirectional parsers, we need PUSH_U transitions of the form $C \overset{a}{\longmapsto}_U C\,F$ to start subcomputations at any position of the input string. We guarantee that, in any computation recognizing the input string, each terminal in the input string is read only once by means of the definition of SWAP_R

and SWAP_L transitions (they can not re-read elements which are in the span of the top element of the stack) and the definition of POP_R and POP_L transitions (they can not pop stack elements spanning overlapping substrings). Pop transitions also ensure we read the input string and not a permutation of it.

We define Bidirectional Push-Down Automata (BPDA) as a tuple $(V_T, V_S, \Theta_\text{B}, \$_0, \$_f)$ with Θ_B containing SWAP_R, SWAP_L, PUSH_R, PUSH_L, POP_R, POP_L and PUSH_U transitions. As an example of the kind of parsers that can be implemented using BPDA, a compilation schema of a context-free grammar into a bidirectional push-down automaton implementing a predictive head-corner parsing strategy is derived. Head-corner parsing strategies can be applied to context-free grammars in which each production has an element of the right-hand side marked as the head of the production. For empty productions $A \to \epsilon$, the empty string ϵ is considered the head of the production. The head-corner relation $>_h$ on $V_N \times (V_N \cup V_T \cup \{\epsilon\})$ is defined by $A >_h X$ if there is a production $A \to \alpha X \beta$ with X the head of the production. If $A \to \epsilon$ then $A >_h \epsilon$. The transitive and reflexive closure of $>_h$ is denoted $>_h^*$. In the resulting automaton, V_T is the set of terminals of the source grammar, V_S is the union of $\{\$_0, \$_f\}$, the set of non-terminals of the source grammar and a set of dotted productions $A \to \alpha \bullet \beta \bullet \gamma$ used to indicate that the part β of the production has been recognized; the initial element $\$_0$ is used to start computations; the final element $\$_f$ is $(S \to \bullet\delta\bullet)$; and Θ_B contains the set of transitions derived by the compilation rules shown in Fig. 4. [**LPRED**] and [**RPRED**] transitions predict a non-terminal to the left and to the right, respectively. Once a production having this non-terminal in its left-hand side has been completely recognized, the dot is advanced by the application of a pair of transitions [**LCOMP1**]-[**LCOMP2**] or [**RCOMP1**]–[**RCOMP2**]. The head-corner of a given non-terminal is found by [$\mathbf{HC}_T$] and [$\mathbf{HC}_\epsilon$] transitions. [$\mathbf{HC}_N$] transitions traverse backwards the chain of head-corners of a given non-terminal. [**LSCAN**] and [**RSCAN**] transitions recognize terminals to the left and to the right, respectively.

4 Context-free languages and BPDA

BPDA exactly accepts the class of context-free languages. Given a CFG $\mathcal{G}$, the language accepted by the bidirectional push-down automaton built following the compilation schema shown in Fig. 4 is the language recognized by $\mathcal{G}$. Therefore, the class of context-free languages is included in the class of languages accepted by BPDA. Given a BPDA $\mathcal{A} = (V_T, V_S, \Theta_\text{B}, \$_0, \$_f)$ we can construct a CFG $\mathcal{G} = (V_N, V_T, S, P)$ where $V_N \in V_S \times V_S$, $S = \langle \$_0, \$_f \rangle$ and the productions in P are obtained from transitions in Θ_B as follows:

- A production $\langle E, F \rangle \to \langle E, C \rangle\, a$ for each $C \overset{a}{\longmapsto}_R F \in \Theta_\text{B}$ and $E \in V_S$.
- A production $\langle E, F \rangle \to a\, \langle E, C \rangle$ for each $C \overset{a}{\longmapsto}_L F \in \Theta_\text{B}$ and $E \in V_S$.
- A production $\langle C, F \rangle \to \epsilon$ for each $C \longmapsto_R C\, F \in \Theta_\text{B}$ and $E \in V_S$.
- A production $\langle C, F \rangle \to \epsilon$ for each $C \longmapsto_L C\, F \in \Theta_\text{B}$ and $E \in V_S$.
- A production $\langle C, F \rangle \to a$ for each $C \overset{a}{\longmapsto}_U C\, F \in \Theta_\text{B}$ and $E \in V_S$.

$$
\begin{array}{lll}
\textbf{[INIT]} & \$_0 \overset{\epsilon}{\longmapsto} \$_0 \ (S) & \\
\textbf{[HC}_T\textbf{]} & (A) \overset{b}{\longmapsto}_U (A)\ (B \to \alpha \bullet b \bullet \gamma) & A >_h^* B >_h b \\
\textbf{[HC}_N\textbf{]} & (C \to \bullet\delta\bullet) \overset{\epsilon}{\longmapsto}_R (B \to \alpha \bullet C \bullet \gamma) & B >_h C \\
\textbf{[HC}_\epsilon\textbf{]} & (A) \overset{\epsilon}{\longmapsto}_U (A)\ (B \to \bullet\bullet) & A >_h^* B \\
\textbf{[LPRED]} & (B \to \alpha C \bullet \beta \bullet \gamma) \longmapsto_L (B \to \alpha C \bullet \beta \bullet \gamma)\ (C) & \\
\textbf{[RPRED]} & (B \to \alpha \bullet \beta \bullet C\gamma) \longmapsto_R (B \to \alpha \bullet \beta \bullet C\gamma)\ (C) & \\
\textbf{[LSCAN]} & (B \to \alpha a \bullet \beta \bullet \gamma) \overset{a}{\longmapsto}_L (B \to \alpha \bullet a\beta \bullet \gamma) & \\
\textbf{[RSCAN]} & (B \to \alpha \bullet \beta \bullet a\gamma) \overset{a}{\longmapsto}_R (B \to \alpha \bullet \beta a \bullet \gamma) & \\
\textbf{[LCOMP1]} & (C)\ (C \to \bullet\delta\bullet) \longmapsto_L (C \to \bullet\delta\bullet) & \\
\textbf{[LCOMP2]} & (B \to \alpha C \bullet \beta \bullet \gamma)\ (C \to \bullet\delta\bullet) \longmapsto_L (B \to \alpha \bullet C\beta \bullet \gamma) & \\
\textbf{[RCOMP1]} & (C)\ (C \to \bullet\delta\bullet) \longmapsto_R (C \to \bullet\delta\bullet) & \\
\textbf{[RCOMP2]} & (B \to \alpha \bullet \beta \bullet C\gamma)\ (C \to \bullet\delta\bullet) \longmapsto_R (B \to \alpha \bullet \beta C \bullet \gamma) &
\end{array}
$$

Fig. 4. Compilation schema for a predictive head-corner strategy

- A production $\langle E, G\rangle \to \langle E, C\rangle\langle C, F\rangle$ for each $CF \longmapsto_R G \in \Theta_{\mathrm{B}}$ and $E \in V_S$.
- A production $\langle E, G\rangle \to \langle C, F\rangle\langle E, C\rangle$ for each $CF \longmapsto_L G \in \Theta_{\mathrm{B}}$ and $E \in V_S$.

Applying induction in the length of the derivations, we can show that $\langle \$_0, \$_f\rangle \overset{*}{\Rightarrow} a_1 \dots a_n$ if and only if a computation of $\mathcal{A}$ starting at $(\$_0, 0, 0)$ attains a configuration $(\$_0, 0, 0)(\$_f, 0, n)$ reading $a_1 \dots a_n$ from the input string, i.e. $\mathcal{G}$ exactly recognizes the language accepted by $\mathcal{A}$. Therefore, the class of languages accepted by BPDA is included in the class of context-free languages.

5 Tabulation of BPDA

As in the case of PDA, the direct execution of BPDA may be exponential with respect to the length of the input string and may even loop. To solve this problem, in this section we extend the S^2 and S^1 tabulation techniques to the case of bidirectional push-down automata.

5.1 The S^2 framework

From [3] we know that extensions of push-down automata can be tabulated by using S^2 items storing the two elements on the top of the configuration stack. In the case of BPDA, S^2 items are of the form $[B, i, j; C, k, l]$, indicating the part of the input string $a_{k+1} \dots a_l$ recognized by the top element C and the part $a_{i+1} \dots a_j$ recognized by the element B placed immediately under C. The

$$\frac{[B,i,j;C,k,l]}{[B,i,j;F,k,l+|a|]} \quad \begin{array}{l} C \overset{a}{\longmapsto}_R F \\ a = a_{l+1} \text{ or } a = \epsilon \end{array} \qquad \frac{[B,i,j;C,k,l]}{[C,k,l;F,m,m+|a|]} \quad \begin{array}{l} C \overset{a}{\longmapsto}_U C\,F \\ a = a_{m+1} \text{ or } a = \epsilon \end{array}$$

$$\frac{[B,i,j;C,k,l]}{[B,i,j;F,k-|a|,l]} \quad \begin{array}{l} C \overset{a}{\longmapsto}_L F \\ a = a_k \text{ or } a = \epsilon \end{array} \qquad \frac{\begin{array}{c}[C,k,l;F,l,m] \\ [B,i,j;C,k,l]\end{array}}{[B,i,j;G,k,m]} \quad C\,F \longmapsto_R G$$

$$\frac{[B,i,j;C,k,l]}{[C,k,l;F,l,l]} \quad C \longmapsto_R C\,F \qquad \frac{\begin{array}{c}[C,k,l;F,m,k] \\ [B,i,j;C,k,l]\end{array}}{[B,i,j;G,m,l]} \quad C\,F \longmapsto_L G$$

$$\frac{[B,i,j;C,k,l]}{[C,k,l;F,k,k]} \quad C \longmapsto_L C\,F$$

Fig. 5. Inference rules for S^2 items

set of inference rules for S^2 items is shown in Fig. 5. Computations start with the item $[\Box, 0, 0, ; \$_0, 0, 0]$ and finish with $[\$_0, 0, 0; \$_f, 0, n]$. The worst case time complexity with respect to the length n of the input string is $\mathcal{O}(n^5)$.

The application of this set of inference rules to an automaton resulting from the compilation schema shown in Fig. 4 yields the set of deduction steps shown in Fig. 6 (where Φ refers to a dotted production or $\$_0$) which is very close to the set of deduction steps corresponding to the predictive head-corner parsing schema **pHC** defined by Sikkel in [9, chapter 11], also working in $\mathcal{O}(n^5)$ time complexity. The main difference is that predictive steps of the schema **pHC** have stronger constraints with respect to the part of the input string to be considered when seeking for a head-corner. A minor difference is that left-completer and right completer steps have been splited into **[LCOMP1]**–**[LCOMP2]** and **[RCOMP1]**–**[RCOMP2]** pairs.

5.2 The S^1 framework

The complexity of the tabular framework can be reduced by considering more compact kinds of item. From [3] we also know that a sound and complete tabular interpretation for a given extension of push-down automata can be obtained using S^1 items that store the top element of the configuration stack. In the case of BPDA, S^1 items are of the form $[C, k, l]$, storing the top element C with the corresponding positions k and l of the substring spanned by it. The set of inference rules for S^1 items is derived from the set of inference rules for S^2, as is shown in Fig. 7. Computations start with the item $[\$_0, 0, 0]$ and finish with $[\$_f, 0, n]$. The worst case time complexity with respect to the length n of the input string is $\mathcal{O}(n^3)$.

As an example of the kind of strategies that can be implemented using the S^1 framework, we show in Fig. 8 the compilation schema of a context-free grammar into a bidirectional push-down automaton implementing a bottom-up pre-

$$\textbf{[INIT]}\,\frac{[\square, 0, 0; \$_0, 0, 0]}{[\$_0, 0, 0; S, 0, 0]}$$

$$\textbf{[HC}_T\textbf{]}\,\frac{\begin{array}{c}[\Phi, s, t; A, l, r]\\ [b, j-1, j]\end{array}}{[A, l, r; B \to \alpha \bullet b \bullet \gamma, j-1, j]} \quad A >_h^* B >_h b$$

$$\textbf{[HC}_N\textbf{]}\,\frac{[A, l, r; C \to \bullet\delta\bullet, i, j]}{[A, l, r; B \to \alpha \bullet C \bullet \gamma]} \quad B >_h C$$

$$\textbf{[HC}_\epsilon\textbf{]}\,\frac{[\Phi, s, t; A, l, r]}{[A, l, r; B \to \bullet\bullet, j, j]} \quad A >_h^* B$$

$$\textbf{[LPRED]}\,\frac{[A, l, r; B \to \alpha C \bullet \beta \bullet \gamma, i, j]}{[B \to \alpha C \bullet \beta \bullet \gamma, i, j; C, i, i]}$$

$$\textbf{[RPRED]}\,\frac{[A, l, r; B \to \alpha \bullet \beta \bullet C\gamma, i, j]}{[B \to \alpha \bullet \beta \bullet C\gamma, i, j; C, j, j]}$$

$$\textbf{[LSCAN]}\,\frac{\begin{array}{c}[A, l, r; B \to \alpha a \bullet \beta \bullet \gamma, j, k]\\ [a, j-1, j]\end{array}}{[A, l, r; B \to \alpha \bullet a\beta \bullet \gamma, j-1, k]}$$

$$\textbf{[RSCAN]}\,\frac{\begin{array}{c}[A, l, r; B \to \alpha \bullet \beta \bullet a\gamma, i, j]\\ [a, j, j+1]\end{array}}{[A, l, r; B \to \alpha \bullet \beta a \bullet \gamma, i, j+1]}$$

$$\textbf{[LCOMP1]}\,\frac{\begin{array}{c}[\Phi, j, k; C, j, j]\\ [C, j, j; C \to \bullet\delta \bullet i, j]\end{array}}{[\Phi, j, k; C \to \bullet\delta \bullet i, j]}$$

$$\textbf{[LCOMP2]}\,\frac{\begin{array}{c}[A, l, r; B \to \alpha C \bullet \beta \bullet \gamma, j, k]\\ [B \to \alpha C \bullet \beta \bullet \gamma, j, k; C \to \bullet\delta \bullet i, j]\end{array}}{[A, l, r; B \to \alpha \bullet C\beta \bullet \gamma, i, k]}$$

$$\textbf{[RCOMP1]}\,\frac{\begin{array}{c}[\Phi, i, j; C, j, j]\\ [C, j, j; C \to \bullet\delta \bullet j, k]\end{array}}{[\Phi, i, j; C \to \bullet\delta \bullet j, k]}$$

$$\textbf{[RCOMP2]}\,\frac{\begin{array}{c}[A, l, r; B \to \alpha \bullet \beta \bullet C\gamma, i, j]\\ [B \to \alpha \bullet \beta \bullet C\gamma, i, j; C \to \bullet\delta \bullet j, k]\end{array}}{[A, l, r; B \to \alpha \bullet \beta C \bullet \gamma, i, k]}$$

Fig. 6. Deduction steps for a predictive head-corner parsing schema

$$\frac{[C,k,l]}{[F,k,l+|a|]} \quad C \overset{a}{\longmapsto}_R F \quad a = a_{l+1} \text{ or } a = \epsilon$$

$$\frac{[C,k,l]}{[F,m,m+|a|]} \quad C \overset{a}{\longmapsto}_U C\,F \quad a = a_{m+1} \text{ or } a = \epsilon$$

$$\frac{[C,k,l]}{[F,k-|a|,l]} \quad C \overset{a}{\longmapsto}_L F \quad a = a_k \text{ or } a = \epsilon$$

$$\frac{[F,l,m] \quad [C,k,l]}{[G,k,m]} \quad C\,F \longmapsto_R G$$

$$\frac{[C,k,l]}{[F,l,l]} \quad C \longmapsto_R C\,F$$

$$\frac{[F,m,k] \quad [C,k,l]}{[G,m,l]} \quad C\,F \longmapsto_L G$$

$$\frac{[C,k,l]}{[F,k,k]} \quad C \longmapsto_L C\,F$$

Fig. 7. Inference rules for S^1 items

$$\textbf{[INIT]} \quad \$_0 \overset{\epsilon}{\longmapsto} \$_0 \;\; (S \to \bullet\,\bullet\,\alpha)$$

$$\textbf{[HC}_T\textbf{]} \quad (A \to \delta_1 \bullet \delta_2 \bullet \delta_3) \overset{a}{\longmapsto}_U (A \to \delta_1 \bullet \delta_2 \bullet \delta_3)\,(B \to \alpha \bullet a \bullet \gamma)$$

$$\textbf{[HC}_N\textbf{]} \quad (A \to \bullet\beta\bullet) \overset{\epsilon}{\longmapsto}_R (B \to \alpha \bullet A \bullet \gamma)$$

$$\textbf{[HC}_\epsilon\textbf{]} \quad (A \to \delta_1 \bullet \delta_2 \bullet \delta_3) \overset{b}{\longmapsto}_U (A \to \delta_1 \bullet \delta_2 \bullet \delta_3)\,(B \to \bullet\bullet)$$

$$\textbf{[LSCAN]} \quad (B \to \alpha a \bullet \beta \bullet \gamma) \overset{a}{\longmapsto}_L (B \to \alpha \bullet a\beta \bullet \gamma)$$

$$\textbf{[RSCAN]} \quad (B \to \alpha \bullet \beta \bullet a\gamma) \overset{a}{\longmapsto}_R (B \to \alpha \bullet \beta a \bullet \gamma)$$

$$\textbf{[LCOMP]} \quad (B \to \alpha A \bullet \beta \bullet \gamma)\,(A \to \bullet\delta\bullet) \longmapsto_L (B \to \alpha \bullet A\beta \bullet \gamma)$$

$$\textbf{[RCOMP]} \quad (B \to \alpha \bullet \beta \bullet A\gamma)\,(A \to \bullet\delta\bullet) \longmapsto_R (B \to \alpha \bullet \beta A \bullet \gamma)$$

Fig. 8. Compilation schema for a bottom-up head-corner strategy

dictive head-corner parsing strategy. [$\mathbf{HC}_T$] and [$\mathbf{HC}_\epsilon$] transitions start the bottom-up recognition of head-corners. [$\mathbf{HC}_N$] transitions traverse backwards the head-corner relation. Terminals are recognized to the left and to the right by **[LSCAN]** and **[RSCAN]** transitions, respectively. Once the right-hand side of a production has been completely recognized, **[LCOMP]** and **[RCOMP]** advance the dot of the production having the left-hand side of that rule to the left or to the right of the dot, respectively.

When the S^1 inference rules are applied to an automaton resulting from the compilation schema shown in Fig. 8, we obtain the set of deduction steps corresponding to the parsing schema **buHC** defined by Sikkel in [9, chapter 11], as shown in Fig. 9, considering that in the parsing schemata framework [9] the antecedent $[A \to \delta_1 \bullet \delta_2 \bullet \delta_3, l, r]$ in steps [$\mathbf{HC}_T$] and [$\mathbf{HC}_\epsilon$] can be filtered out as it does not restrict the application of these steps.

$$[\mathbf{INIT}]\ \frac{[\$_0, 0, 0]}{[S \rightarrow \bullet\ \bullet\ \alpha, 0, 0]}$$

$$[\mathbf{HC}_T]\ \frac{[A \rightarrow \delta_1 \bullet \delta_2 \bullet \delta_3, l, r] \quad [a, j-1, j]}{[B \rightarrow \alpha \bullet a \bullet \gamma, j-1, j]}$$

$$[\mathbf{HC}_N]\ \frac{[A \rightarrow \bullet\beta\bullet, i, j]}{[B \rightarrow \alpha \bullet A \bullet \gamma, i, j]}$$

$$[\mathbf{HC}_\epsilon]\ \frac{[A \rightarrow \delta_1 \bullet \delta_2 \bullet \delta_3, l, r]}{[B \rightarrow \bullet\bullet, j, j]}$$

$$[\mathbf{LSCAN}]\ \frac{[B \rightarrow \alpha a \bullet \beta \bullet \gamma, j, k] \quad [a, j-1, j]}{[B \rightarrow \alpha \bullet a\beta \bullet \gamma, j-1, k]}$$

$$[\mathbf{RSCAN}]\ \frac{[B \rightarrow \alpha \bullet \beta \bullet a\gamma, i, j] \quad [a, j, j+1]}{[B \rightarrow \alpha \bullet \beta a \bullet \gamma, i, j+1]}$$

$$[\mathbf{LCOMP}]\ \frac{[A \rightarrow \bullet\delta\bullet, i, j] \quad [B \rightarrow \alpha A \bullet \beta \bullet \gamma, j, k]}{[B \rightarrow \alpha \bullet A\beta \bullet \gamma, i, k]}$$

$$[\mathbf{RCOMP}]\ \frac{[A \rightarrow \bullet\delta\bullet, j, k] \quad [B \rightarrow \alpha \bullet \beta \bullet A\gamma, i, j]}{[B \rightarrow \alpha \bullet \beta A \bullet \gamma, i, k]}$$

Fig. 9. Deduction steps for a bottom-up head-corner parsing schema

6 Conclusions

In order to provide a common framework for the description of bidirectional parsing algorithms for context-free grammars, we have defined a new class of bidirectional push-down automata which works in polynomial time. We have also shown how tabular parsing algorithms can be derived from the automaton describing the parsing strategy, and the tabulation technique associated to the automata model. As illustration, we have considered the case of the predictive head-corner and bottom-up head-corner strategies proposed in [9] but the approach can be applied to the other bidirectional strategies defined in the literature. This approach can also be extended to automata models for extensions

of context-free grammars. In this direction, we have investigated a bidirectional version of Linear Indexed Automata for Tree Adjoining Grammars [1].

The use of bidirectional push-down automata to define parsers allowed us to concentrate on the parsing strategy itself, abstracting for details of implementation such as the input positions spanned by a production or the information we must track into items to guarantee the correctness of a parsing strategy.

Acknowledgements

This research has been partially supported by Plan Nacional de Investigación Científica, Desarrollo e Innovación Tecnológica (Grant TIC2000-0370-C02-01), Ministerio de Ciencia y Tecnología (Grant HP2001-0044), Xunta de Galicia (Grants PGIDT01PXI10506PN and PGIDIT02PXIB30501PR) and Universidade da Coruña.

References

1. Miguel A. Alonso, Víctor J. Díaz, and Manuel Vilares. Bidirectional automata for tree adjoining grammars. In *Proc. of the Seventh International Workshop on Parsing Technologies (IWPT-2001)*, pages 42–53, Beijing, China, October 2001. Tsinghua University Press.
2. Eric de la Clergerie. *Automates à Piles et Programmation Dynamique. DyALog : Une Application à la Programmation en Logique.* PhD thesis, Université Paris 7, Paris, France, 1993.
3. Eric de la Clergerie and Bernard Lang. 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 1994.
4. J. Earley. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102, 1970.
5. Bernard Lang. Towards a uniform formal framework for parsing. In Masaru Tomita, editor, *Current Issues in Parsing Technology*, pages 153–171. Kluwer Academic Publishers, Norwell, MA, USA, 1991.
6. Mark-Jan Nederhof. An optimal tabular parsing algorithm. In *Proc. of 32nd Annual Meeting of the Association for Computational Linguistics*, pages 117–124, Las Cruces, NM, USA, June 1994. ACL.
7. Mark-Jan Nederhof. Reversible pushdown automata and bidirectional parsing. In J. Dassow, G. Rozenberg, and A. Salomaa, editors, *Developments in Language Theory II*, pages 472–481. World Scientific, Singapore, 1996.
8. Stuart M. Shieber, Yves Schabes, and Fernando C. N. Pereira. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24(1–2):3–36, July-August 1995.
9. Klaas Sikkel. *Parsing Schemata — A Framework for Specification and Analysis of Parsing Algorithms.* Texts in Theoretical Computer Science — An EATCS Series. Springer-Verlag, Berlin/Heidelberg/New York, 1997.