Parsing Beyond Context-Free Grammars: Thread Automata

Laura Kallmeyer
Heinrich-Heine-Universität Düsseldorf

Winter 2021/22

Overview

[Kal10]

Idea of Thread Automata (1)

- Thread automata (TA) have been proposed in [Vil02]. See also [Kal10] for a description of TA.
- TA accept (at least) the class of all LCFRLs, maybe even a proper superset.
- TA are non-deterministic and, if all possibilities are pursued independently from each other, they are of exponential complexity.
- However, in combination with a compact representation of sub-derivations as items and with tabulation techniques, they become polynomial.

Idea of Thread Automata (2)

Overall idea of TA:

- We have a set of threads, one of which is the active thread.
- Each thread has a unique path that situates it within the tree-shaped thread structure.
- Whenever a new thread is started, its path is a concatenation of the parent thread path and a new symbol. This way, from a given active thread, we can always find its parent thread (the one that started it) and its daughter threads.
- The moves of the automaton are the following: We can change the content of the active thread, start a new daughter thread or move into an existing daughter thread, or move into the parent thread while, eventually, terminating the active thread.

Idea of Thread Automata (3)

Example: Consider the following simple RCG for $\{a^n b^n e c^n d^n \mid n \geq 0\}$:

$$\begin{array}{ll} S(XeY) \to A(X, Y) & S(e) \to \varepsilon \\ A(aXb, cYd) \to A(X, Y) & A(ab, cd) \to \varepsilon \end{array}$$

In the corresponding TA,

- we would start with an S thread.
- From here, we can read an e and then terminate.
- Or we start a daughter A thread that is suspended once the first component is finished.
- Then we continue the mother S thread, read the e and then resume the daughter.

Idea of Thread Automata (4)

$w = aabbeccdd$ (only successful configurations)

thread set	operation
$[1 : S(\bullet XeY) \to A(X,Y)]$	
$[1 : \ldots], [11 : A(\bullet aXb, cYd) \to A(X,Y)]$	predict
$[1 : \ldots], [11 : A(a \bullet Xb, cYd) \to A(X,Y)]$	scan
$[1 : \ldots], [11 : \ldots], [111 : A(\bullet ab, cd) \to \varepsilon]$	predict
$[1 : \ldots], [11 : \ldots], [111 : A(ab\bullet, cd) \to \varepsilon]$	scan (twice)
$[1 : \ldots], [11 : A(aX \bullet b, cYd) \to A(X,Y)], [111 : \ldots]$	suspend
$[1 : \ldots], [11 : A(aXb\bullet, cYd) \to A(X,Y)], [111 : \ldots]$	scan
$[1 : S(Xe \bullet Y) \to A(X,Y)], [11 : \ldots], [111 : \ldots]$	suspend, scan
$[1 : \ldots], [11 : A(aXb, \bullet cYd) \to A(X,Y)], [111 : \ldots]$	resume
$[1 : \ldots], [11 : A(aXb, c \bullet Yd) \to A(X,Y)], [111 : \ldots]$	scan
$[1 : \ldots], [11 : \ldots], [111 : A(ab, cd\bullet) \to \varepsilon]$	resume, scan (twice)
$[1 : \ldots], [11 : A(aXb, cYd\bullet) \to A(X,Y)]$	suspend, scan
$[1 : S(XeY\bullet) \to A(X,Y)]$	suspend

Idea of Thread Automata (5)

- TA for simple RCGs perform a top-down recognition.
- If an additional tabulation is included, the automaton amounts to an incremental Earley recognition.
- [Vil02] has implemented TA in general.
- [KM09] implements an incremental Earley chart parsing of ordered simple RCG following the same strategy.

TA for Simple RCG (1)

[Vil02] gives a general definition of TA and shows then how to construct equivalent TA for Tree Adjoining Grammars and for simple RCGs.

In the following, we first introduce the specific TA for ordered simple RCGs that we call *SRCG-TA*. Only later, general TA are defined.

We call a simple RCG rule with a dot in the lhs a dotted rule.

TA for Simple RCG (2)

Definition 1 (Thread Automaton for Simple RCG)

A *thread automaton for an ordered simple RCG* $G = \langle N_G, T_G, S_G, P_G \rangle$ (SRCG-TA) is a tuple $\langle N, T, S', ret, \mathcal{U}, \Theta \rangle$ where

- $N = N_G \cup \{S', ret\} \cup \{\gamma \,|\, \gamma$ is a dotted rule in $G\}$, $T = T_G$ are the non-terminals and terminals with $S', ret \in N \setminus N_G$ the start and end symbols;
- $\mathcal{U} = \{1, \dots, m\}$ where m the maximal rhs length in G is the set of labels used to identify threads.
- Θ is a finite set of transitions.

TA for Simple RCG (3)

A configuration is a tree-shaped set of threads, one of them being the active thread, together with a position in the input that separates the part that has been recognized from the remaining part of the input.

Every thread has a thread path $p \in \mathcal{U}$ and its content is a non-terminal symbol.

Definition 2 (Thread, Configuration)

Let $M = \langle N, T, S', ret, \mathcal{U}, \Theta \rangle$ be a SRCG-TA.

- A *thread* is a pair $p : A$ with $p \in \mathcal{U}^*, A \in N$. p is the thread path, and A is the content of the thread.
- A *thread store* is a set of threads whose addresses are closed by prefix.
- A *configuration* of M is a tuple $\langle i, p, \mathcal{S} \rangle$ where i is an input position, $\mathcal{S}$ is a thread store and p is a thread path in $\mathcal{S}$.

TA for Simple RCG (4)

The transitions defined within Θ are the following:

- **Call** starts a new thread, either for the start predicate or for a daughter predicate:

  If active thread $[p : S']$, then add new thread $[p1 : S]$ (where S start symbol of G), set active thread to $p1$.

  $S' \to [S']S$ (initial call),

  If active thread $[p : \gamma]$, γ a dotted rule with the dot (position k, j) preceding the first variable of the rhs non-terminal A and A is the ith rhs element, then add new thread $[pi : A]$ and set active thread to pi.
  $\gamma_{k,j} \to [\gamma_{k,j}]A$ if $\kappa(\gamma_{k,j}) = A$.

- **Predict** predicts a new clause for a non-terminal:
  If active thread $[p : A]$ with $A \in N_G$ and if γ is a dotted A-rule with the dot at the leftmost position, then replace active thread with $[p : \gamma]$.
  $A \to \gamma_{1,0}$ for all A-clauses γ.

TA for Simple RCG (5)

- **Scan** moves the dot over a terminal in the left-hand side while scanning the next input symbol:

  If input position i and active thread $[p : \gamma]$ where γ a dotted rule such that the dot precedes a terminal that is the $(i + 1)$st input symbol, then move dot over this terminal in active thread and increment input position.

  $\gamma_{k,i} \overset{\gamma(k,i+1)}{\rightarrow} \gamma_{k,i+1}$ if $\gamma(k, i + 1)$ is a terminal.

- **Publish** marks the end of a predicate:

  If active thread $[p : \gamma]$ where γ a dotted rule such that the dot is at the end of the lhs, then replace active thread with $[p : \mathit{ret}]$. $\gamma_{k,j} \overset{\varepsilon}{\rightarrow} \mathit{ret}$ where the arity of the left-hand side predicate in γ is k and the kth argument in the left-hand side has length j.

TA for Simple RCG (6)

- **Suspend** suspends a daughter thread and resumes the parent:
  1. If $[pi : ret]$ is the active thread and in its parent thread $[p : \gamma]$ the dot precedes the last variable of the ith rhs element, then remove active thread, move dot over this variable in p thread and set active thread to p.

     $[\gamma_{k,i}]ret \rightarrow \gamma_{k,i+1}$ if $\gamma(k, i+1)$ is a variable that is the last argument of a right-hand side predicate in γ, and
  2. If $[pi : \gamma]$ is the active thread with the dot at the end of the jth lhs component, the jth component is not the last, and if the parent thread is $[p : \beta]$ with the dot preceding the variable that is the jth argument of the ith rhs element, then move the dot over this variable in the p thread and set active thread to p.

     $[\gamma_{k,i}]\beta_{l,j} \rightarrow \gamma_{k,i+1}[\beta_{l,j}]$ if $\gamma(k, i+1)$ is a variable X, β is a B-clause and X is the lth argument of B in the right-hand side of γ but not its last argument, and the lth argument of the left-hand side in β has length j

TA for Simple RCG (7)

- **Resume** resumes an already present daughter thread:

  If active thread is $[p : \gamma]$ where the dot precedes a variable that is the jth ($j > 1$) argument of the ith rhs element and if $pi : \beta]$ is the daughter thread where the dot is at the end of the $(j-1)$th lhs argument, then move the dot to the beginning of the jth lhs argument in the pi thread and set active thread to pi.

  $\gamma_{k,i}[\beta_{l,j}] \rightarrow [\gamma_{k,i}]\beta_{l+1,0}$ if $\gamma(k, i+1)$ is a variable X, β is a B-clause and X is the $(l+1)$th argument of B in the right-hand side of γ, and the lth argument of the left-hand side in β has length j.

TA for Simple RCG (8)

The set of possible configurations $C(M, w)$ for a given input SRCG-TA M and a given input w contains all configurations that are reachable from $\langle 0, \varepsilon, \{[\varepsilon : S']\}\rangle$ via the reflexive transitive closure of the transitions.

We can define the set of possible configurations, based on these operations. For this, we use deduction rules. The initial configuration is the active thread $\varepsilon : S$ with input position 0. The final configuration (i.e., the goal item) has input position $|w| = n$ and contains the thread $u : F$ where $u = \delta(S)$ as active thread and the still present initial thread $\varepsilon : aS$.

The language of a SRCG-TA is the set of words that allow us, starting from the initial thread set $\{\varepsilon : S'\}$, to reach the set $\{\varepsilon : S', 1 : ret\}$ after having scanned the entire input.

Definition 3 (Language)

Let $M = \langle N, T, S', ret, \mathcal{U}, \Theta\rangle$ be a SRCG-TA. The language of M is defined as follows:

$$L(M) = \{w \mid \langle |w|, 1, \{\varepsilon : S', 1 : ret\}\rangle \in C(M, w)\}.$$

Example (1)

Ordered simple RCG with the following rules:

$$\begin{aligned}&\alpha : S(XYZ) \rightarrow A(X, Y, Z)\\&\beta : A(aX, aY, aZ) \rightarrow A(X, Y, Z)\\&\gamma : A(b, b, b) \rightarrow \varepsilon\end{aligned}$$

We encode dotted rules as $r_{i,j}$ where r the name of the rule, $\langle i, j \rangle$ the position of the dot: the dot precedes the j element of the ith argument of the lhs.

Ex.: $\beta_{2,0}$ encodes $A(aX, \bullet aY, aZ) \rightarrow A(X, Y, Z)$

Input $w = ababab$.

Example (2)

thread set	rem. input	
$\varepsilon : S'$	*ababab*	
$\varepsilon : S', 1 : S$	*ababab*	call
$\varepsilon : S', 1 : \alpha_{1,0}$	*ababab*	predict
$\varepsilon : S', 1 : \alpha_{1,0}, 11 : A$	*ababab*	call
$\varepsilon : S', 1 : \alpha_{1,0}, 11 : \beta_{1,0}$	*ababab*	predict
$\varepsilon : S', 1 : \alpha_{1,0}, 11 : \beta_{1,1}$	*babab*	scan
$\varepsilon : S', 1 : \alpha_{1,0}, 11 : \beta_{1,1}, 111 : A$	*babab*	call
$\varepsilon : S', 1 : \alpha_{1,0}, 11 : \beta_{1,1}, 111 : \gamma_{1,0}$	*babab*	predict
$\varepsilon : S', 1 : \alpha_{1,0}, 11 : \beta_{1,1}, 111 : \gamma_{1,1}$	*abab*	scan
$\varepsilon : S', 1 : \alpha_{1,0}, 11 : \beta_{1,2}, 111 : \gamma_{1,1}$	*abab*	suspend
$\varepsilon : S', 1 : \alpha_{1,1}, 11 : \beta_{1,2}, 111 : \gamma_{1,1}$	*abab*	suspend

Example (3)

$\varepsilon : S', 1 : \alpha_{1,1}, 11 : \beta_{2,0}, 111 : \gamma_{1,1}$	$abab$	resume
$\varepsilon : S', 1 : \alpha_{1,1}, 11 : \beta_{2,1}, 111 : \gamma_{1,1}$	bab	scan
$\varepsilon : S', 1 : \alpha_{1,1}, 11 : \beta_{2,1}, 111 : \gamma_{2,0}$	bab	resume
$\varepsilon : S', 1 : \alpha_{1,1}, 11 : \beta_{2,1}, 111 : \gamma_{2,1}$	ab	scan
$\varepsilon : S', 1 : \alpha_{1,1}, 11 : \beta_{2,2}, 111 : \gamma_{2,1}$	ab	suspend
$\varepsilon : S', 1 : \alpha_{1,2}, 11 : \beta_{2,2}, 111 : \gamma_{2,1}$	ab	suspend
$\varepsilon : S', 1 : \alpha_{1,2}, 11 : \beta_{3,0}, 111 : \gamma_{2,1}$	ab	resume
$\varepsilon : S', 1 : \alpha_{1,2}, 11 : \beta_{3,1}, 111 : \gamma_{2,1}$	b	scan
$\varepsilon : S', 1 : \alpha_{1,2}, 11 : \beta_{3,1}, 111 : \gamma_{3,0}$	b	resume
$\varepsilon : S', 1 : \alpha_{1,2}, 11 : \beta_{3,1}, 111 : \gamma_{3,1}$	ε	scan
$\varepsilon : S', 1 : \alpha_{1,2}, 11 : \beta_{3,1}, 111 : ret$	ε	publish
$\varepsilon : S', 1 : \alpha_{1,2}, 11 : \beta_{3,2}$	ε	suspend

Example (4)

$\varepsilon : S', 1 : \alpha_{1,2}, 11 : \mathit{ret}$	ε	publish
$\varepsilon : S', 1 : \alpha_{1,3}$	ε	suspend
$\varepsilon : S', 1 : \mathit{ret}$	ε	publish

Input accepted with the last configuration.

General Definition of TA (1)

We will now give the general definition of TA.

Definition 4 (Thread Automaton)

A *Thread Automaton* is a tuple $\langle N, T, S, F, \kappa, \mathcal{K}, \delta, \mathcal{U}, \Theta \rangle$ where

- N and T are non-terminal and terminal alphabets with $S, F \in N$ the start and end symbols;
- κ, the triggering function, is a partial function from N to some finite set $\mathcal{K}$
- $\mathcal{U}$ is a finite set of labels used to identify threads.
- δ is a partial function from N to $\mathcal{U} \cup \{\perp\}$ used to specify daughter threads that can be created or resumed at some point.
- Θ is a finite set of transitions.

General Definition of TA (2)

In the simple RCG case, κ is not needed, i.e., can just be the identity function.

δ is used to indicate, for the dot preceding a given variable in the lhs, which of the rhs elements contains this variable as an argument. This determines the daughter thread that processes this variable.
Whenever the dot is at the end of a component, δ has value $\perp$, and when preceding a terminal, it is undefined.

Example: Assuming a rule $\alpha : A(XY, aZ) \rightarrow A(X, Z)B(Y)$, we have

$\delta(\alpha_{10}) = \delta(\alpha_{21}) = 1, \delta(\alpha_{11}) = 2,$
$\delta(\alpha_{12}) = \delta(\alpha_{22}) = \perp, \delta(\alpha_{20})$ undefined

General Definition of TA (3)

Definition 5 (TA Transitions)

Let $M = \langle N, T, S, F, \kappa, \mathcal{K}, \delta, \mathcal{U}, \Theta \rangle$ be a TA. All transitions in Θ have one of the following forms:

- $B \overset{\alpha}{\to} C$ with $B, C \in N, \alpha \in T^*$ (**SWAP** operation)
- $b \to [b]C$ with $b \in \mathcal{K}, C \in N$ (**PUSH** operation)
- $[B]C \to D$ with $B, C, D \in N$ (**POP** operation)
- $b[C] \to [b]D$ with $b \in \mathcal{K}, C, D \in N$ (**SPUSH** operation)
- $[B]c \to D[c]$ with $c \in \mathcal{K}, B, D \in N$ (**SPOP** operation)

General Definition of TA (4)

The set of configurations for w, $C(M, w)$, is then defined by the following deduction rules:

- Initial configuration: $\dfrac{}{\langle 0, \varepsilon, \{\varepsilon : S\}\rangle}$
- Swap: $\dfrac{\langle i, p, \mathcal{S} \cup p : B\rangle}{\langle i + |\alpha|, p, \mathcal{S} \cup p : C\rangle}$ $B \overset{\alpha}{\to} C, w_{i+1} \dots w_{i+|\alpha|} = \alpha$
- Push:

$$\frac{\langle i, p, \mathcal{S} \cup p : B\rangle}{\langle i, pu, \mathcal{S} \cup p : B \cup pu : C\rangle} \quad \begin{array}{l} b \to [b]C, \kappa(B) = b, \delta(B) = u, \\ pu \notin dom(\mathcal{S}) \end{array}$$

General Definition of TA (5)

- Pop:

$$\frac{\langle i, pu, \mathcal{S} \cup p : B \cup pu : C\rangle}{\langle i, p, \mathcal{S} \cup p : D\rangle} \quad [B]C \rightarrow D, \delta(C) = \bot, pu \notin dom(\mathcal{S})$$

- Spush:

$$\frac{\langle i, p, \mathcal{S} \cup p : B \cup pu : C\rangle}{\langle i, pu, \mathcal{S} \cup p : B \cup pu : D\rangle} \quad b[C] \rightarrow [b]D, \kappa(B) = b, \delta(B) = u$$

- Spop:

$$\frac{\langle i, pu, \mathcal{S} \cup p : B \cup pu : C\rangle}{\langle i, p, \mathcal{S} \cup p : D \cup pu : C\rangle} \quad [B]c \rightarrow D[c], \kappa(C) = c, \delta(C) = \bot$$

General Definition of TA (6)

The language of a TA is the set of words that allow us, starting from the initial thread set $\{\varepsilon : S\}$, to reach the set $\{\varepsilon : S, \delta(S) : F\}$ after having scanned the entire input.

Definition 6 (Language)

Let $M = \langle N, T, S, F, \kappa, \mathcal{K}, \delta, \mathcal{U}, \Theta\rangle$ be a TA,
The language of M is defined as follows:

$$L(M) = \{w \mid \langle n, \delta(S), \{\varepsilon : S, \delta(S) : F\}\rangle \in C(M, w)\}.$$

General Definition of TA (7)

Take again the TA equivalent to the simple RCG with the following clauses:

$\alpha : S(XYZ) \rightarrow A(X, Y, Z)$
$\beta : A(aX, aY, aZ) \rightarrow A(X, Y, Z)$
$\gamma : A(b, b, b) \rightarrow \varepsilon$

Transitions of the corresponding TA (start symbol S'):

Call:	$S' \rightarrow [S']S$	$\alpha_{1,0} \rightarrow [\alpha_{1,0}]A$	$\beta_{1,1} \rightarrow [\beta_{1,1}]A$
Predict:	$S \rightarrow \alpha_{1,0}$	$A \rightarrow \beta_{1,0}$	$A \rightarrow \gamma_{1,0}$
Scan:	$\beta_{1,0} \xrightarrow{a} \beta_{1,1}$	$\beta_{2,0} \xrightarrow{a} \beta_{2,1}$	$\beta_{3,0} \xrightarrow{a} \beta_{3,1}$
	$\gamma_{1,0} \xrightarrow{b} \gamma_{1,1}$	$\gamma_{2,0} \xrightarrow{b} \gamma_{2,1}$	$\gamma_{3,0} \xrightarrow{b} \gamma_{3,1}$

General Definition of TA (8)

Suspend:

$[\alpha_{1,0}]\beta_{1,2} \to \alpha_{1,1}[\beta_{1,2}]$	$[\alpha_{1,1}]\beta_{2,2} \to \alpha_{1,2}[\beta_{2,2}]$	$[\alpha_{1,2}]\mathit{ret} \to \alpha_{1,3}$
$[\alpha_{1,0}]\gamma_{1,1} \to \alpha_{1,1}[\gamma_{1,1}]$	$[\alpha_{1,1}]\gamma_{2,1} \to \alpha_{1,2}[\gamma_{2,1}]$	
$[\beta_{1,1}]\beta_{1,2} \to \beta_{1,2}[\beta_{1,2}]$	$[\beta_{2,1}]\beta_{2,2} \to \beta_{2,2}[\beta_{2,2}]$	$[\beta_{3,1}]\mathit{ret} \to \beta_{3,2}$
$[\beta_{1,1}]\gamma_{1,1} \to \beta_{1,2}[\gamma_{1,1}]$	$[\beta_{2,1}]\gamma_{2,1} \to \beta_{2,2}[\gamma_{2,1}]$	

Resume:

$\alpha_{1,1}[\beta_{1,2}] \to [\alpha_{1,1}]\beta_{2,0}$	$\beta_{2,1}[\beta_{1,2}] \to [\beta_{2,1}]\beta_{2,0}$
$\alpha_{1,1}[\gamma_{1,0}] \to [\alpha_{1,1}]\gamma_{2,0}$	$\beta_{2,1}[\gamma_{1,0}] \to [\beta_{2,1}]\gamma_{2,0}$
$\alpha_{1,2}[\beta_{2,2}] \to [\alpha_{1,2}]\beta_{3,0}$	$\beta_{2,1}[\beta_{2,2}] \to [\beta_{2,1}]\beta_{3,0}$
$\alpha_{1,2}[\gamma_{2,0}] \to [\alpha_{1,2}]\gamma_{3,0}$	$\beta_{3,1}[\gamma_{2,0}] \to [\beta_{3,1}]\gamma_{3,0}$

Publish:

$\alpha_{1,3} \to \mathit{ret}$	$\beta_{3,2} \to \mathit{ret}$	$\gamma_{3,1} \to \mathit{ret}$

TA for TAG (1)

- We use the position left/right above/below (depicted with a dot) that we know from the TAG Earley parsing.
- Whenever, in the active thread, we are left above a possible adjunction site, we can predict an adjunction by starting a sub-thread (PUSH).
- When reaching the position left above a foot node, we can suspend the thread and resume the parent (SPOP).
- Whenever we arrive right below an adjunction site, we can resume the daughter of the adjoined tree whose content is the foot node (SPUSH).
- Whenever we arrive right above the root of an auxiliary tree, we do a POP, i.e., finish this thread and resume the parent.

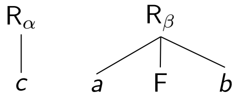
TA for TAG (2)

- We use a special symbol *ret* to mark the fact that we have completely traversed the elementary tree and we can therefore finish this thread.
- Besides moving this way from one elementary tree to another, we can move down, move left and move up inside a single elementary tree (while eventually scanning a terminal) using the SWAP operation.

TA for TAG (3)

Example:

Elementary trees:



(R_α and R_β allow for adjunction of β.)

TA for TAG (4)

Sample thread set of corresponding TA for input *aacbb*:

thread set	operation
$[1 : {}^\bullet R_\alpha]$	
$[1 : {}^\bullet R_\alpha], [11 : {}^\bullet R_\beta]$	PUSH
$[1 : {}^\bullet R_\alpha], [11 : {}^\bullet R_\beta], [111 : {}^\bullet R_\beta]$	PUSH
$[1 : {}^\bullet R_\alpha], [11 : {}^\bullet R_\beta], [111 : {}_\bullet R_\beta]$	SWAP
$[1 : {}^\bullet R_\alpha], [11 : {}^\bullet R_\beta], [111 : {}^\bullet a]$	SWAP
$[1 : {}^\bullet R_\alpha], [11 : {}^\bullet R_\beta], [111 : a^\bullet]$	SWAP (scan *a*)
$[1 : {}^\bullet R_\alpha], [11 : {}^\bullet R_\beta], [111 : {}^\bullet F]$	SWAP
$[1 : {}^\bullet R_\alpha], [11 : {}_\bullet R_\beta], [111 : {}^\bullet F]$	SPOP
$[1 : {}^\bullet R_\alpha], [11 : {}^\bullet a], [111 : {}^\bullet F]$	SWAP
$[1 : {}^\bullet R_\alpha], [11 : a^\bullet], [111 : {}^\bullet F]$	SWAP (scan *a*)
$[1 : {}^\bullet R_\alpha], [11 : {}^\bullet F], [111 : {}^\bullet F]$	SWAP

TA for TAG (5)

$[1 : {}_\bullet R_\alpha], [11 : {}^\bullet F], [111 : {}^\bullet F]$	SPOP
$[1 : {}^\bullet c], [11 : {}^\bullet F], [111 : {}^\bullet F]$	SWAP
$[1 : c^\bullet], [11 : {}^\bullet F], [111 : {}^\bullet F]$	SWAP (scan c)
$[1 : R_{\alpha\bullet}], [11 : {}^\bullet F], [111 : {}^\bullet F]$	SWAP
$[1 : R_{\alpha\bullet}], [11 : F^\bullet], [111 : {}^\bullet F]$	SPUSH
$[1 : R_{\alpha\bullet}], [11 : {}^\bullet b], [111 : {}^\bullet F]$	SWAP
$[1 : R_{\alpha\bullet}], [11 : b^\bullet], [111 : {}^\bullet F]$	SWAP (scan b)
$[1 : R_{\alpha\bullet}], [11 : R_{\beta\bullet}], [111 : {}^\bullet F]$	SWAP
$[1 : R_{\alpha\bullet}], [11 : R_{\beta\bullet}], [111 : F^\bullet]$	SPUSH
$[1 : R_{\alpha\bullet}], [11 : R_{\beta\bullet}], [111 : {}^\bullet b]$	SWAP
$[1 : R_{\alpha\bullet}], [11 : R_{\beta\bullet}], [111 : b^\bullet]$	SWAP (scan b)

TA for TAG (6)

$[1 : R_{\alpha\bullet}], [11 : R_{\beta\bullet}], [111 : R_{\beta\bullet}]$	SWAP
$[1 : R_{\alpha\bullet}], [11 : R_{\beta\bullet}], [111 : R_{\beta}{}^{\bullet}]$	SWAP
$[1 : R_{\alpha\bullet}], [11 : R_{\beta\bullet}], [111 : ret]$	SWAP
$[1 : R_{\alpha\bullet}], [11 : R_{\beta}{}^{\bullet}]$	POP
$[1 : R_{\alpha\bullet}], [11 : ret]$	SWAP
$[1 : R_{\alpha}{}^{\bullet}]$	POP
$[1 : ret]$	SWAP

TA for TAG (7)

TA for our sample TAG:

$M = \langle N, T, S, ret, \kappa, \mathcal{K}, \delta, \mathcal{U}, \Theta \rangle$ is as follows:

- N contains all symbols ${}^\bullet X, {}_\bullet X, X_\bullet, X^\bullet$ where X is a node in one of the elementary trees, i.e., $X \in \{R_\alpha, c, R_\beta, a, F, b\}$. Furthermore, N contains a special symbol *ret* and a special symbol S.
- $T = \{a, b, c\}$.
- S is the initial thread symbol and *ret* is the final thread symbol.
- $\mathcal{K} = N, \kappa(A) = A$ for all $A \in N$.
- $\mathcal{U} = \{1\}$, $\delta(X) = 1$ for all $A \in N \setminus \{{}^\bullet F, ret\}, \delta(ret) = \bot, \delta({}^\bullet F) = \bot$.

TA for TAG (8)

- Transitions Θ:

$S \to [S]^\bullet R_\alpha$	start initial tree
${}^\bullet R_\alpha \to {}_\bullet R_\alpha$, ${}^\bullet R_\beta \to {}_\bullet R_\beta$	predict no adjunction
${}_\bullet R_\alpha \to {}^\bullet c$, ${}_\bullet R_\beta \to {}^\bullet a$	move down
${}^\bullet c \xrightarrow{c} c^\bullet$, ${}^\bullet a \xrightarrow{a} a^\bullet$, ${}^\bullet b \xrightarrow{b} b^\bullet$	scan
$a^\bullet \to {}^\bullet F$, $F^\bullet \to {}^\bullet b$	move right
$c^\bullet \to R_{\alpha\bullet}$, $b^\bullet \to R_{\beta\bullet}$	move up
$R_{\alpha\bullet} \to R_\alpha{}^\bullet$, $R_{\beta\bullet} \to R_\beta{}^\bullet$	move up if no adjunction
${}^\bullet R_\alpha \to [{}^\bullet R_\alpha]^\bullet R_\beta$, ${}^\bullet R_\beta \to [{}^\bullet R_\beta]^\bullet R_\beta$	predict adjoined tree
$[{}^\bullet R_\alpha]^\bullet F \to {}_\bullet R_\alpha[{}^\bullet F]$, $[{}^\bullet R_\beta]^\bullet F \to {}_\bullet R_\beta[{}^\bullet F]$	back to adjunction site
$R_{\alpha\bullet}[{}^\bullet F] \to [R_{\alpha\bullet}]F^\bullet$, $R_{\beta\bullet}[{}^\bullet F] \to [R_{\beta\bullet}]F^\bullet$	resume adjoined tree
$R_\alpha{}^\bullet \to ret$, $R_\beta{}^\bullet \to ret$	complete elementary tree
$[R_{\alpha\bullet}]ret \to R_\alpha{}^\bullet$, $[R_{\beta\bullet}]ret \to R_\beta{}^\bullet$	terminate adjunction, go back

References I

[Kal10] Laura Kallmeyer.
Parsing Beyond Context-Free Grammars.
Cognitive Technologies. Springer, Heidelberg, 2010.

[KM09] Laura Kallmeyer and Wolfgang Maier.
An incremental Earley parser for simple Range Concatenation Grammar.
In *Proceedings of IWPT 2009*, 2009.

[Vil02] Éric Villemonte de La Clergerie.
Parsing mildly context-sensitive languages with thread automata.
In *Proc. of COLING'02*, August 2002.