Análisis Sintáctico de TAGs usando Analizadores Deductivos

Díaz Madrigal, Víctor J. Carrillo Montero, Vicente Toro Bonilla, Miguel
Facultad de Informática y Estadística (Universidad de Sevilla)
Avda. Reina Mercedes s/n 41012-SEVILLA (SPAIN)
E-mai: {vjdiaz,carrillo}@lsi.us.es

Resumen

La definición de analizadores sintácticos utilizando sistemas deductivos (analizadores deductivos), supone un enfoque con un grado importante de abstracción que beneficia el estudio comparado y el prototipado rápido de distintos analizadores. Las gramáticas TAG (Tree Adjoining Grammar) es un formalismo gramatical que presenta ventajas lingüísticas importantes, pero en el plano teórico también presenta considerables costes computacionales. Creemos que el estudio de los analizadores para TAGs desde la perspectiva de los analizadores deductivos, al formalizar- y unificar propuestas de diverso origen, puede aportar nuevas ideas que mejoren dichos costes, al menos en situaciones prácticas. En este trabajo se hace una comparativa de dos analizadores para TAGs. Consideramos que los dos analizadores escogidos, uno debido a Schabes y el otro a Nederhof, son de especial relevancia por sus propiedades computacionales. La descripción original propuesta por Nederhof es un analizador deductivo, a diferencia de la propuesta de Schabes. Por tanto, antes de abordar el estudio comparativo, se presenta la adaptación a reglas deductivas del reconocedor de Schabes. Las transformaciones realízadas en este último se ajustan al enfoque utilizado por Nederhof para definir su reconocedor, lo cual permite que los resultados obtenidos por el estudio sean homogéneos.

1. Introducción

Las gramáticas TAG (Tree Adjoining Grammar) [Sch90] presentan un poder expresivo mayor que las gramáticas incontextuales CFG (Context Free Grammar) dentro de la jerarquía de Chomsky. Uno de sus principales atractivos es su fuerte lexicalización, ya que permite diluir la barrera entre gramática y lexicón. Esta caracteristicn está en concordancia con las nuevas tendencias lingüísticas que incorporan cada vez mayor información gramatical dentro del lexicón. Sin embargo, las gramáticas TAG presentan reconocedores sintácticos cuya eficiencia teórica en el peor de los casos - $O(n^6)$ siendo n la longitud de la cadena de entrada [Sat94]- es superior respecto a la obtenida en las gramáticas CFGs ($O(n^3)$).

Una TAG es una tupla (Σ, N, S, I, A) donde Σ es el conjunto de terminales, N el conjunto de no terminales, S$\in$N el axioma, I el conjunto de árboles iniciales y A el conjunto de árboles auxiliares. Los árboles en el conjunto I $\cup$ A se denominan elementales y sus nodos interiores están etiquetados con símbolos no terminales. La raíz de los árboles iniciales debe además ser el axioma. Los nodos frontera deben ser terminales o la palabra vacía, salvo uno en los árboles auxiliares (denominado nodo pie) que debe estar etiquetado con el mismo no terminal que la raíz. El camino que va desde la raíz al nodo pie de un árbol auxiliar se denomina espina.

La operación de composición en las TAGs es la adjunción, que consiste en introducir un árbol auxiliar en un nodo no terminal etiquetado igual que la raíz del árbol auxiliar. Al introducir dicho árbol auxiliar, el árbol original es partido en dos subárboles respecto al nodo donde se ha producido la adjunción. El lenguaje de una gramática TAG es justamente la cadena correspondiente a la frontera de todo árbol inicial o derivado a partir de un árbol inicial o derivado a partir de un árbol inicial mediante posibles adjunciones.

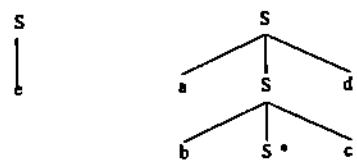

Figura 1. Ejemplo de gramática TAG

Este trabajo presenta un estudio comparativo

de dos analizadores para TAGs que consideramos especialmente representativos: uno debido a Schabes [Sch90] y otro a Nederhof [Ned97]. Ambos presentan características comunes: la entrada es leída de izquierda a derecha, cumplen la propiedad de prefijo válido, los costes temporales se ven reducidos en el caso promedio y ambos están basados en algoritmos Chart. Sin embargo, presentan una diferencia teórica: el primero tiene unos costes temporales de $O(n^9)$ y el segundo de $O(n^6)$ en el peor de los casos. Este estudio va encaminado a mostrar que, aunque existe una diferencia teórica, el comportamiento medio no presenta una diferencia tan significativa.

El análisis comparativo se hará según el enfoque *Parsing as Deduction* [SSP95], ya que consideramos que es un modelo muy apropiado para abordar el estudio de analizadores. Según este modelo, un reconocedor se define mediante un conjunto de reglas deductivas (analizador deductivo). Esta clase de sistemas supone una descripción clara y precisa del analizador, que elimina detalles de implementación como el control o las estructuras de datos involucradas. Este nivel de abstracción facilita la comparación de distintos analizadores y el prototipado rápido usando programación lógica.

La notación básica utilizada para representar una regla de inferencia será

$$N \quad \frac{A_1 \ldots A_k}{B} \quad P(A_1, \ldots, A_k, B)$$

donde N es el nombre de la regla, A_i (antecedentes) y B (consecuente) son fórmulas esquemáticas que pueden incluir metavariables sintácticas instanciables en términos cuando la regla es utilizada. Las condiciones laterales $P(A_1,...A_k,B)$ pueden referirse a reglas de producción de la gramática de entrada. Las fórmulas permitirán representar derivaciones parciales correctas (ítems) o referenciar posiciones concretas de la cadena de entrada. Las fórmulas objetivo establecerán cuando una cadena es gramatical respecto a una gramática y cadena de entrada. Por tanto, analizar una cadena de entrada es equivalente a encontrar una derivación que alcance una fórmula objetivo.

2. *Un Analizador Deductivo basado en la Propuesta de Schabes*

Aunque Nederhof presenta su reconocedor mediante un analizador deductivo, éste no es el caso de la propuesta de Schabes. Este autor presenta un algoritmo basado en técnicas de programación dinámica que utiliza la noción de árbol punteado como base para la descripción de los estados.

Un árbol punteado es un árbol elemental que contiene un único nodo punteado. Un nodo punteado es un símbolo con un punto situado en una de cuatro posiciones: left-above (la) $^{\bullet}A$, left-below (lb) $.A$, right-above (ra) $A^{\bullet}$ y right-below (rb) $A.$. El autor define un recorrido a través de los nodos de un árbol elemental (figura 2) de forma que puedan ser capturadas todas las posibles adjunciones en un árbol elemental.

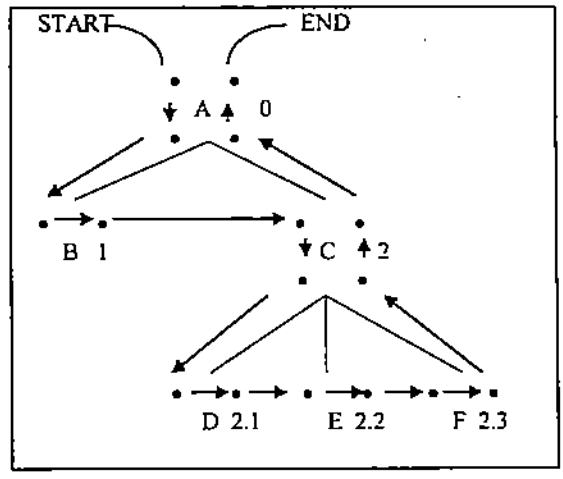

Figura 2. Recorrido de árbol elemental

Supongamos una gramática TAG y una cadena de entrada con longitud n, el reconocedor de Schabes viene descrito mediante estados de la forma (figura 3)

[γ, dot, pos, l, fl, fr, star, tl, bl]

donde γ es el nombre de un árbol elemental, dot la dirección de un nodo en el árbol γ, pos un elemento en el conjunto {la,lb,ra,rb}, star una dirección en γ, y los índices l, fl, fr, tl, bl números naturales que varían en el rango 0..n.

El algoritmo define n+1 conjuntos de estados inicialmente vacíos, y haciendo uso de un grupo de procesos, los amplia iterativamente. Dadas las características del algoritmo, éste puede ser adaptado para obtener un analizador deductivo ya que los procesos y los estados pueden ser considerados reglas deductivas e ítems respectivamente.

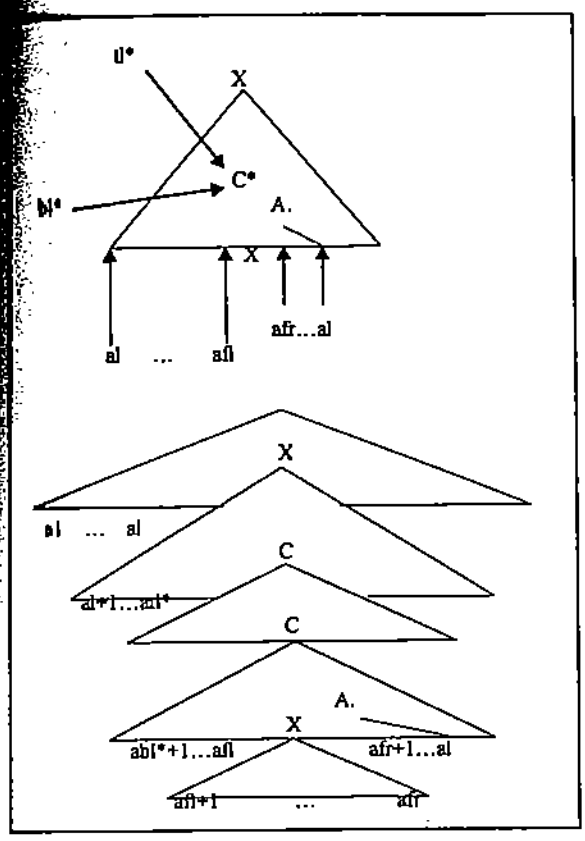

Figura 3. Significado de los índices en Schabes

Sin embargo, la información incluida en los estados del algoritmo de Schabes requiere ser revisada con mayor atención al ser transformada en ítems, si deseamos que la comparación entre los dos analizadores sea más exacta. Esta revisión va dirigida a traducir el concepto de árbol punteado de Schabes al de regla punteada de Nederhof.

Los ítems del analizador deductivo basado en el reconocedor de Schabes tendrán los siguientes elementos:

[R, i, l, fl, fr, star, tl, bl]

de forma que mantenemos los componentes l, fl, fr, tl, bl y star presentes en los estados. Desaparecen los componentes γ, dot y pos que serán sustituidos por el concepto de regla punteada generalizada R que explicaremos posteriormente. Y añadimos un índice adicional i, variando entre [0,n], que hará innecesario el concepto de conjunto de estado ya que quedará inmerso dentro del propio estado.

Pasemos ahora a detallar como transformar la noción de árbol punteado al de regla punteada generalizada. Primero adoptaremos una representación de los árboles elementales semejante a la utilizada por Nederhof en su analizador deductivo, pero adaptada a las necesidades presentes en la descripción del algoritmo de Schabes. Cada árbol elemental es asociado con un conjunto de producciones, de forma que a cada nodo interior M, cuyos hijos son la secuencia ordenada $M_1 \ldots M_n$, asociaremos la regla CFG

$$M \rightarrow M_1 \ldots M_n$$

Así, podemos asociar a una gramática TAG un conjunto de reglas CFG [DCT98] resultado de la unión de todas las reglas CFG obtenidas a partir de cada árbol elemental.

Para cada nodo hoja M, salvo en el caso del nodo pie, se define Etiq(M) que representa la etiqueta asociada al nodo M. Esta etiqueta pertenecerá al conjunto Σ o será la palabra vacía ε. Para cada nodo M, etiquetado con un no terminal, se define el predicado Adj(M,t) que indica si el árbol auxiliar t puede ser adjuntado en M. Asimismo, para cada nodo interior se define Hijos(M) como la secuencia ordenada de sus nodos hijos. Pie(M) será un predicado que nos dice si M es el nodo pie de un árbol auxiliar. Estas expresiones podrán participar en las condiciones laterales de las reglas deductivas.

Para referirnos a la raíz de un árbol elemental t usaremos la notación R_t, y para referirnos al nodo pie de un árbol auxiliar t utilizaremos la notación F_t. Por razones técnicas se incluye un nodo más para cada árbol elemental t que denotaremos mediante Π. Este nodo solo tendrá un hijo que será justamente la raíz del árbol inicial R_t.

Los puntos en las reglas punteadas deben, por un lado, dividir la parte derecha en dos mitades que indiquen que fragmento de regla ha sido derivado y, por otro, tendrán que dar cuenta de la posición (la,lb,ra,rb) en la que está situado el punto. De esta forma, las reglas punteadas clásicas que presentan la siguiente notación

$$A \rightarrow \alpha \bullet \beta \qquad A \in N, \alpha,\beta \in (\Sigma \cup N)^*$$

no cumplen la segunda de las necesidades. Si generalizamos dicha notación mediante

$$A \rightarrow \alpha\ (la)\ \beta \qquad A \in N, \alpha,\beta \in (\Sigma \cup N)^*$$

vemos que la información sobre la situación de los puntos es alcanzada al mismo tiempo que la

división de la regla en parte explorada y por explorar. Considerando $Y \in (\Sigma \cup N \cup \{\varepsilon\})$ la completa exploración de la regla según la primera notación sería

$$A \rightarrow \gamma Y \bullet \qquad A \in N, \gamma \in (\Sigma \cup N)^*$$

Sin embargo, usando el recorrido definido por Schabes a través de los árboles elementales, esta situación vendría reflejada mediante

$$A ::= \gamma\ (ra) Y \qquad A \in N, \gamma \in (\Sigma \cup N)^*$$

Con estas consideraciones, la transformación de cada uno de los procesos en reglas deductivas se realiza del siguiente modo: cada proceso genera un nuevo ítem (consecuente) siempre que existan previamente un conjunto de ítems (antecedentes) y que se cumplan unas restricciones (condiciones laterales). El ítem objetivo será un ítem que alcance la posición ra para la raíz de algún árbol inicial t:

$[\Pi \rightarrow (ra)\ R_t, n, 0, -, -, -, -, -]$ para algún $t \in I$

La regla INIC es el axioma (figura 4), el cual introduce un ítem para cada árbol inicial t empezando en la raíz. SC1 y SC2 suponen la confirmación de las hojas terminales de un árbol respecto a la entrada. MDD baja un nivel en la exploración de un árbol elemental. MDU1 y MDU2 conducen la exploración hacia los hermanos inmediatos o hacia el padre inmediato de un nodo no terminal. LP1, LP21 y LP22 tratan la adjunción o no adjunción en un no terminal. En las reglas LC1 y LC2 se ha alcanzado el nodo pie de un árbol auxiliar, y se pretende seguir la exploración a través del subárbol partido por la adjunción. RP1 y RP2 tratan de terminar la exploración del árbol auxiliar y en la regla RC se reconduce la exploración del árbol donde se ha producido una adjunción, una vez que el árbol auxiliar ha sido totalmente explorado.

INIC	$\dfrac{}{[T \rightarrow (la)\ R_t, 0, 0, -, -, -, -, -]}$	$t \in I$
SC1	$\dfrac{[T \rightarrow \alpha\ (la)\ M\beta, i, l, fl, fr, st, tl, bl]}{[T \rightarrow \alpha M\ (ra)\ \beta, i+1, l, fl, fr, st, tl, bl]}$	$Etiq(M) = a_{i+1}$
SC2	$\dfrac{[T \rightarrow \alpha\ (la)\ M\beta, i, l, fl, fr, st, tl, bl]}{[T \rightarrow \alpha M\ (ra)\ \beta, i, l, fl, fr, st, tl, bl]}$	$Etiq(M) = \varepsilon$
MDD	$\dfrac{[N \rightarrow \alpha (lb)\ M\beta, i, l, fl, fr, st, tl, bl]}{[M \rightarrow (la)\ \gamma, i, l, fl, fr, st, tl, bl]}$	$Hijos(M) = \gamma$
MDU1	$\dfrac{[N \rightarrow \alpha\ (ra)\ XY\beta, i, l, fl, fr, st, tl, bl]}{[N \rightarrow \alpha X\ (la)\ Y\beta, i, l, fl, fr, st, tl, bl]}$	$X \in (N \cup \Sigma)$
MDU2	$\dfrac{[M \rightarrow \gamma\ (ra)\ X, i, l, fl, fr, st, tl, bl]}{[N \rightarrow \alpha\ (rb)\ M\beta, i, l, fl, fr, st, tl, bl]}$	$Hijos(M) = \gamma X$
LP1	$\dfrac{[N \rightarrow \alpha\ (la)\ M\beta, i, l, fl, fr, st, tl, bl]}{[\Pi \rightarrow (la)\ R_t, i, i, -, -, -, -, -]}$	$\forall t \in A \wedge Adj(M,t)$
LP21	$\dfrac{[N \rightarrow \alpha\ (la)\ M\beta, i, l, fl, fr, st, tl, bl]}{[N \rightarrow \alpha\ (lb)\ M\beta, i, l, fl, fr, st, tl, bl]}$	$\neg\ Pie(M)$
LP22	$\dfrac{[N \rightarrow \alpha\ (la)\ Ft\beta, i, l, -, -, st, tl, bl]}{[N \rightarrow \alpha\ (lb)\ Ft\beta, i, l, i, -, st, tl, bl]}$	
LC1	$\dfrac{[N \rightarrow \alpha\ (lb)\ Ft\beta, i, l, i, -, st, tl, bl]\ \ [N' \rightarrow \alpha'\ (la)\ Ft'\beta', l, l', -, -, st', tl', bl']}{[N' \rightarrow \alpha'\ (lb)\ Ft'\beta', i, l', i, -, Ft', l, i]}$	$Adj(Ft', t)$
LC2	$\dfrac{[N \rightarrow \alpha\ (lb)\ Ft\beta, i, l, i, -, st, tl, bl]\ \ [N' \rightarrow \alpha'\ (la)\ M\beta', l, l', fl', fr', st', tl', bl']}{[N' \rightarrow \alpha'\ (lb)\ M\beta', i, l', fl', fr', M, l, i]}$	$\neg Pie(M) \wedge Adj(M,t)$
RP1	$\dfrac{[N \rightarrow \alpha\ (rb)\ M\beta, i, l, fl, fr, M, tl, bl]\ \ [N' \rightarrow \alpha'\ (lb)\ Ft\beta', bl, tl, bl, -, st', tl', bl']}{[N' \rightarrow \alpha'\ (rb)\ Ft'\beta', i, tl, bl, i, st', tl', bl']}$	$Adj(M,t)$
RP2	$\dfrac{[N \rightarrow \alpha\ (rb)\ M\beta, i, l, fl, fr, st, tl, bl]}{[N \rightarrow \alpha\ (ra)\ M\beta, i, l, fl, fr, st, tl, bl]}$	$M \neq st \wedge Adj(M,t)$
RC	$\dfrac{[\Pi \rightarrow (ra)\ R_t, i, l, fl, fr, -, -, -]\ \ [N \rightarrow \alpha\ (la)\ M\beta, l, l', fl', fr', st', tl', bl']\ \ [N \rightarrow \alpha\ (rb)\ M\beta, fr, l', fl', fr', st'', l, fl]}{[N \rightarrow \alpha\ (ra)\ M\beta, i, l', fl', fr', st', tl', bl']}$	$t \in A \wedge Adj(M,t)$

Figura 4. Analizador deductivo de Schabes

3. *Estudio Comparativo y Conclusiones*

Una de las aportaciones del modelo *Parsing as Deduction* es la existencia de un metaintérprete para programas lógicos obtenidos a partir de sistemas deductivos gramaticales. En [SSP95] se demuestra que dicho intérprete es completo y consistente.

La gramática que se suministra al metaintérprete se basa en la notación DCG, pero teniendo en cuenta que un árbol elemental se asocia con un conjunto de reglas DCG. La gramática para el ejemplo de la figura 1 será

```
ini(a,s) :- [s(a,0)].
s(a,0 ) :- [e].
aux(b,s) :- [s(b,1)].
s(b,1) :- [á,s(b,2),d].
s(b,2) :- [b,s(b,ft),c].
```

donde los símbolos no terminales etiquetados con nt en el árbol g en la posición p es representado mediante el término prolog

nt(g,p). En caso de ser el nodo pie, la posición p es representada mediante el átomo ft. Los símbolos añadidos Π son representados mediante el functor ini (aux) si el árbol es inicial (auxiliar). Los terminales son representados directamente mediante átomos.

El sistema de deducción se basa en la definición de item. Para ello la tupla de valores correspondiente a un ítem debe ser transformada en un término prolog item. La naturaleza de este término dependerá del analizador en cuestión. En nuestro caso los valores naturales (i,l,fl,fr,tl,bl) vendrán representados mediante naturales (un valor no instanciado se hará equivalente al valor negativo –1). La regla punteada R → α(la) Mβ es representada mediante cuatro parámetros: uno incluye la cabeza de la regla R, dos listas PROLOG Alpha y [M| Beta] y un parámetro que tome los valores la,lb,ra,rb. Los valores star serán términos que representan nodos, por ejemplo M..

Una vez definida la estructura de los ítems, debemos definir el sistema deductivo mediante un programa PROLOG que incluya la definición de los ítems iniciales (initial_item/1), finales (final_item/1) y las reglas (inference/4). Este último predicado incluirá el nombre de la regla, una lista de ítems con los antecedentes, un ítem consecuente y una lista con predicados PROLOG que reflejen las condiciones laterales.
Las condiciones laterales son de fácil implementación usando predicados y no entraremos en detalles. Ejemplos de reglas de inferencias correspondientes a SCAN2 y LP1 serían:

```
inference(scan2,
[item(N,Alpha,[M|Beta],la,I,L,FL,FR,ST,TL,BL)],
item(N,Alpha,[M|Beta],ra,I,L,FL,FR,ST,TL,BL),
[empty_word(M)]).

inference(lp1,
[item(N,Alpha,[M|Beta],la,I,L,FL,FR,ST,TL,BL)],
item(top(Aux,Cat),[],[R],la,I,I,-1,-1,-1,-1,-1),
[M=..[Cat,Tree,Dot],(top(Aux,Cat) --->
[R]),adj(M,Aux)]).
```

Para mostrar los resultados obtenidos por el estudio comparativo, hemos escogido seis gramáticas TAGs, G_1 a G_6, que presentan características que consideramos de especial relevancia (recursivas izquierda y derecha, con adjunciones sólo posibles en la espina o fuera de ella, y gramáticas que combinan estas características). Para cada gramática hemos realizado un conjunto de seis pruebas, E_1 a E_6, que combinan el aumento de la longitud de la entrada y la complejidad de las adjunciones. Dada la descripción deductiva de ambos analizadores y la naturaleza semejante de los ítems involucrados en su definición, una buena medida de comparación es el número de ítems generados. Los resultados que hemos obtenido pueden verse en la tabla. De los resultados podemos concluir que tanto la gramática de entrada como las características de la entrada pueden influir en el comportamiento de ambos analizadores, y que esta influencia no siempre perjudica al analizador de Schabes, a pesar de que su eficiencia teórica es peor. Efectivamente la gramática 2 (recursiva derecha pura) beneficia al reconocedor de Schabes . Aún más, la descripción que mostramos hace uso de las reglas MDD, MDU1 y MDU2 que generan ítems extra no significativos ya que se limitan prácticamente a situar el punto de forma oportuna a través de los árboles elementales. De hecho, la función de estas dos reglas podría ser absorbida por las demás reglas dando lugar a un número menor de ítems.

		E1	E2	E3	E4	E5	E6
G1	N	5	23	41	58	77	77
	S	8	32	56	56	80	104
G2	N	5	16	36	66	107	160
	S	8	19	30	41	52	63
G3	N	8	16	29	47	70	98
	S	22	34	47	61	76	92
G4	N	9	27	50	85	127	242
	S	15	42	80	135	201	384
G5	N	6	32	84	58	58	176
	S	10	54	147	98	98	392
G6	N	18	33	53	56	80	82
	S	63	99	143	140	192	197

Figura 5. Comparativa entre N(ederhof) y S(chabes)

Considerando estas circunstancias, podemos concluir que ambos reconocedores siguen siendo alternativas interesantes, a pesar de que su eficiencia teórica no sea equivalente. No obstante, queda hacer un estudio más detallado que haga uso de gramáticas con un número de árboles importante, y una comparativa formal de las estrategias utilizadas por ambos.

Referencias

[DCT98] Díaz, V; Carrillo, V; Toro, M. Elementary Tree Representation. Workshop on Tabulation in Parsing and Deduction (TAPD'98) pp 10-15. Paris (France) 1998.

[Ned97] Nederhof, M. Solving the Correct-Prefix Property for TAGs. 5th Meeting of Mathematics of Language. Schloss Dagstuhl (Germany) 1997. Accepted for publication in Computational Linguistics Journal.

[Sat94] Satta, G. Tree-adjoining Grammar Parsing and Boolean Matrix Multiplication. Computational Linguistics, 20(2) pp 173-191, 1994.

[Sch90] Schabes, Y. Mathematical and Computational Aspects of Lexicalized Grammars. Ph. D. University of Pennsylvania, Philadelphia, 1990.

[SSP95] Shieber, S; Schabes, Y; Pereira, F. Principles and Implementation of Deductive Parsing. Journal of Logic and Computation, 24(1&2) pp 3-36, 1995.