Ling 5801: Lecture Notes 8
From CFGs to Pushdown Automata

We can recognize CFGs using ***pushdown automata***.

Contents

8.1 PDAs are FSAs extended with an infinite pushdown store at each state

A Pushdown Automaton (PDA) is a tuple $\langle Q, X, S, F, M \rangle$, where:

- Q is a finite set of states
- X is a finite set of observation symbols
- $S \subseteq Q$ is a set of start states
- $F \subseteq Q$ is a set of final states
- $M \subseteq Q \times (Q \cup \{\epsilon\}) \times (X \cup \{\epsilon\}) \times Q \times (Q \cup \{\epsilon\})$ is a set of store,state transitions, of form:
  - $\langle q, \epsilon, \epsilon, q', q \rangle$ — called an 'expansion' (or 'stack push'):

    transition from state q to q', replacing empty string ϵ at front of store with q
  - $\langle q, q'', x, q', q'' \rangle$ — called a 'state transition':

    transition from state q to q' on observation x, replacing q'' at front of store with q''
  - $\langle q, q'', \epsilon, q', \epsilon \rangle$ — called a 'reduction' (or 'stack pop/pull'):

    transition from state $q \in F$ to q', replacing q'' at front of store with empty string ϵ

From 'Candyland' to 'Trivial Pursuit': pieces carry stacked-on sub-pieces corresp. to spaces

8.2 Language accepted by PDA $A = \langle Q_A, X_A, S_A, F_A, M_A \rangle$

The automaton nondeterminisitically explores all possible states combined with all possible stores:

- The automaton starts at a start state with an empty store.
- The automaton pushes and pulls only to/from the front of the store.
- The automaton accepts the input on an empty store at any final state.

$L(A) = \{x_{1..T} \mid q \in S_A,\ \langle q, \epsilon \rangle \in V_A(x_{1..T})\}$

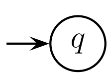
where $V_A(x_{t..T})$ returns a set of state,store pairs from which observations $x_{1..T}$ are acceptable:

$$\begin{aligned} V_A(x_{t..T}) = & \{\langle q, \epsilon \rangle \mid q \in F_A,\ x_{t..T} = \epsilon\} \\ & \cup \{\langle q, q''\alpha \rangle \mid \langle q, q'', x_t, q', q'' \rangle \in M_A,\ \langle q', q''\alpha \rangle \in V_A(x_{t+1..T})\} \\ & \cup \{\langle q, \alpha \rangle \mid \langle q, \epsilon, \epsilon, q', q \rangle \in M_A,\ \langle q', q\alpha \rangle \in V_A(x_{t..T})\} \\ & \cup \{\langle q, q''\alpha \rangle \mid \langle q, q'', \epsilon, q', \epsilon \rangle \in M_A,\ \langle q', \alpha \rangle \in V_A(x_{t..T})\} \end{aligned}$$

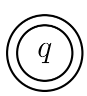
8.3 Graphical representation of PDAs

PDAs can be represented graphically:

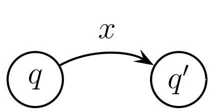
- start states: $q \in S$



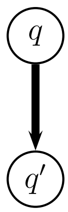
- final states: $q \in F$



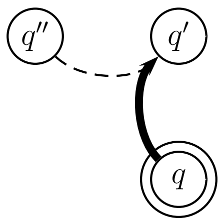
- state transitions: $\langle q, \epsilon, x, q', \epsilon \rangle \in M$



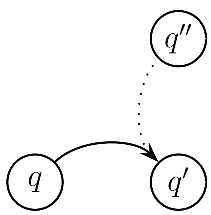
- expansions: $\langle q, \epsilon, \epsilon, q', q \rangle \in M$ (like transitions, but push previous state onto store)



- reductions: $\langle q, q'', \epsilon, q', \epsilon \rangle \in M$, $\;q_2 \in F$ (like trans, but remove last state from store)



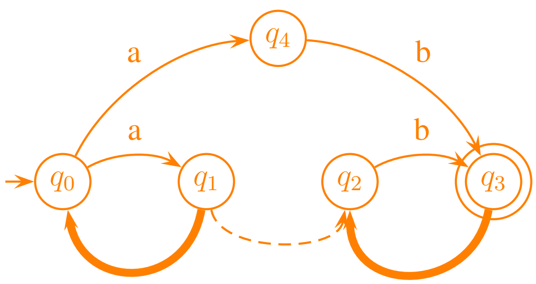
- conditional transitions: $\langle q, q'', \epsilon, q', q'' \rangle \in M$



8.4 Example PDA for: $a^n b^n; n > 0$ (S $\rightarrow$ a S b, S $\rightarrow$ a b)

Q:$\{q_0, q_1, q_2, q_3, q_4\}$, X:{a,b}, S:$\{q_0\}$, F:$\{q_3\}$,

M:$\{\langle q_0, \epsilon, \text{a}, q_1, \epsilon\rangle, \langle q_2, \epsilon, \text{b}, q_3, \epsilon\rangle, \langle q_0, \epsilon, \text{a}, q_4, \epsilon\rangle, \langle q_4, \epsilon, \text{b}, q_3, \epsilon\rangle, \langle q_1, \epsilon, \epsilon, q_0, q_1\rangle, \langle q_3, q_1, \epsilon, q_2, \epsilon\rangle\}$



8.5 PDAs can recognize CFGs; $\mathcal{L}(\textbf{CFG}) \subseteq \mathcal{L}(\textbf{PDA})$

Given any CFG $\langle C_G, X_G, S_G, R_G \rangle$, we can define a PDA $\langle Q, X, S, F, M \rangle$

(assume binary-branching CFG, since all CFGs can be translated into one):

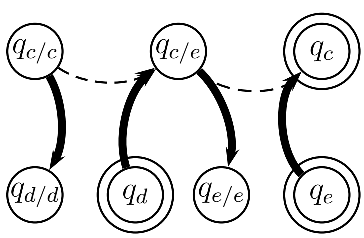
$Q = \{q_{c/c}, q_{c/e}, q_c \mid c \rightarrow d\ e \in R_G\} \cup \{q_{x/x}, q_x \mid x \in X_G\}$

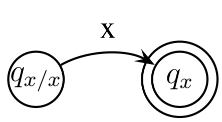
$X = X_G$

$S = \{q_{c/c} \mid c \in S_G\}$

$F = \{q_c \mid c \rightarrow d\ e \in R_G \text{ or } c \in X_G\}$

$$\begin{aligned} M = \{ & \langle q_{c/c}, \epsilon, \epsilon, q_{d/d}, q_{c/c} \rangle, \\ & \langle q_d, q_{c/c}, \epsilon, q_{c/e}, \epsilon \rangle, \\ & \langle q_{c/e}, \epsilon, \epsilon, q_{e/e}, q_{c/e} \rangle, \\ & \langle q_e, q_{c/e}, \epsilon, q_c, \epsilon \rangle \mid c \rightarrow d\ e \in R_G\} \cup \{\langle q_{x/x}, \epsilon, x, q_x, \epsilon \rangle \mid x \in X_G\} \end{aligned}$$

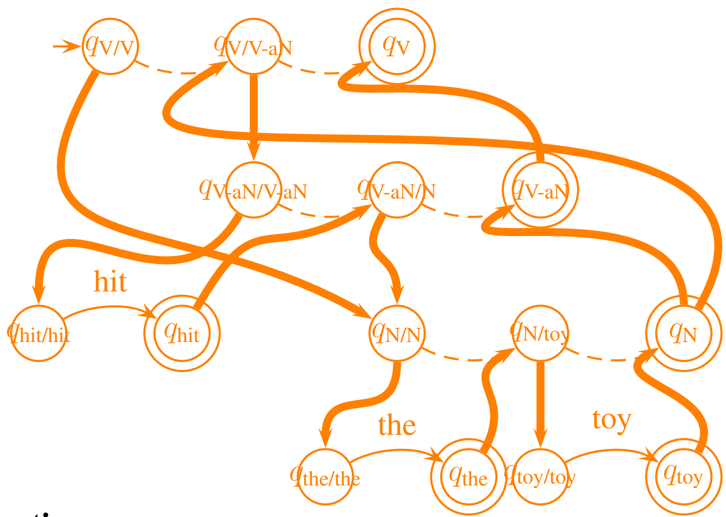




for example, a grammar with the following rules:

V $\rightarrow$ N V-aN
V-aN $\rightarrow$ hit N
N $\rightarrow$ the toy

would yield the following PDA:



Practice

How would the store look after each expansion, reduction, or transition in the above PDA?

$q_{\text{V/V}}$	(start state)
$q_{\text{N/N}}\ q_{\text{V/V}}$	(expand, pushing $q_{\text{V/V}}$ onto store)
$\ldots$?	

8.6 Right-expansion elimination and left-expansion elimination

Easy to implement recognizer if only one expansion, reduction per transition

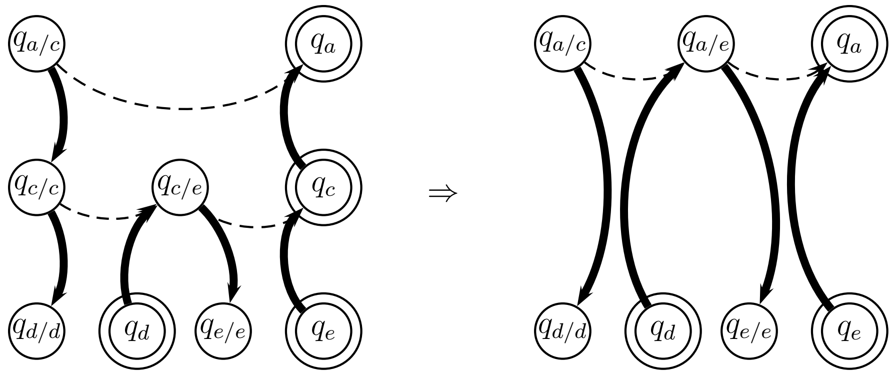
1. Right-expansion elimination ('awaited transition') – right child 'c' disappears:

$$M^{(0)} = M$$

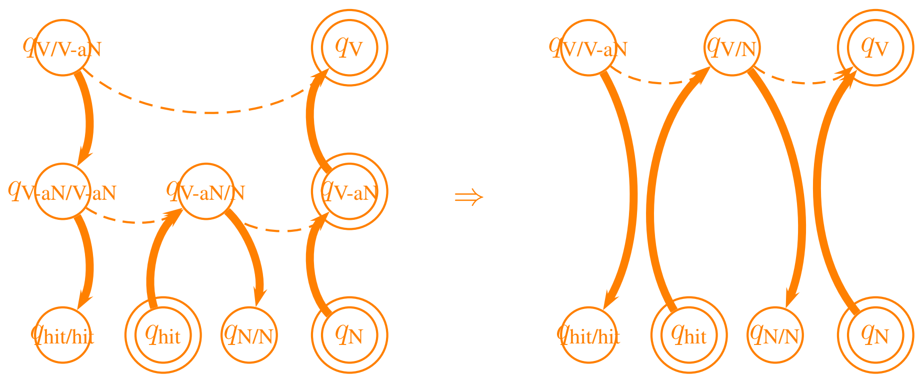
$$\begin{aligned} M^{(k)} = M^{(k-1)} \cup \{\, & \langle q_{a/c}, \epsilon, \epsilon, q_{d/d}, q_{a/c} \rangle, \langle q_d, q_{a/c}, \epsilon, q_{a/e}, \epsilon \rangle, \\ & \langle q_{a/e}, \epsilon, \epsilon, q_{e/e}, q_{a/e} \rangle, \langle q_e, q_{a/e}, \epsilon, q_a, \epsilon \rangle \\ & \mid \langle q_{a/c}, \epsilon, \epsilon, q_{c/c}, q_{a/c} \rangle \in M^{(k-1)}, \langle q_{c/c}, \epsilon, \epsilon, q_{d/d}, q_{c/c} \rangle \in M^{(k-1)}, \\ & \quad \langle q_d, q_{c/c}, \epsilon, q_{a/e}, \epsilon \rangle \in M^{(k-1)}, \langle q_{c/e}, \epsilon, \epsilon, q_{e/e}, q_{c/e} \rangle \in M^{(k-1)}, \\ & \quad \langle q_e, q_{c/e}, \epsilon, q_c, \epsilon \rangle \in M^{(k-1)}, \langle q_c, q_{a/c}, \epsilon, q_a, \epsilon \rangle \in M^{(k-1)} \} \end{aligned}$$

$$M' = M^{|R|}$$

Recognition is equivalent because child sub-models are preserved in order:



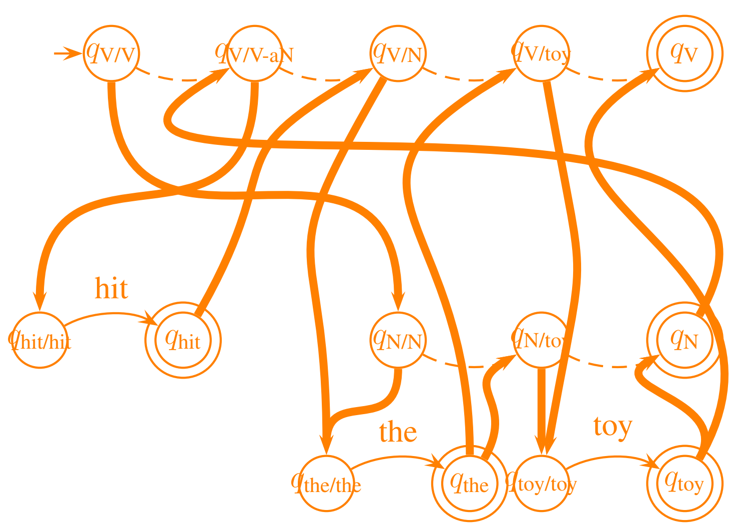
for example:



Repeat by mapping $q_{a/e}$, q_a in result to $q_{a/c}$, q_a in subsequent iteration.

Now all sequences of reductions are replaced with a single reduction!

Result (hiding unnecessary structure):



2. Left-expansion elimination ('active transition') – left child 'c' disappears:

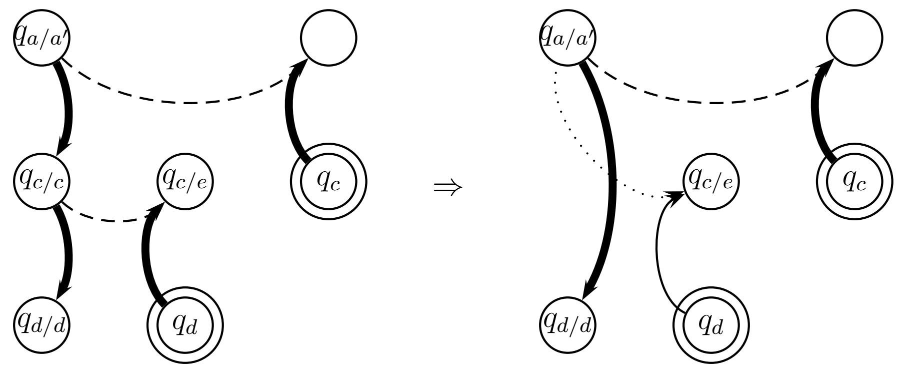
$$M^{(0)} = M$$

$$\begin{aligned} M^{(k)} = M^{(k-1)} \cup \{ & \langle q_{a/a'}, \epsilon, \epsilon, q_{d/d}, q_{a/a'} \rangle, \langle q_d, q_{a/a'}, \epsilon, q_{c/e}, q_{a/a'} \rangle \\ & \mid \langle q_{a/a'}, \epsilon, \epsilon, q_{c/c}, q_{a/a'} \rangle \in M^{(k-1)}, \langle q_{c/c}, \epsilon, \epsilon, q_{d/d}, q_{c/c} \rangle \in M^{(k-1)}, \\ & \langle q_d, q_{c/c}, \epsilon, q_{c/e}, \epsilon \rangle \in M^{(k-1)} \} \end{aligned}$$

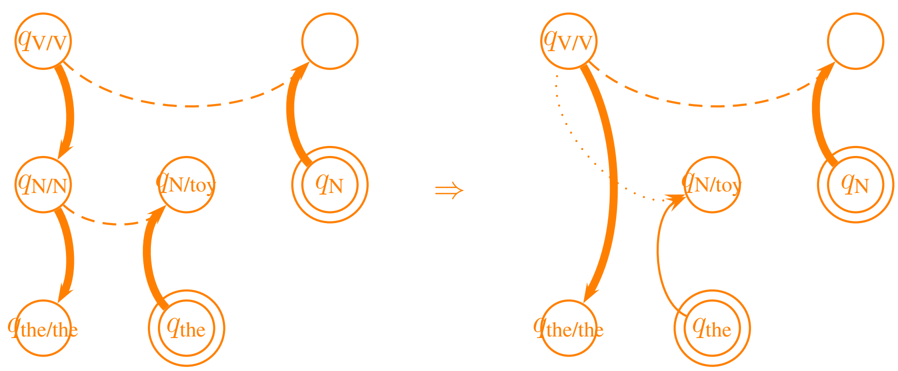
$$M' = M^{|R|}$$

Add 'conditional' ϵ-transition: $\langle q_d, q_{a/a'}, \epsilon, q_{c/e}, q_{a/a'} \rangle$ (drawn with dotted line)

Recognition is equivalent because child sub-models preserved in order:



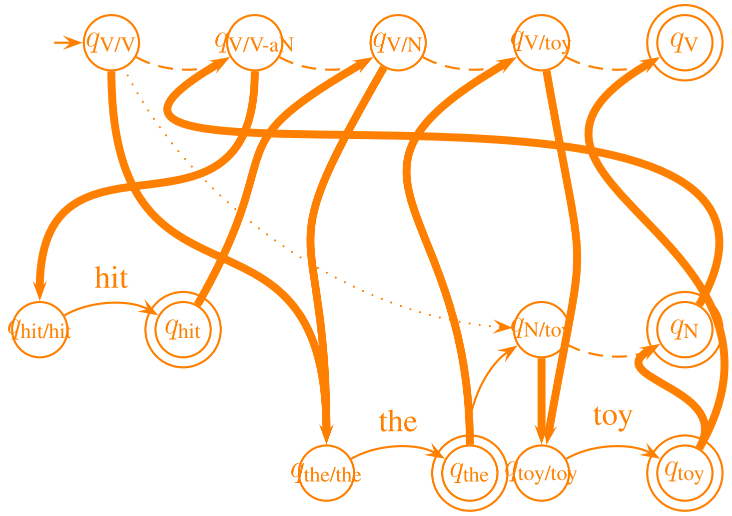
for example:



Repeat by mapping q_d, $q_{a/a'}$ in result to q_c, $q_{a/a'}$ in subsequent iteration.

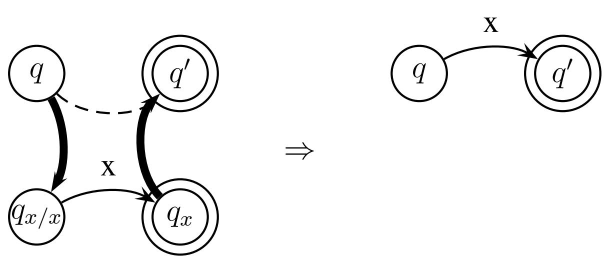
Now all sequences of expansions are replaced with a single expansion!

Result (hiding unnecessary structure):

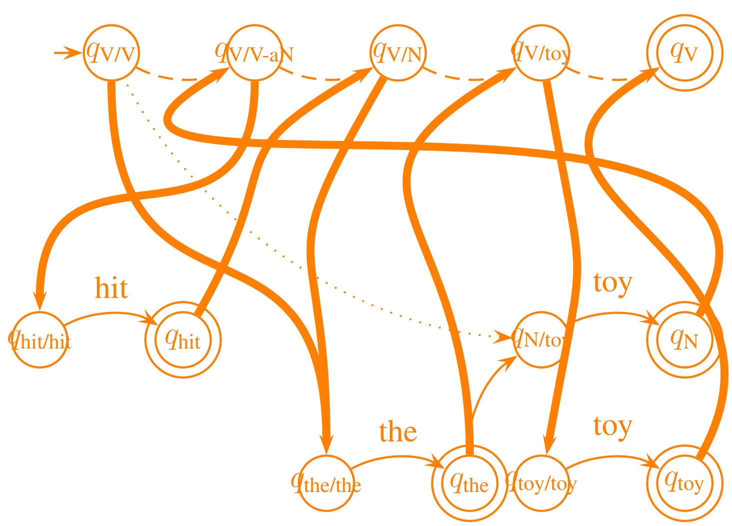


3. Transition flattening:

$$M' = M \cup \{\langle q, \epsilon, x, q', \epsilon\rangle \mid \langle q, \epsilon, \epsilon, q_{x/x}, q\rangle, \langle q_{x/x}, \epsilon, x, q_x, \epsilon\rangle, \langle q_x, q, \epsilon, q', \epsilon\rangle \in M, q' \in F\}$$

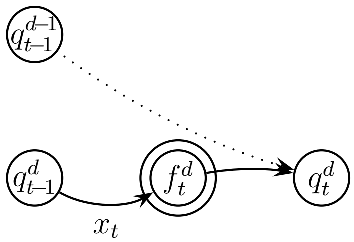


Result (hiding unnecessary structure):

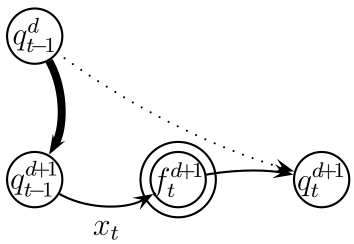


8.7 Implementation of PDA

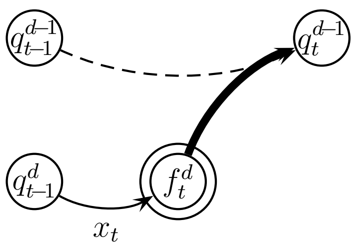
Since only one expansion/reduction between transitions, we only need four combinations:

a) 

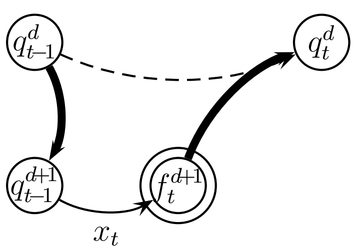
(no expand + trans + no reduce)

b) 

(expand + trans + no reduce)

c) 

(no expand + trans + reduce)

d) 

(expand + trans + reduce)

Recognition algorithm:

for each time step t:
  for each previous state q^d_{t-1} and store σ (where $d-1$ is the depth of the store):
    a) for each final state f^d_t and current state q^d_t: (expand +) trans
      $V[t, q^d_t\sigma] = V[t, q^d_t\sigma]$ or $(V[t-1, q^d_{t-1}\sigma]$ and $M[q^d_{t-1}, \epsilon, x_t, f^d_t, \epsilon]$
      and $M[f^d_t, q^{d-1}_{t-1}, \epsilon, q^d_t, q^{d-1}_{t-1}])$
    b) for each previous state at deeper level q^{d+1}_{t-1}, final state f^{d+1}_t and current state q^{d+1}_t:

$V[t, q_t^{d+1} q_t^d \sigma] = V[t, q_t^{d+1} q_t^d \sigma]$ or $(V[t-1, q_{t-1}^d \sigma]$ and $M[q_{t-1}^d, \epsilon, \epsilon, q_{t-1}^{d+1}, q_{t-1}^d]$
and $M[q_{t-1}^{d+1}, \epsilon, x_t, f_t^{d+1}, \epsilon]$
and $M[f_t^{d+1}, q_{t-1}^d, \epsilon, q_t^{d+1}, q_{t-1}^d])$

c) for each final state f_t^d and current state q_t^{d-1}: (expand +) trans + reduce
$V[t, q_t^{d-1} \sigma] = V[t, q_t^{d-1} \sigma]$ or $(V[t-1, q_{t-1}^d q_{t-1}^{d-1} \sigma]$ and $M[q_{t-1}^d, \epsilon, x_t, f_t^d, \epsilon]$
and $M[f_t^d, q_{t-1}^{d-1}, \epsilon, q_t^{d-1}, \epsilon])$

d) for each previous state at deeper level q_{t-1}^{d+1} and final state f_t^{d+1} and current state q_t^d:
$V[t, q_t^d q_t^{d-1} \sigma] = V[t, q_t^d q_t^{d-1} \sigma]$ or $(V[t-1, q_{t-1}^d q_{t-1}^{d-1} \sigma]$ and $M[q_{t-1}^d, \epsilon, \epsilon, q_{t-1}^{d+1}, q_{t-1}^d]$
and $M[q_{t-1}^{d+1}, \epsilon, x_t, f_t^{d+1}, \epsilon]$
and $M[f_t^{d+1}, q_{t-1}^d, \epsilon, q_t^d, \epsilon])$

Correctness can be shown from definition of accepted languages.

But, iterating over all possible infinite stores σ results in exponential complexity.

8.8 Example

Transforming the following CFG:

T → V T (a top-level discourse type)
V → N V-aN
V-aN → V-aN R-aN
V-aN → hit N
R-aN → off N
A-aN → off N
N → N A-aN
N → the cat
N → the toy
N → the mat

then recognizing 'the cat hit the toy off the mat' results in the following sequence: