Parsing Beyond Context-Free Grammars: Embedded Push-Down Automata

Laura Kallmeyer & Tatiana Bladier
Heinrich-Heine-Universität Düsseldorf

Sommersemester 2018

Overview

[Kal10]

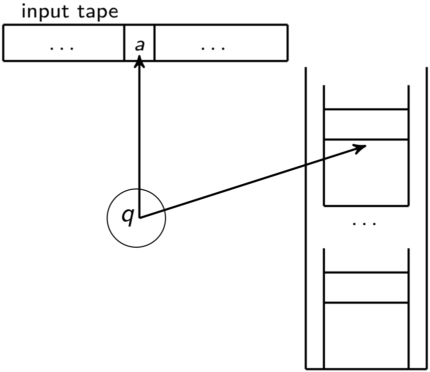
Intuition (1)

For a language L, there is a TAG G with $L = L(G)$ iff there is an embedded PDA (EPDA) M with $L(G) = L(M)$.

An EPDA is an extension of PDA:

- An EPDA uses a stack of non-empty push-down stores (nested stack)
- Each push-down store contains stack symbols
- An EPDA is a "second-order" push-down automaton

Intuition (2)



Intuition (3)

- An EPDA uses a stack of non-empty push-down stores
- Stack pointer always points to top symbol of top stack
- The two stages of a move:
  - The top-most push-down store Υ is treated as in the PDA case (replace top-most stack symbol by new sequence of stack symbols)
  - The resulting new push-down store Υ' is replaced by a sequence of k push-down stores, including Υ' ($k \geq 0$).
- Input accepted if stack empty or automaton in a special final state (equivalent as for PDA)

Intuition (4)

Use an EPDA to recognize $L_4 = \{a^n b^n c^n d^n \mid n > 0\}$. How?

- Each input symbol corresponds to a different state
- For each *a* encountered in the input,
  - B is pushed on the top-most stack (to ensure that number of *a*s equal to number of *b*s and *c*s)
  - Below the top-most stack, an extra stack with a single D is introduced (ensures that $\#_a = \#_d$)
- For each *b* encountered in the input,
  - If the top-most symbol of the top-most stack is B,
  - below the top-most stack, an extra stack with a single C is introduced (ensures that $\#_b = \#_c$)

Intuition (5)

- After reading all *a*s and *b*s, we now should have a sequence of stacks, each one with a single symbol $x \in \{C, D\}$, $\#_C = \#_D$, with all C-stacks preceding all D-stacks. Now we delete the stacks:
- For each c encountered in the input,
  - If the top-most symbol of the top-most stack is C,
  - delete stack and proceed
- For each d encountered in the input,
  - If the top-most symbol of the top-most stack is D,
  - delete stack and proceed
- Accept if no input symbols left and stack empty.

Definition of an EPDA (1)

Definition 1 (Embedded Push-Down Automaton)

An *Embedded Push-Down Automaton (EPDA)* M is a 7-tuple $\langle Q, \Sigma, \Gamma, \delta, q_0, Q_F, Z_0 \rangle$, where

- Q is a finite set of states, $q_0 \in Q$ is the start state and $Q_F \subseteq Q$ is the set of final states.
- Γ is the finite set of stack symbols and $Z_0 \in \Gamma$ is the initial stack symbol.
- Σ is the finite set of input symbols.
- δ is the transition function $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to P_{fin}(Q \times \Upsilon^* \times \Gamma^* \times \Upsilon^*)$, where $\Upsilon = \Gamma^*$ correspond to push-downs of stack symbols.

Definition of an EPDA (2)

We can give an instantaneous description of an EPDA by a *configuration*. A configuration is of type $Q \times \Upsilon^* \times \Sigma^* \times \Sigma^*$, i.e., it consists of

- the current state $q \in Q$,
- the stack of stacks $s \in \Upsilon^*$,
- the already recognized part of the input $w_1 \in \Sigma^*$ and
- the part $w_2 \in \Sigma^*$ which is yet to be recognized.

Within Υ^*, we mark each start (bottom) of a stack with the symbol $\ddagger$ (assuming without loss of generality that $\ddagger \notin \Gamma$) and, as a convention, the top is the rightmost element.

The initial configuration of an EPDA is $\langle q_0, \ddagger Z_0, \varepsilon, w \rangle$, where the automaton is in the start state q_0, there is only one stack on the stack, this single stack contains only the initial stack symbol Z_0 and the entire input is still to be recognized.

Definition of an EPDA (3)

Definition 2 (EPDA transition)

Let $\langle Q, \Sigma, \Gamma, \delta, q_0, Q_F, Z_0 \rangle$ be an EPDA, $\Upsilon = \{\ddagger\gamma \,|\, \gamma \in \Gamma^*\}$.

- For all $q_1, q_2 \in Q, a \in (\Sigma \cup \{\varepsilon\}), w_1, w_2 \in \Sigma^*, \alpha, \alpha_1, \alpha_2 \in \Upsilon^*, Z \in \Gamma, \beta, \gamma \in \Gamma^*$,
  - **a)** $\langle q_1, \alpha\ddagger\beta Z, w_1, aw_2 \rangle \vdash \langle q_2, \alpha\alpha_1\ddagger\beta\gamma\alpha_2, w_1 a, w_2 \rangle$

    if $\langle q_2, \alpha_1, \gamma, \alpha_2 \rangle \in \delta(q_1, a, Z)$ and $\beta\gamma \neq \varepsilon$.
  - **b)** $\langle q_1, \alpha\ddagger Z, w_1, aw_2 \rangle \vdash \langle q_2, \alpha\alpha_1\alpha_2, w_1 a, w_2 \rangle$

    if $\langle q_2, \alpha_1, \varepsilon, \alpha_2 \rangle \in \delta(q_1, a, Z)$.
- $\overset{*}{\vdash}$ is the reflexive transitive closure of $\vdash$.

Definition of an EPDA (4)

Note that empty transitions are allowed ($a \in (\Sigma \cup \{\varepsilon\})$), i.e., transitions that do not read an input symbol.

The case b) covers the special case where the top-most stack is emptied. We then assume that this stack gets deleted and therefore even its bottom-stack symbol $\ddagger$ disappears.

Definition of an EPDA (5)

We now define the two modes of acceptance for EPDA:

Definition 3 (Language of an EPDA)

Let $M = \langle Q, \Sigma, \Gamma, \delta, q_0, Q_F, Z_0 \rangle$ be an EPDA.

1. M accepts the languages $L(M)$ in its final states:

$$L(M) = \{w \mid \langle q_0, \ddagger Z_0, \varepsilon, w \rangle \overset{*}{\vdash} \langle q_f, \alpha, w, \varepsilon \rangle \text{ for some } q_f \in Q_F, \alpha \in \Upsilon^* \}.$$

2. M accepts the languages $N(M)$ by empty stack:

$$N(M) = \{w \mid \langle q_0, \ddagger Z_0, \varepsilon, w \rangle \overset{*}{\vdash} \langle q, \varepsilon, w, \varepsilon \rangle \text{ for some } q \in Q \}.$$

Definition of an EPDA (6)

The two modes of acceptance yield the same sets of languages [VS87]:

Lemma 4

1. *For every EPDA M, there is an EPDA M' such that $L(M) = N(M')$.*
2. *For every EPDA M, there is an EPDA M' such that $N(M) = L(M')$.*

Definition of an EPDA (7)

- To show the first part, for a given M, we have to add transitions that move into a new "stack-emptying" state q' once we have reached a final state and that then empty the stack.
- For the second part, we add to M a new initial state and a new initial stack symbol. From these we move to the original initial symbols, perform the run of the automaton M and, once we reach a configuration where only our new stack symbol remains on the stack, move into a new final state.

Sample EPDAs (1)

EPDA for L_4: $M = \langle Q, \Sigma, \Gamma, \delta, q_0, Q_F, Z_0 \rangle$ with $N(M) = L_4$

$Q = \{q_0, q_1, q_2, q_3\}$, $Q_F = \emptyset$, $Z_0 = \#$, $\Sigma = \{a, b, c, d\}$,
$\Gamma = \{\#, B, C, D\}$

Transition function δ:

$$
\begin{array}{ll}
\delta(q_0, a, \#) = \{(q_0, \ddagger D, B, \varepsilon)\} & \delta(q_0, a, B) = \{(q_0, \ddagger D, BB, \varepsilon)\} \\
\delta(q_0, b, B) = \{(q_1, \ddagger C, \varepsilon, \varepsilon)\} & \delta(q_1, b, B) = \{(q_1, \ddagger C, \varepsilon, \varepsilon)\} \\
\delta(q_1, c, C) = \{(q_2, \varepsilon, \varepsilon, \varepsilon)\} & \delta(q_2, c, C) = \{(q_2, \varepsilon, \varepsilon, \varepsilon)\} \\
\delta(q_2, d, D) = \{(q_3, \varepsilon, \varepsilon, \varepsilon)\} & \delta(q_3, d, D) = \{(q_3, \varepsilon, \varepsilon, \varepsilon)\}
\end{array}
$$

Sample EPDAs (2)

Recognition of *aabbccdd* with M:

$$\begin{aligned}
& (q_0, \ddagger\#, \varepsilon, aabbccdd) \\
\vdash\; & (q_0, \ddagger D \ddagger B, a, abbccdd) \\
\vdash\; & (q_0, \ddagger D \ddagger D \ddagger BB, aa, bbccdd) \\
\vdash\; & (q_1, \ddagger D \ddagger D \ddagger C \ddagger B, aab, bccdd) \\
\vdash\; & (q_1, \ddagger D \ddagger D \ddagger C \ddagger C, aabb, ccdd) \\
\vdash\; & (q_2, \ddagger D \ddagger D \ddagger C, aabbc, cdd) \\
\vdash\; & (q_2, \ddagger D \ddagger D, aabbcc, dd) \\
\vdash\; & (q_3, \ddagger D, aabbccd, d) \\
\vdash\; & (q_3, \varepsilon, aabbccdd, \varepsilon)
\end{aligned}$$

Sample EPDAs (3)

EPDA $M = \langle Q, \Sigma, \Gamma, \delta, q_0, Q_F, \# \rangle$ with $L(M) = L_4$.

$Q = \{q_0, q_1, q_2, q_3, q_4\}$, $Q_F = \{q_4\}$, $\Sigma = \{a, b, c, d\}$,
$\Gamma = \{B, C, D, \#\}$.

Transition function δ:

$$
\begin{array}{ll}
\delta(q_0, a, \#) = \{(q_0, \ddagger\# \ddagger D, B, \varepsilon)\} & \delta(q_0, a, B) = \{(q_0, \ddagger D, BB, \varepsilon)\} \\
\delta(q_0, b, B) = \{(q_1, \ddagger C, \varepsilon, \varepsilon)\} & \delta(q_1, b, B) = \{(q_1, \ddagger C, \varepsilon, \varepsilon)\} \\
\delta(q_1, c, C) = \{(q_2, \varepsilon, \varepsilon, \varepsilon)\} & \delta(q_2, c, C) = \{(q_2, \varepsilon, \varepsilon, \varepsilon)\} \\
\delta(q_2, d, D) = \{(q_3, \varepsilon, \varepsilon, \varepsilon)\} & \delta(q_3, d, D) = \{(q_3, \varepsilon, \varepsilon, \varepsilon)\} \\
\delta(q_3, \varepsilon, \#) = \{(q_4, \varepsilon, \varepsilon, \varepsilon)\} &
\end{array}
$$

Sample EPDAs (4)

Recognize *aabbccdd* with M:

$$
\begin{array}{l}
\quad (q_0, \ddagger\#, \epsilon, aabbccdd) \\
\vdash (q_0, \ddagger\# \ddagger D \ddagger B, a, abbccdd) \\
\vdash (q_0, \ddagger\# \ddagger D \ddagger D \ddagger BB, aa, bbccdd) \\
\vdash (q_1, \ddagger\# \ddagger D \ddagger D \ddagger C \ddagger B, aab, bccdd) \\
\vdash (q_1, \ddagger\# \ddagger D \ddagger D \ddagger C \ddagger C, aabb, ccdd) \\
\vdash (q_2, \ddagger\# \ddagger D \ddagger D \ddagger C, aabbc, cdd) \\
\vdash (q_2, \ddagger\# \ddagger D \ddagger D, aabbcc, dd) \\
\vdash (q_3, \ddagger\# \ddagger D, aabbccd, d) \\
\vdash (q_3, \ddagger\#, aabbccdd, \epsilon) \\
\vdash (q_4, \epsilon, aabbccdd, \epsilon)
\end{array}
$$

TAG and EPDA (1)

Proposition 1

For every TAG G there is an EPDA M and vice versa such that $L(G) = L(M)$ [VS87].

Vijay-Shanker's proof shows how to construct an equivalent Modified Head Grammar (MHG) for a given EPDA and vice versa. Since the equivalence between MHG and TAG has been established earlier, this proves the equivalence between TAG and EPDA.

TAG and EPDA (2)

Construction of an equivalent EPDA for a given TAG:

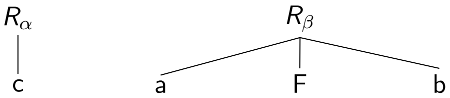
- We assume one stack symbol for each node. The EPDA simulates a top-down traversal of the derived tree.
- The symbol corresponding to the next node to be expanded is the top-most stack symbol of the automaton.
- When we adjoin to a node, we add the root node symbol of the new auxiliary tree to the current top stack.
- When moving down in a tree along the spine of an auxiliary tree, we place new stacks above and below the current one. These encode the parts to the left and the right of the spine of the adjoined auxiliary tree.
- When moving down without being on the spine of some auxiliary tree, we simply replace the mother node symbol by the daughters (in reverse order, i.e., the leftmost daughter on top).

TAG and EPDA (3)

- In order to separate adjunction from moving to the daughters, we distinguish top and bottom ($\top$ and $\perp$) node names on the stack.
- For a node N, the symbol $N^\top$ is replaced with $N^\perp$ if no adjunction is predicted and with the symbols $N^\perp R_\beta$ if adjunction of β is predicted and R_β is the root node of β.

TAG and EPDA (4)

Sample TAG:



(R_α and R_β allow for adjunction of β.)

TAG and EPDA (5)

Equivalent EPDA: $M = \langle Q, \Sigma, \Gamma, \delta, q_0, Q_F, Z_0 \rangle$ with

$Q = \{q_0, q_1, q_2, q_3\}$, $Q_F = \emptyset$, $Z_0 = \#$, $\Sigma = \{a, b, c\}$,
$\Gamma = \{\#, R_\alpha, R_\beta, F, A, B, C\}$

Transition function δ:

$\langle q, \varepsilon, R_\alpha^\top, \varepsilon \rangle \in \delta(q, \varepsilon, \#)$	start initial tree
$\langle q, \varepsilon, R_\alpha^\perp, \varepsilon \rangle \in \delta(q, \varepsilon, R_\alpha^\top)$	no adjunction at R_α
$\langle q, \varepsilon, C, \varepsilon \rangle \in \delta(q, \varepsilon, R_\alpha^\perp)$	move down
$\langle q, \varepsilon, R_\alpha^\perp R_\beta^\top, \varepsilon \rangle \in \delta(q, \varepsilon, R_\alpha^\top)$	adjunction of β
$\langle q, \varepsilon, R_\beta^\perp R_\beta^\top, \varepsilon \rangle \in \delta(q, \varepsilon, R_\beta^\top)$	adjunction of β
$\langle q, \varepsilon, R_\beta^\perp, \varepsilon \rangle \in \delta(q, \varepsilon, R_\beta^\top)$	no adjunction at R_β

TAG and EPDA (6)

$\langle q, \ddagger B, F, \ddagger A \rangle \in \delta(q, \varepsilon, R_\beta^\perp)$	move down
$\langle q, \varepsilon, \varepsilon, \varepsilon \rangle \in \delta(q, \varepsilon, F)$	no adjunction at F, move back
$\langle q, \varepsilon, \varepsilon, \varepsilon \rangle \in \delta(q, a, A)$	match a with input
$\langle q, \varepsilon, \varepsilon, \varepsilon \rangle \in \delta(q, b, B)$	match b with input
$\langle q, \varepsilon, \varepsilon, \varepsilon \rangle \in \delta(q, c, C)$	match c with input

Acceptance with the empty stack.

TAG and EPDA (7)

Sample run for the input *aacbb*:

Stacks	remaining input	
$\ddagger\#$	aacbb	
$\ddagger R_\alpha^\top$	aacbb	start traversal of α
$\ddagger R_\alpha^\perp R_\beta^\top$	aacbb	predict adjunction of β
$\ddagger R_\alpha^\perp R_\beta^\perp R_\beta^\top$	aacbb	predict adjunction of β
$\ddagger R_\alpha^\perp R_\beta^\perp R_\beta^\perp$	aacbb	predict no adjunction
$\ddagger B \ddagger R_\alpha^\perp R_\beta^\perp F \ddagger A$	aacbb	move down in β
$\ddagger B \ddagger R_\alpha^\perp R_\beta^\perp F$	acbb	scan a
$\ddagger B \ddagger R_\alpha^\perp R_\beta^\perp$	acbb	leave β
$\ddagger B \ddagger B \ddagger R_\alpha^\perp F \ddagger A$	acbb	move down in β

TAG and EPDA (8)

$\ddagger B \ddagger B \ddagger R_\alpha^\perp F$	cbb	scan a
$\ddagger B \ddagger B \ddagger R_\alpha^\perp$	cbb	leave β
$\ddagger B \ddagger B \ddagger C$	cbb	move down in α
$\ddagger B \ddagger B$	bb	scan c
$\ddagger B$	b	scan b
ε	ε	scan b

References I

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

[VS87] K. Vijay-Shanker.
A Study of Tree Adjoining Grammars.
PhD thesis, University of Pennsylvania, 1987.