DEEP PUSHDOWN TRANSDUCERS AND STATE TRANSLATION SCHEMES

Peter Solár
Doctoral Degree Programme (5), FIT BUT
E-mail: xsolar05@stud.fit.vutbr.cz

Supervised by: Alexander Meduna
E-mail: meduna@fit.vutbr.cz

Abstract: This paper presents deep pushdown transducers and state translation schemes as two models which can be used in syntax-directed translation. *Deep pushdown transducers* are based on *deep pushdown automata*. These transducers can expand non-input pushdown symbols deeper in a pushdown. *State translation schemes* fundamentally work like state grammars but differ in possibility to produce two different output strings in one derivation.

Keywords: parsing, pushdown automata, deep pushdown automata, state grammars, pushdown transducers, deep pushdown transducers, syntax-directed translation scheme, state translation scheme

1 INTRODUCTION

If we talk about formal language theory, we usually focus on two core models - grammar and automaton. In a grammar we aim to generate a string belonging to the language using production rules. On the other hand, an automaton is supposed to run on a given input sequence and tries to decide if this input sequence is a string belonging to the language recognized by the automaton. However there exists other formal models. In this paper, we focus on transducers and translation schemes. Transducers, simply said, are automata which are enhanced by the possibility to produce some output string while trying to accept an input string. Likewise, translation schemes are extended versions of grammars.

This paper presents deep pushdown transducers and state translation schemes as two models which can be used in syntax-directed translation. *Deep pushdown transducers* are based on deep pushdown automata presented by Meduna in 2006 ([7]). *State translation schemes* are based on state grammars introduced by Kasai in 1970 ([5]). These two models are equivalent and describe the same infinite hierarchy of languages lying between context-free and context-sensitive languages (see [7]). Deep pushdown automaton has the possibility to expand a non-input pushdown symbol deeper in its pushdown, not only on the pushdown top. State grammar is a context-free grammar with additional mechanism which consists in adding states (similarly to finite automaton). At each derivation step state grammar is in some state which influences the choice of the next applied production rule. Next state is included in the production rule.

2 PRELIMINARIES

This paper assumes that the reader is familiar with the theory of automata and formal languages (see [1], [2], [3] and [6]).

$\mathbb{N}^+$ denotes the set of all positive integers. For an alphabet, Σ, Σ^* represents the free monoid generated by Σ under the operation of concatenation. The identity of Σ^* is denoted by ε. A homomorphism f over V^* is a coding if $f(A) \in \{A, \varepsilon\}$ for every $A \in V$.

A function $f : A \to B$ is called injective (one-to-one) if each element of B appears at most once as the image of an element of A. A function $f : A \to B$ is called surjective (onto) if $f(A) = B$. That is, if each element of B is the image of at least one element of A. A function that is both injective and surjective is called bijective. A permutation of a set A is a bijective function $\phi(A) : A \to A$.

Set $\Sigma^+ = \Sigma^* - \{\varepsilon\}$. Algebraically, Σ^+ is thus the free semigroup generated by Σ under the operation of concatenation. For a string $w \in \Sigma^*$, $|w|$ denotes the length of string w. For $W \subseteq \Sigma$, $occur(w, W)$ denotes the number of occurrences of symbols from W in w. $alph(w)$ denotes the set of symbols occurring in string w.

A *pushdown transducer* (see [4]) is a 8-tuple $M^T = (Q, \Sigma_I, \Gamma, \Sigma_O, R, s, S, F)$, where Q is a finite set of the states, Σ_I is an input alphabet, Γ is a pushdown alphabet, Σ_O is an output alphabet, $\Sigma_I \subseteq \Gamma$, $s \in Q$ is the start state, $S \in \Gamma$ is the start pushdown symbol, $F \subseteq Q$ is a set of finite states. Γ and Σ_O are pairwise disjoint. $R \subseteq Q \times (\Sigma_I \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \times Q \times \Gamma^* \times (\Sigma_O \cup \{\varepsilon\})$. Instead of $(p, a, A, q, w, a') \in R$, where $p, q \in Q$, $a \in \Sigma_I$, $A \in \Gamma - \Sigma_I$, $w \in \Gamma^*$, $a' \in \Sigma_O$, we usually write $r = paA \to qwa'$ $(r \in R)$ and call r a *rule*.

A *state grammar* (see [5]) is a 5-tuple $G = (V, W, T, P, S)$, where V is a total alphabet, W is a finite set of states, $T \subseteq V$ is an alphabet of terminals, $S \in (V - T)$ is the start symbol and $P \in (W \times (V - T)) \times (W \times V+)$ is a finite relation. Instead of $(q, A, p, v) \in P$, we write $(q, A) \to (p, v) \in P$ throughout.

For every $z \in V^*$, set ${}_G states(z) = \{q \mid (q, B) \to (p, v) \in P$, where $B \in (V - T) \cap alph(z)$, $v \in V^+$, $q, p \in W\}$. If $(q, A) \to (p, v) \in P$, $x, y \in V^*$, ${}_G states(x) \cap \{q\} = \varnothing$, then G makes a derivation step from (q, xAy) to (p, xvy), symbolically written as $(q, xAy) \Rightarrow (p, xvy)[(q, A) \to (p, v)]$ in G. In addition, if n is a positive integer satisfying $occur(xA, V - T) \leq n$, we say that $(q, xAy) \Rightarrow (p, xvy)$ $[(q, A) \to (p, v)]$ is n-limited, symbolically written as (q, xAy) ${}_n\Rightarrow$ (p, xvy) $[(q, A) \to (p, v)]$. Usually if there is no possibility of confusion we simplify $(q, xAy) \Rightarrow (p, xvy)$ $[(q, A) \to (p, v)]$ to $(q, xAy) \Rightarrow (p, xvy)$ and (q, xAy) ${}_n\Rightarrow$ (p, xvy) $[(q, A) \to (p, v)]$ to (q, xAy) ${}_n\Rightarrow$ (p, xvy). In the standard manner we extend $\Rightarrow$ to $\Rightarrow^m$, $m \geq 0$. Based on $\Rightarrow^m$ we can define $\Rightarrow^+$ and $\Rightarrow^*$. Let $n \in \mathbb{N}^+$ and $\alpha, \beta \in (W \times V)$. To express that every derivation step in $\alpha \Rightarrow^m \beta$, $\alpha \Rightarrow^+ \beta$ and $\alpha \Rightarrow^* \beta$ is n-limited, we write $\alpha_n\Rightarrow^m \beta$, $\alpha_n\Rightarrow^+ \beta$ and $\alpha_n\Rightarrow^* \beta$.

The language of G, $L(G)$, is defined as $L(G) = \{w \in T^* \mid (q, S) \Rightarrow^* (p, w), q, p \in W\}$. Moreover, we define for every $n \geq 1$, $L(G, n) = \{w \in T^* \mid (q, S)$ ${}_n\Rightarrow^*$ $(p, w), q, p \in W\}$. A derivation of the form (q, S) ${}_n\Rightarrow^*$ (p, w), where $q, p \in W$ and $w \in T^*$, represents a *successful n-limited generation* of w in G.

A *deep pushdown automaton* (see [7]) is a 7-tuple ${}_dM = (Q, \Sigma, \Gamma, R, s, S, F)$, where d is the maximum depth at which non-input symbol can be expanded, Q is a finite set of the states, Σ is an input alphabet, Γ is a pushdown alphabet, $\Sigma \subseteq \Gamma$, $\Gamma - \Sigma$ contains a special bottom symbol denoted by #, $s \in Q$ is the start state, $S \in \Gamma$ is the start pushdown state, $F \subseteq Q$ is a set of finite states. Sets $\mathbb{N}$, Q and Γ are pairwise disjoint. $R \subseteq (\mathbb{N}^+ \times Q \times (\Gamma - (\Sigma \cup \{\#\})) \times Q \times (\Gamma - \{\#\})^+) \bigcup (\mathbb{N}^+ \times Q \times \{\#\} \times Q \times (\Gamma - \{\#\})^* \{\#\})$. Instead of $(m, p, A, q, w) \in R$, where $m \leq d$, $p, q \in Q$, $A \in \Gamma - \Sigma$, $w \in \Gamma^+$, we usually write $r = mpA \to qw$ and call r a *rule*.

A *configuration* of the deep pushdown automaton ${}_dM$ is a triple $Q \times \Sigma^* \times (\Gamma - \{\#\})^* \{\#\}$. Let χ be a set of all configurations of automaton ${}_dM$ and let $x, y \in \chi$ be two configurations. $x \vdash y$ is a move between these two configuration. If $x = (p, au, az), y = (q, u, z)$, where $p, q \in Q, a \in \Sigma, u \in \Sigma^*, z \in \Gamma^*$, then ${}_dM$ *pops* its pushdown from x to y, x ${}_p\vdash$ y. ${}_dM$ *expands* its pushdown if $x = (p, au, wAz), y = (q, au, wvz), r = p(A) \vdash q(v) \in R$, accordingly to the rule r. Symbolically x ${}_e\vdash$ $y[p(A) \to q(v)]$ or x ${}_e\vdash$ y if there is only one usable rule. In the standard manner we can extend ${}_p\vdash$, ${}_e\vdash$ and $\vdash$ to ${}_p\vdash^m$, ${}_e\vdash^m$ and $\vdash^m$, respectively, for $m \geq 0$. Then based on ${}_p\vdash^m$, ${}_e\vdash^m$ and $\vdash^m$, define ${}_p\vdash^+$, ${}_p\vdash^*$, ${}_e\vdash^+$, ${}_e\vdash^*$, $\vdash^+$ and $\vdash^*$.

Let ${}_dM$ be of maximal depth $d \in \mathbb{N}$. We define a *language accepted by* ${}_dM$ as $L({}_dM) = \{w \in \Sigma^* : (s,w,S\#) \vdash^* (f,\varepsilon,\#) \in {}_dM \text{ with } f \in F\}$, *language accepted by* ${}_dM$ *by empty pushdown* as $L_{emptyPD}({}_dM) = \{w \in \Sigma^* : (s,w,S\#) \vdash^* (q,\varepsilon,\#) \in {}_dM \text{ with } q \in Q\}$ and *language accepted by* ${}_dM$ *by entering final state* as $L_{final}({}_dM) = \{w \in \Sigma^* : (s,w,S\#) \vdash^* (f,\varepsilon,v\#) \in {}_dM \text{ with } f \in F, v \in \Gamma^*\}$.

For every state grammar G and for every $n \geq 1$, there exists a deep pushdown automaton of depth n, ${}_nM$, such that $L(G,n) = L({}_nM)$.

3 DEFINITIONS

3.1 STATE TRANSLATION SCHEME

As an analogy to syntax-directed translation schemes used in translation of context-free languages we can define state translation schemes as an extension of state grammars that describes the process of translating input strings into output strings. The scheme associates one production rule of the target state language to each production rule of the source state language.

A *state translation scheme* is a 6-tuple $\tau = (V,W,T_I,T_O,P,S)$, where V is a total alphabet, W is a finite set of states, $T_I \subseteq V$ is an alphabet of input terminals, $T_O \subseteq V$ is an alphabet of output terminals, $N = V - (T_I \cup T_O)$, $S \in N$ is the start symbol, $\bar{N} \subseteq N$ and $P \in (W \times (N)) \times (W \times (\bar{N} \cup T_I)^* \times (\phi(\bar{N}) \cup T_O)^*)$ is a finite relation. Instead of $(p,A,q,x,y) \in P$, we write $(p,A) \to (q,x,y) \in P$ throughout. For every $z \in (N \cup T_I)^*$, set ${}_\tau states(z) = \{p \mid (p,B) \to (q,x,y) \in P$, where $B \in N \cap alph(z)$, $x \in (N \cup T_I)^+$, $y \in (N \cup T_O)^+$, $p,q \in W\}$.

A *translation form* is a pair (u,v) where u is a sentential form of the underlying grammar and v is the translation resulted by the derivation of u.

Let $\vartheta(\tau)$ be a *translation* ϑ defined by state translation scheme τ, $\vartheta(\tau) = \{(x,y) : (S,S) \Rightarrow^* (x,y)$, $x \in T_I$, $y \in T_O\}$.

3.2 DEEP PUSHDOWN TRANSDUCER

The deep pushdown transducer differs from the standard pushdown transducer by possibility to expand non-input pushdown symbols deeper in the pushdown.

A *deep pushdown transducer* is an 8-tuple ${}_dM^T = (Q, \Sigma_I, \Gamma, \Sigma_O, R, s, S, F)$, where d is the maximum depth at which non-input symbol can be expanded, Q is a finite set of the states, Σ_I is an input alphabet, Γ is an pushdown alphabet, $\Sigma_I \subseteq \Gamma$, Σ_O is an output alphabet, $\Sigma_O \not\subseteq \Gamma$, $\Gamma - \Sigma_I$ contains a special bottom symbol denoted by #, $s \in Q$ is the start state, $S \in \Gamma$ is the start pushdown symbol, $F \subseteq Q$ is a set of finite states. Sets $\mathbb{N}$, Q, Γ and Σ_O are pairwise disjoint.

$R \subseteq (\mathbb{N}^+ \times Q \times (\Gamma - (\Sigma_I \cup \{\#\})) \times Q \times (\Gamma - \{\#\})^+ \times (\Sigma_O \cup \{\varepsilon\})) \bigcup (\mathbb{N}^+ \times Q \times (\Gamma - (\Sigma_I \cup \{\#\})) \times Q \times (\Gamma - \{\#\})^* \times \Sigma_O) \bigcup (\mathbb{N}^+ \times Q \times \{\#\} \times Q \times ((\Gamma - \{\#\})^* \{\#\}) \times (\Sigma_O \cup \{\varepsilon\})) \bigcup (\mathbb{N}^+ \times Q \times \{\#\} \times Q \times (((\Gamma - \{\#\})^* \{\#\}) \cup \{\varepsilon\}) \times \Sigma_O)$. Instead of $(m,p,A,q,w,a') \in R$, where $m \leq d$, $p,q \in Q$, $A \in \Gamma - \Sigma_I$, $w \in \Gamma^+$, $a' \in \Sigma_O \cup \{\varepsilon\}$ we usually write $r = mpA \to qwa'$ and call r a *rule*.

A *configuration* of the deep pushdown transducer ${}_dM^T$ is a 4-tuple $Q \times (\Sigma_I)^* \times (\Gamma - \{\#\})^* \{\#\} \times (\Sigma_O)^*$. Let χ be a set of all configurations of deep pushdown transducer ${}_dM^T$ and let $x,y \in \chi$ be two configurations. $x \vdash y$ is a move between these configuration. If $x = (p,au,az,b), y = (q,u,z,bc)$, where $p,q \in Q, a \in \Sigma_I, u \in (\Sigma_I)^*, z \in \Gamma^*, b,c \in (\Sigma_O)^*$, then ${}_dM^T$ *pops* its pushdown from x to y, $x\ {}_p\vdash y$. ${}_dM^T$ *expands* its pushdown if $x = (p,au,wAz,b), y = (q,au,wvz,bc), r = p(A) \to q(v)c \in R$ accordingly to the rule r, symbolically written $x\ {}_e\vdash y[p(A) \to q(v)c]$ or $x\ {}_e\vdash y$ if there is only one usable rule. In the standard manner we can extend ${}_p\vdash$, ${}_e\vdash$ and $\vdash$ to ${}_p\vdash^m$, ${}_e\vdash^m$ and $\vdash^m$, respectively, for $m \geq 0$. Then based on ${}_p\vdash^m$, ${}_e\vdash^m$ and $\vdash^m$, define ${}_p\vdash^+$, ${}_p\vdash^*$, ${}_e\vdash^+$, ${}_e\vdash^*$, $\vdash^+$ and $\vdash^*$.

We can define a *language accepted by* ${}_dM^T$ as $L^{Acc}({}_dM^T) = \{w \in (\Sigma_I)^* : (s,w,S\#,\varepsilon) \vdash^* (f,\varepsilon,\#,u) \in {}_dM^T$ with $f \in F, u \in (\Sigma_O)^*\}$ and a *language generated by* ${}_dM^T$ as $L^{Gen}({}_dM^T) = \{u \in (\Sigma_O)^* : (s,w,S\#,\varepsilon) \vdash^* (f,\varepsilon,\#,u) \in {}_dM^T$ with $f \in F, w \in (\Sigma_I)^*\}$. In the same manner we can define a *language accepted by* ${}_dM^T$ *by empty pushdown* as $L^{Acc}_{emptyPD}({}_dM^T) = \{w \in (\Sigma_I)^* : (s,w,S\#,\varepsilon) \vdash^* (q,\varepsilon,\#,u) \in {}_dM^T$ with $u \in (\Sigma_O)^*\}$, a *language generated by* ${}_dM^T$ *by empty pushdown* as $L^{Gen}_{emptyPD}({}_dM^T) = \{u \in (\Sigma_O)^* : (s,w,S\#,\varepsilon) \vdash^* (q,\varepsilon,\#,u) \in {}_dM^T$ with $w \in (\Sigma_I)^*\}$, a *language accepted by* ${}_dM^T$ *by entering the final state* as $L^{Acc}_{final}({}_dM^T) = \{w \in (\Sigma_I)^* : (s,w,S\#,\varepsilon) \vdash^* (f,\varepsilon,v\#,u) \in {}_dM^T$ with $f \in F, v \in \Gamma^*, u \in (\Sigma_O)^*\}$ and a *language generated by* ${}_dM^T$ *by entering the final state* as $L^{Gen}_{final}({}_dM^T) = \{u \in (\Sigma_O)^* : (s,w,S\#,\varepsilon) \vdash^* (f,\varepsilon,v\#,u) \in {}_dM^T$ with $f \in F, v \in \Gamma^*, w \in (\Sigma_I)^*\}$.

According to theorems presented in [8] we can assume $L^{Acc}({}_dM^T) = L^{Acc}_{emptyPD}({}_dM^T)$.

Let $\vartheta({}_dM^T)$ be a *translation* ϑ defined by deep pushdown transducer ${}_dM^T$, $\vartheta({}_dM^T) = \{(x,y) : (s,x,S,\varepsilon) \vdash^* (q,\varepsilon,z,y), x \in \Sigma_I, y \in \Sigma_O, z \in (\Gamma - \{\#\})^* \{\#\}, q \in Q\}$ and $\vartheta_e({}_dM^T)$ a *translation* ϑ defined by deep pushdown transducer with empty pushdown ${}_dM^T$, $\vartheta({}_dM^T) = \{(x,y) : (s,x,S,\varepsilon) \vdash^* (q,\varepsilon,\varepsilon,y), x \in T_I, y \in T_O\}$

4 RESULTS

Theorem 1 *Let $\tau = (V,W,T_I,T_O,P,S)$ be a state translation scheme. Then there exists deep pushdown transducer ${}_dM^T = (Q, \Sigma_I, \Gamma, \Sigma_O, R, s, S, F)$ such $\vartheta({}_dM^T) = \vartheta(\tau)$.*

Proof Let $\tau = (V,W,T_I,T_O,P,S)$ be a state translation scheme. Without loss of generality we can assume that $T_I \cap T_O = \emptyset$. Set $\Sigma_I = T_I$, $\Sigma_O = T_O$, $N = V - (T_I \cup T_O)$ and $\Gamma = V \cup \{\#\}$. Further define homomorphism ξ over $(\{\#\} \cup V)^*$ as $\xi(A) = A$ for each $A \in \{\#\} \cup N$ and $h(a) = \varepsilon$ for each $a \in V - N$.

Then we can construct deep pushdown transducer ${}_dM^T = (Q, \Sigma_I, \Gamma, \Sigma_O, R, s, S, F)$, where

$Q = \{s\} \cup \{\langle p,u\rangle : p \in W, u \in prefix(N^*\{\#\}^d, d), |u| \le d\}$,

$F = \{\langle p,u\rangle : p \in W, u \in prefix(\{\#\}^d, d), |u| \le d\}$

and R is built using the following steps:

1. set $R = \emptyset$
2. add $1\, sS \to \langle p,S\rangle S\varepsilon$ to R for each rule $pS \to qx, y \in P$, $p,q \in W$, $x \in \Sigma_I^+$, $y \in \Sigma_O^*$
3. add $|uA|\, \langle p,uAv\rangle A \to \langle q, prefix(u\xi(x)v,d)\rangle x_0y_0B_1x_1y_1 \ldots B_kx_ky_k\varepsilon$ to R for each $pA \to qx, y \in P$, $x = x_0B_1x_1 \ldots B_kx_k$, $y = y_0B_1y_1 \ldots B_ky_k$, $p,q \in W$, $k \ge 0$, $1 \le j \le k$, $0 \le i \le k$, $A,B_j \in N, \langle p,uAv\rangle \in Q, x_i \in \Sigma_I^*, y_i \in \Sigma_O^*, u \in N^*, v \in N^*\{\#\}^*, |uAv| = d, p \notin {}_\tau states(u)$
4. add $|uA|\, \langle p,uv\rangle A \to \langle q,uAv\rangle A\varepsilon$ to R for each $A \in N, p \in W, u \in N^*, v = \{\#\}^*, |uv| = d-1, p \notin {}_G states(u)$
5. add $1\, qa \to q\varepsilon\varepsilon$ to R for each $a \in \Sigma_I$ and each $q \in Q$
6. add $1\, q\varepsilon \to q\varepsilon b$ to R for each $b \in \Sigma_O$ and each $q \in Q$

Basic idea: Deep pushdown transducer ${}_dM^T$ simulates n-limited derivations in τ. It always records the first d nonterminals occuring in the current sentential form. If there appear less than d nonterminals in the sentential form, it completes them to d with symbols #). Transducer ${}_dM^T$ simulates a derivation step in the pushdown and records the newly generated nonterminals in the state. When scheme τ successfully completes the generation of terminal strings, transducer ${}_dM^T$ completes reading input string and generating output string, empties its pushdown and enters the final state.

Theorem 2 *Let ${}_dM^T = (Q, \Sigma_I, \Gamma, \Sigma_O, R, s, S, F)$ be a deep pushdown transducer. Then there exists state translation scheme $\tau = (V, W, T_I, T_O, P, S)$ such $\vartheta(\tau) = \vartheta({}_dM^T)$.*
Proof Basic idea: State translation scheme τ simulates the application of a deep pushdown transducer rule using left-to-right scan of the sentential form until it reaches the *i*th occurrence of nonterminal. If this occurrence equals to nonterminal on the left-hand side of the simulated rule, it replaces this with its right-hand side. Then it returns to the beginning of the sentential form.

5 CONCLUSION

In this paper, there were presented state translation schemes and deep pushdown transducers as two possible models used in syntax-directed translation of languages generated by (*n*-limited) state grammars. These models work exactly like models they are based on but have mechanisms to generate an output sentential form. In this paper, there were presented main ideas of conversions between these models.

There are several open problems regarding to this paper. The presented deep pushdown transducers work nondeterministically. This problem prevents the use in practice where it is crucial to use deterministic models. We considered only a true pushdown expansion and so presented transducers do not allow replacing pushdown symbols with an empty string. What will be the language family accepted by the transducer with *erasing rules*? Also languages generated by presented transducers and translation schemes should be more investigated and classified.

ACKNOWLEDGEMENT

This work was partially supported by the BUT IGA grant FIT-S-14-2299 Research and application of advanced methods in ICT.

REFERENCES

[1] Aho, A. V., Ullman, J. D.: *The Theory of Parsing, Translation and Compiling, Volume I: Parsing*, Prentice Hall, Englewood Clifs, New Jersey, 1972, ISBN 0139145567

[2] Autebert, J., Berstel, J., Boasson, L.: Context-free languages and pushdown automata. In: Rozenberg, G., Salomaa, A., (eds.) *Handbook of Formal Languages*, vol. 1. Springer, 1997, ISBN 978-3540604204

[3] Dassow, J., Paŭn, G.: *Regulated Rewriting in Formal Language Theory.* AkademieVerlag, Berlin, 1989, ISBN 978-0387514147

[4] Gurari, E.: *An Introduction to the Theory of Computation*, Computer Science Press, 1989, ISBN 0-7167-8182-4

[5] Kasai, T.: An hierarchy between context-free and context-sensitive languages. In: *Journal of Computer and System Sciences* vol. 4, pp. 492–508, 1970, ISSN 0022-0000

[6] Meduna, A.: *Automata and Languages: Theory and Applications.* Springer, London, 2000, ISBN 978-1852330743

[7] Meduna, A.: Deep Pushdown Automata. In: *Acta informatica*, vol. 98, pp. 114–124, 2006, ISSN 0001-5903

[8] Solár, P.: Deep Pushdown Transducers and Parallel Deep Pushdown Transducers. In: *Proceedings of the 19th Conference STUDENT EEICT 2013*, pp. 207-211, VUT v Brně, Brno, 2013, ISBN 978-80-214-4695-3