5-2018

Pushdown Automata Correspond to Context Free Grammars

Doug Baldwin
SUNY Geneseo, baldwin@geneseo.edu

Pushdown Automata Correspond to Context Free Grammars

Doug Baldwin
Department of Mathematics, SUNY Geneseo

May 2018

The main relationship between context free grammars and pushdown automata is that both models define the same languages. In other words, every context free language is accepted by some pushdown automaton, and every pushdown automaton accepts a context free language. This document proves part of this claim for the pushdown automaton model used in Maheshwari and Smid's text [2]. Specifically, the following pages prove that every pushdown automaton in that model accepts a context free language.

1 Preliminaries

The main claim is...

Theorem 1. If $P = (\Sigma, \Gamma, Q, \delta, q_0)$ is a pushdown automaton, then $L(P)$ is context free.

Proof. As detailed in the following sections, a context free grammar G can be constructed from P such that $L(G) = L(P)$. Since every context free grammar generates a context free language, the existence of G proves that $L(P)$ is context free. □

Theorems similar to Theorem 1 appear in almost all automata theory texts [1, 3]. What makes the proof for Maheshwari and Smid's automata more complicated than those other proofs is that Maheshwari and Smid allow automata to make transitions that read the input tape without advancing it (i.e., transitions with tape action "N"). Such transitions allow automata to "peek" at an input symbol and make decisions based on it while also saving it to use in picking later transitions. This ability to peek at the next input symbol is unusual in

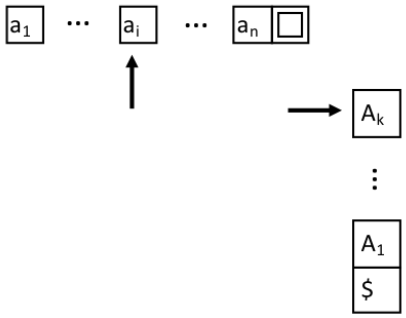

Figure 1: A pushdown automaton configuration.

pushdown automata, but it is widely use in parsing context free languages, and is called "lookahead." In Maheshwari and Smid's automata, lookahead happens whenever an automaton makes a transition $raA \to s\mathrm{N}X$. From the moment the automaton makes such a transition until it makes its next transition with tape action "R," the tape head is stationary over the a, and so the automaton cannot make transitions that require any other tape symbol. It is sometimes convenient to say that the automaton "has" lookahead a, or is computing "with" lookahead a, in this situation. Between a transition with tape action "R" and the next transition with action "N" the automaton has no lookahead; it can be helpful to think of the lookahead being the empty string ϵ in this situation. This convention allows us to treat lookahead as a string of either 1 or 0 symbols, and there are places in the following that take advantage of that treatment.

Maheshwari and Smid use a graphical presentation of pushdown automaton configurations, as in Figure 1. Because the proof of Theorem 1 needs to talk at length about configurations, it is helpful to also have a textual notation for them. This document uses the notation (r, w, X) to indicate a configuration in which the automaton is in state r, has string w at and to the right of the tape head, and has string X on its stack. If there is a lookahead symbol, it will be the first symbol of w; the blank at the end of w is normally not shown. The first symbol of X is at the top of the stack. Thus, for example, if the automaton in Figure 1 were in state q_3, its textual configuration would be $(q_3, a_i \ldots a_n, A_k \ldots A_1\$)$.

A textual notation for configurations also provides a concise notation for summarizing computations. The notation $(r, w, X) \vdash (s, u, Y)$ means that an automaton can move in a single transition from configuration (r, w, X) to configuration (s, u, Y). Similarly, $(r, w, X) \vdash^* (s, u, Y)$ (note the "*") means the automaton can move from (r, w, X) to (s, u, Y) in zero or more transitions.

Sections 2 and 3 are the full proof of Theorem 1. Figure 2 is a "roadmap" that may clarify how the details in those sections contribute to the overall result. Section 2 explains how to construct grammar G to generate the strings that P

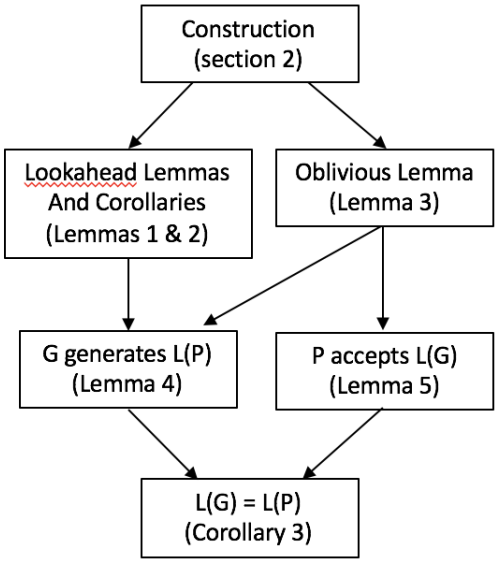

Figure 2: A "roadmap" to the proof.

accepts, without actually proving that G does that. Section 3 presents the proof, via five lemmas whose results are tied together by a final corollary. Two of the lemmas, the "Lookahead Lemmas," help reason about how G describes lookahead in P. The "Oblivious Lemma" says that computations in P cannot depend on information too far away on the input tape or too far down the stack; this helps decompose computations by P into parts that correspond to variables in G. Lemmas 4 and 5 are the heart of the proof. The first basically says (although the exact phrasing is more general) that every string generated by G is accepted by P, while the second says that every string accepted by P is generated by G. The main theorem follows directly from these results, in Corollary 3.

2 The Construction

Because a pushdown automaton must empty its stack in order to accept its input, the process of shrinking the stack by one symbol is a key step towards accepting input. But this is a "process," not just a single transition, because in one transition an automaton can both remove the top symbol from the stack and add any number of new symbols in its place. The size of the stack only decreases when all of these replacements for a removed symbol have also been

removed. But of course the replacements may also be replaced, so it can take a long time before the automaton makes enough transitions that remove symbols without replacing them (transitions of the forms $raA \to s\mathrm{R}\epsilon$ or $raA \to s\mathrm{N}\epsilon$) to clear away the original symbol and all of its replacements. Eventually, however, this must happen if the automaton is going to accept its input, and when it does the original symbol is said to have been "cleared" from the stack. In general, a computation that clears symbol A from the stack can also continue on and remove symbols below A from the stack. However, there is always a shortest computation that clears A, i.e., a computation that stops as soon as A and the last symbol generated from it have been removed and before anything below A has been used. This notion of a shortest computation to clear A can be stated more precisely as

Definition 1. Let (r, wu, AX) and (s, u, X) be configurations of pushdown automaton P, for some states r and s, strings $w \in \Sigma^*$, $u \in (\Sigma \cup \{\Box\})^*$, and $X \in \Gamma^*$, and stack symbol A. A computation by P that takes it from (r, wu, AX) to (s, u, X) is a *shortest* such computation if it never removes any symbol of X from the stack.

Realizing that clearing symbols from the stack is central to acceptance provides a way to construct a grammar whose derivations in some sense mimic accepting computations by P. In particular, given $P = (\Sigma, \Gamma, Q, \delta, q_0)$, let grammar $G = (V, \Sigma, R, S)$, where the variable set V consists of start symbol S and a collection of variables denoted V_{qaArb}. In each variable V_{qaArb}, q and r are states from Q, a and b are elements of $\Sigma \cup \{\Box, \epsilon\}$, and A is a stack symbol from Γ. Intuitively, V_{qaArb} represents the strings that P reads from its tape during a shortest computation that clears A from the stack and takes the automaton from state q with lookahead symbol a to state r with lookahead b. If the automaton has no lookahead in state q or r then a or b is ϵ.

The rules of the grammar, R, are as follows:

1. For every transition in P of the form $qaA \to s_1\mathrm{R}B_n \dots B_2B_1$, with $a \neq \Box$, there are rules of the forms

$$\mathrm{V}_{qaArb} \to a\mathrm{V}_{s_1\epsilon B_1 s_2 b_2}\mathrm{V}_{s_2 b_2 B_2 s_3 b_3} \dots \mathrm{V}_{s_n b_n B_n r b} \quad \text{and}$$
$$\mathrm{V}_{q\epsilon Arb} \to a\mathrm{V}_{s_1\epsilon B_1 s_2 b_2}\mathrm{V}_{s_2 b_2 B_2 s_3 b_3} \dots \mathrm{V}_{s_n b_n B_n r b}$$

   for all states $s_1 \dots s_n$ and r, and all $b_2 \dots b_n, b \in \Sigma \cup \{\Box, \epsilon\}$.

2. For every transition in P of the form $qaA \to s_1\mathrm{N}B_n \dots B_2B_1$, there are rules of the forms

$$\mathrm{V}_{qaArb} \to \mathrm{V}_{s_1 a B_1 s_2 b_2}\mathrm{V}_{s_2 b_2 B_2 s_3 b_3} \dots \mathrm{V}_{s_n b_n B_n r b} \quad \text{and}$$
$$\mathrm{V}_{q\epsilon Arb} \to \mathrm{V}_{s_1 a B_1 s_2 b_2}\mathrm{V}_{s_2 b_2 B_2 s_3 b_3} \dots \mathrm{V}_{s_n b_n B_n r b}$$

   for all states $s_1 \dots s_n$ and r, and all $b_2 \dots b_n, b \in \Sigma \cup \{\Box, \epsilon\}$.

3. For every transition in P of the form $qaA \to r\mathrm{R}\epsilon$, with $a \neq \Box$, there are rules of the forms
$$\mathrm{V}_{qaAr\epsilon} \to a \quad \text{and}$$
$$\mathrm{V}_{q\epsilon Ar\epsilon} \to a$$

4. For every transition in P of the form $qaA \to r\mathrm{N}\epsilon$, there are rules of the forms
$$\mathrm{V}_{qaAra} \to \epsilon \quad \text{and}$$
$$\mathrm{V}_{q\epsilon Ara} \to \epsilon$$

5. Finally, G contains rules of the forms
$$S \to \mathrm{V}_{q_0\epsilon\$r\epsilon} \quad \text{and}$$
$$S \to \mathrm{V}_{q_0\epsilon\$r\Box}$$
for all states r.

3 Correctness

As suggested by the roadmap in Figure 2, the proof that $L(G)$ really does equal $L(P)$ works by showing that each language is a subset of the other. First, however, it's necessary to prove some supporting lemmas. The first two describe how the lookahead symbols embedded in grammar variables actually appear (or not) in strings derived from those variables.

Lemma 1 (First Lookahead Lemma)**.** Let q and r be states, a and b be members of $\Sigma \cup \{\Box\}$, and A be a stack symbol, such that V_{qaArb} derives the empty string. Then $b = a$.

Proof. The proof is by induction on the length of the derivation. (In this and all following proofs, assume for the sake of concreteness that derivations are left-most derivations; since derivations from context free grammars are independent of order, this assumption doesn't affect the generality of the proofs.)

For the base case, suppose the derivation has only one step. The only rule in G that derives ϵ in a single step from a variable of the form V_{qaArb} with $a \in \Sigma \cup \{\Box\}$ is $\mathrm{V}_{qaAra} \to \epsilon$, which has $a = b$.

For the induction, assume that $a = b$ in any derivation $\mathrm{V}_{qaArb} \Rightarrow^* \epsilon$ of at most $k \geq 1$ steps. Consider a derivation of $k + 1$ steps. The first step in this derivation must use a rule of the form

$$\mathrm{V}_{qaArb} \to \mathrm{V}_{s_1aB_1s_2b_2}\mathrm{V}_{s_2b_2B_2s_3b_3} \cdots \mathrm{V}_{s_nb_nB_nrb}.$$

Since the entire derivation produces the empty string in $k+1$ steps, each of the variables on the right side of this rule also derives the empty string, and does so in at most k steps. Thus for every variable $\mathrm{V}_{s_ib_iB_is_{i+1}b_{i+1}}$ where $b_i \neq \epsilon$, the induction hypothesis guarantees that $b_{i+1} = b_i$. And in fact no variable with

$b_i = \epsilon$ can be used in this derivation. (Suppose otherwise, and let $\mathrm{V}_{s_i b_i B_i s_{i+1} b_{i+1}}$ be the leftmost such variable. The induction hypothesis applies to all preceding variables $\mathrm{V}_{s_1 a B_1 s_2 b_2}$ through $\mathrm{V}_{s_{i-1} b_{i-1} B_{i-1} s_i b_i}$, and so $a = b_2 = \ldots = b_i$. But since $a \neq \epsilon$, this also means $b_i \neq \epsilon$, which contradicts the assumption that b_i is ϵ.) Since the induction hypothesis applies to each variable used in the derivation, it follows that $a = b_1 = b_2 = \ldots = b$. □

The First Lookahead Lemma generalizes to certain strings of variables: if

$$\mathrm{V}_{s_1 a_1 A_1 s_2 a_2} \ldots \mathrm{V}_{s_{i-1} a_{i-1} A_{i-1} s_i a_i} \mathrm{V}_{s_i a_i A_i s_{i+1} a_{i+1}} \ldots \mathrm{V}_{s_n a_n A_n s_{n+1} a_{n+1}} \Rightarrow^* \epsilon$$

then $a_1 = a_2 = \ldots = a_{n+1}$. This generalization is useful enough to state and prove formally:

Corollary 1. Let $s_1, \ldots s_{n+1}$ be states in Q, $a_1 \ldots a_{n+1}$ be members of $\Sigma \cup \{\Box\}$, and $A_1 \ldots A_n$ be symbols in Γ such that

$$\mathrm{V}_{s_1 a_1 A_1 s_2 a_2} \ldots \mathrm{V}_{s_{i-1} a_{i-1} A_{i-1} s_i a_i} \mathrm{V}_{s_i a_i A_i s_{i+1} a_{i+1}} \ldots \mathrm{V}_{s_n a_n A_n s_{n+1} a_{n+1}} \Rightarrow^* \epsilon$$

Then $a_1 = a_2 = \ldots = a_{n+1}$.

Proof. The proof is by contradiction. Suppose that the lookahead symbols a_i are not all equal. Then there must be some variable, $\mathrm{V}_{s_j a_j A_j s_{j+1} a_{j+1}}$ for which $a_j \neq a_{j+1}$. But since the entire string of variables derives the empty string, each individual variable must too, and so in particular $\mathrm{V}_{s_j a_j A_j s_{j+1} a_{j+1}} \Rightarrow^* \epsilon$. Now by the First Lookahead Lemma, $a_j = a_{j+1}$, which contradicts the choice of $\mathrm{V}_{s_j a_j A_j s_{j+1} a_{j+1}}$ as having $a_j \neq a_{j+1}$. Thus there can be no such variable, and so $a_1 = a_2 = \ldots = a_{n+1}$. □

Lemma 2 (Second Lookahead Lemma)**.** Let q and r be states, a and b be members of $\Sigma \cup \{\Box\}$, and A be a stack symbol, such that V_{qaArb} derives a non-empty string w. Then the leftmost symbol in w is a.

Proof. The proof is by induction on the number of rewriting steps until the leftmost symbol of w appears.

For the base case, suppose the leftmost symbol appears after one rewriting. Then the first rewriting step must use either a rule of the form

$$\mathrm{V}_{qaArb} \rightarrow a \mathrm{V}_{s_1 \epsilon B_1 s_2 b_2} \mathrm{V}_{s_2 b_2 B_2 s_3 b_3} \ldots \mathrm{V}_{s_n b_n B_n r b}$$

from construction rule 1 or

$$\mathrm{V}_{qaAr\epsilon} \rightarrow a$$

from construction rule 3. In both cases the derived string begins with a.

For the induction step, assume that whenever a derivation $\mathrm{V}_{qaArb} \Rightarrow^* w$ uses at most $k \geq 1$ rewriting steps to produce the leftmost symbol of w, that

symbol is a. Consider a derivation in which the leftmost symbol of w appears after $k+1$ rewritings. This derivation must begin with a step of the form

$$\mathrm{V}_{qaArb} \to \mathrm{V}_{s_1aB_1s_2b_2}\mathrm{V}_{s_2b_2B_2s_3b_3}\ldots\mathrm{V}_{s_nb_nB_nrb}.$$

Since w is non-empty, some one of the variables from the right side, let it be $\mathrm{V}_{s_ib_iB_is_{i+1}b_{i+1}}$, derives the leftmost symbol of w; the prefix $\mathrm{V}_{s_1aB_1s_2b_2}\ldots\mathrm{V}_{s_{i-1}b_{i-1}B_{i-1}s_ib_i}$ derives the empty string. If that prefix isn't empty, $a = b_2 = \ldots = b_i$, by the corollary to the First Lookahead Lemma, while if the prefix is empty then $i = 1$ and so b_i is actually a. In either case the variable that derives the leftmost symbol of w is of the form $\mathrm{V}_{s_iaB_is_{i+1}b_{i+1}}$. This variable derives the leftmost symbol of w in at most k steps and $a \in \Sigma \cup \{\Box\}$, so by the induction hypothesis the leftmost symbol of w is a. □

Like the First Lookahead Lemma, the second also generalizes to strings, and the generalization is again worth proving:

Corollary 2. Let $s_1, \ldots s_{n+1}$ be states in Q, $a_1 \ldots a_{n+1}$ be members of $\Sigma \cup \{\Box\}$, and $A_1 \ldots A_n$ be symbols in Γ such that

$$\mathrm{V}_{s_1a_1A_1s_2a_2}\ldots\mathrm{V}_{s_{i-1}a_{i-1}A_{i-1}s_ia_i}\mathrm{V}_{s_ia_iA_is_{i+1}a_{i+1}}\ldots\mathrm{V}_{s_na_nA_ns_{n+1}a_{n+1}} \Rightarrow^* w$$

for some non-empty string w. Then the leftmost symbol in w is a_1.

Proof. Since w is non-empty, some variable, let it be $\mathrm{V}_{s_ja_jA_js_{j+1}a_{j+1}}$, derives the leftmost symbol of w, while the prefix $\mathrm{V}_{s_1a_1A_1s_2a_2}\ldots\mathrm{V}_{s_{j-1}a_{j-1}A_{j-1}s_ja_j}$ derives the empty string. If this prefix isn't empty, then $a_j = a_1$ from the corollary to the First Lookahead Lemma. If the prefix is empty, then $j = 1$. In either case the variable that derives the leftmost symbol of w is of the form $\mathrm{V}_{s_ia_1A_is_{i+1}a_{i+1}}$, which by the Second Lookahead Lemma derives a string that begins with a_1. □

The next lemma formalizes an intuitively sensible idea, namely that computations by pushdown automata aren't influenced by input beyond that examined by the computation, or by stack contents below those examined by the computation. In other words, pushdown automata are oblivious to such extra information, and, for that matter, to whether it exists at all.

Lemma 3 (Oblivious Lemma)**.** Let q and r be states in Q, w be a string in Σ^*, a be a symbol in $\Sigma \cup \{\Box\}$, u be a string from $(\Sigma \cup \{\Box\})^*$, A be a symbol from Γ, and X be a string from Γ^*. Furthermore, suppose P has a shortest computation that takes it from configuration (q, wau, AX) to configuration (r, au, X). Then P also has shortest computations that take it from (q, wav, AY) to (r, av, Y) for all strings $v \in (\Sigma \cup \{\Box\})^*$ and $Y \in \Gamma^*$.

Proof. The computation that takes P from (q, wau, AX) to (r, au, X) can't depend on any part of u, because to do so it would have to move past at least a on the tape. Further, because this is a shortest computation, it can't depend

on any part of X. Therefore, the computation would proceed in exactly the same fashion if there were different strings, v and Y, following a and below A. Because the computation proceeds in exactly the same fashion, in particular it never removes any symbol of Y from the stack, and so is still a shortest computation. □

We now turn to the main lemmas of the proof. The first essentially says that any string derived by G corresponds to a computation in P:

Lemma 4. Let q and r be states in P, a and b be elements of $\Sigma \cup \{\Box, \epsilon\}$, A be a symbol from Γ, and w be a string in Σ^*, such that $\mathrm{V}_{qaArb} \Rightarrow^* w$ in G. Then $(q, wb, A) \vdash^* (r, b, \epsilon)$ in P.

Proof. The proof is by induction on the length of the derivation of w.

For the base case, suppose the derivation has only one step. Two parts of the construction generate rules that can directly derive strings of terminals, namely parts 3 and 4. We consider each separately:

1. Rules of the forms
$$\mathrm{V}_{qcAr\epsilon} \to c \quad \text{and}$$
$$\mathrm{V}_{q\epsilon Ar\epsilon} \to c$$
from part 3, created from transitions of the form $qcA \to r\mathrm{R}\epsilon$. With one of these rules as the only step in the derivation of w, w must equal c. Furthermore, from configuration $(q, wb, A) = (q, cb, A)$, P can take the transition to reach configuration (r, b, ϵ), and so $(q, wb, A) \vdash^* (r, b, \epsilon)$.

2. Rules of the forms
$$\mathrm{V}_{qcArc} \to \epsilon \quad \text{and}$$
$$\mathrm{V}_{q\epsilon Arc} \to \epsilon$$
from part 4, created from transitions of the form $qcA \to r\mathrm{N}\epsilon$. In order to use either of these rules in a derivation of the form $\mathrm{V}_{qaArb} \Rightarrow^* w$, c must equal b and w must equal ϵ. Applying these equalities to configuration (q, wb, A) in P yields $(q, wb, A) = (q, \epsilon c, A) = (q, c, A)$. From this configuration, the transition $qcA \to r\mathrm{N}\epsilon$ takes the automaton to configuration $(r, c, \epsilon) = (r, b, \epsilon)$, and so $(q, wb, A) \vdash^* (r, b, \epsilon)$.

For the induction step, assume that whenever $\mathrm{V}_{qaArb} \Rightarrow^* w$ in at most $k \geq 1$ steps in G, $(q, wb, A) \vdash^* (r, b, \epsilon)$ in P. Consider how derivations of $k + 1$ steps must proceed. Once again there are two possibilities, corresponding to parts 1 and 2 in the construction:

1. The derivation begins by using a rule from part 1, either
$$\mathrm{V}_{qaArb} \to a\mathrm{V}_{s_1\epsilon B_1 s_2 b_2}\mathrm{V}_{s_2 b_2 B_2 s_3 b_3} \ldots \mathrm{V}_{s_n b_n B_n r b}$$

or

$$\mathrm{V}_{q\epsilon Arb} \to a\mathrm{V}_{s_1\epsilon B_1 s_2 b_2}\mathrm{V}_{s_2 b_2 B_2 s_3 b_3}\ldots\mathrm{V}_{s_n b_n B_n r b}.$$

Both of these rules imply that w is of the form $w = aw_1w_2\ldots w_n$ where $\mathrm{V}_{s_i b_i B_i s_{i+1} b_{i+1}} \Rightarrow^* w_i$ (taking b_1 to be ϵ, s_{n+1} to be r, and b_{n+1} to be b). The derivation of w then proceeds as

$$\begin{array}{rcll} \mathrm{V}_{qaArb} & \to & a\mathrm{V}_{s_1\epsilon B_1 s_2 b_2}\mathrm{V}_{s_2 b_2 B_2 s_3 b_3}\ldots\mathrm{V}_{s_n b_n B_n r b} & (1) \\ & \Rightarrow^* & aw_1\mathrm{V}_{s_2 b_2 B_2 s_3 b_3}\ldots\mathrm{V}_{s_n b_n B_n r b} & \\ & \ldots & & \\ & \Rightarrow^* & aw_1w_2\ldots w_n = w & \end{array}$$

Now consider P in configuration $(q, wb, A) = (q, aw_1w_2\ldots w_nb, A)$. The rule used in step (1) of the derivation corresponds to a transition $qaA \to s_1\mathrm{R}B_n\ldots B_2B_1$, a transition that takes P from $(q, aw_1w_2\ldots w_nb, A)$ to $(s_1, w_1w_2\ldots w_nb, B_1B_2\ldots B_n)$. From this configuration, P begins a series of computations corresponding to the variables $\mathrm{V}_{s_i b_i B_i s_{i+1} b_{i+1}}$. Specifically, each of the variables has a derivation $\mathrm{V}_{s_i b_i B_i s_{i+1} b_{i+1}} \Rightarrow^* w_i$, which takes at most k steps. Thus by the induction hypothesis and the Oblivious Lemma, P would have computations

$$(s_i, w_iw_{i+1}\ldots w_nb, B_iB_{i+1}\ldots B_n) \vdash^* (s_{i+1}, w_{i+1}\ldots w_nb, B_{i+1}\ldots B_n)$$

if the symbol following each w_i in $w_{i+1}\ldots w_nb$ were b_{i+1}. To see that it is, consider the possible values for b_{i+1}: if $b_{i+1} = \epsilon$, then w_{i+1} can be written as $w_{i+1} = \epsilon w_{i+1}$ and so w_i is trivially followed by b_{i+1}. If $b_{i+1} \neq \epsilon$, then the Lookahead Lemmas and their corollaries guarantee that either $w_{i+1}\ldots w_n$ is non-empty and begins with b_{i+1}, or that $w_{i+1}\ldots w_n$ is empty but $b_{i+1} = b$. In both cases, w_i is in fact followed by b_{i+1}, and so P does indeed have a computation that takes it from configuration $(s_i, w_iw_{i+1}\ldots w_nb, B_iB_{i+1}\ldots B_n)$ to $(s_{i+1}, w_{i+1}\ldots w_nb, B_{i+1}\ldots B_n)$. The collective effect of these computations is

$$\begin{array}{rcl} (q, wb, A) & \vdash & (s_1, w_1w_2\ldots w_nb, B_1B_2\ldots B_n) \\ & \vdash^* & (s_2, w_2\ldots w_nb, B_2\ldots B_n) \\ & \vdash^* & \ldots \\ & \vdash^* & (r, b, \epsilon). \end{array}$$

2. The derivation begins by using a rule from part 2, either

$$\mathrm{V}_{qaArb} \to \mathrm{V}_{s_1 a B_1 s_2 b_2}\mathrm{V}_{s_2 b_2 B_2 s_3 b_3}\ldots\mathrm{V}_{s_n b_n B_n r b}$$

or

$$\mathrm{V}_{q\epsilon Arb} \to \mathrm{V}_{s_1 a B_1 s_2 b_2}\mathrm{V}_{s_2 b_2 B_2 s_3 b_3}\ldots\mathrm{V}_{s_n b_n B_n r b}.$$

These rules in G correspond to a transition in P of the form

$$qaA \to s_1\mathrm{N}B_1B_2\ldots B_n. \qquad (2)$$

We can then write w as $w = w_1 w_2 \ldots w_n$ where $\mathrm{V}_{s_i b_i B_i s_{i+1} b_{i+1}} \Rightarrow^* w_i$ (letting $b_1 = a$, $s_{n+1} = r$, and $b_{n+1} = b$) and reason from the Lookahead Lemmas and their corollaries that $w_1 w_2 \ldots w_n b$ begins with a, either because the first non-empty w_i is derived from $\mathrm{V}_{s_i a B_i s_{i+1} b_{i+1}}$ or because $w_1 w_2 \ldots w_n$ is empty and $a = b$. Thus P can take transition (2) from configuration (q, wb, A) to configuration $(s_1, wb, B_1 B_2 \ldots B_n) = (s_1, w_1 w_2 \ldots w_n b, B_1 B_2 \ldots B_n)$. Now just as in case 1, for each derivation $\mathrm{V}_{s_i b_i B_i s_{i+1} b_{i+1}} \Rightarrow^* w_i$, there is a computation

$$(s_i, w_i w_{i+1} \ldots w_n b, B_i B_{i+1} \ldots B_n) \vdash^* (s_{i+1}, w_{i+1} \ldots w_n b, B_{i+1} \ldots B_n)$$

in P. Continuing as in case 1, the collective effect of the initial transition and these computations is $(q, wb, A) \vdash^* (r, b, \epsilon)$.

This completes the inductive argument that for all derivations in G of the form $\mathrm{V}_{qaArb} \Rightarrow^* w$, there is a series of transitions in P through which $(q, wb, A) \vdash^* (r, b, \epsilon)$. □

The second main lemma says that certain computations in P correspond to strings derived by G:

Lemma 5. Let q and r be states, b be a symbol in $\Sigma \cup \{\square\}$, w be a string from Σ^*, a be the leftmost symbol in wb, and A be a symbol in Γ such that $(q, wb, A) \vdash^* (r, b, \epsilon)$ in P. Then if this computation by P ends with a lookahead symbol (which must be b), $\mathrm{V}_{qaArb} \Rightarrow^* w$ and $\mathrm{V}_{q\epsilon Arb} \Rightarrow^* w$ in G; if the computation by P does not end with a lookahead symbol, $\mathrm{V}_{qaAr\epsilon} \Rightarrow^* w$ and $\mathrm{V}_{q\epsilon Ar\epsilon} \Rightarrow^* w$ in G.

Proof. The proof is by induction on the number of transitions taken by P during the computation $(q, wb, A) \vdash^* (r, b, \epsilon)$.

For the base case, suppose the computation involves only one transition. Since a transition moves the tape head past either 0 or 1 input positions, there are two possibilities for this transition:

1. A transition of the form $qbA \to r\mathrm{N}\epsilon$ that moves the tape head 0 positions. Since this transition does not move the tape head, it ends with lookahead symbol b, and the tape configuration after the transition (i.e., b) is the same as the tape configuration before (wb), so $w = \epsilon$. Thus a, the leftmost symbol in wb, must be b. Corresponding to transition $qbA \to r\mathrm{N}\epsilon$, step 4 in the construction created rules $\mathrm{V}_{qbArb} \to \epsilon$ and $\mathrm{V}_{q\epsilon Arb} \to \epsilon$. Thus, we have a transition with lookahead symbol b, and derivations $\mathrm{V}_{qbArb} = \mathrm{V}_{qaArb} \Rightarrow^* \epsilon = w$ and $\mathrm{V}_{q\epsilon Arb} \Rightarrow^* w$.

2. A transition of the form $qaA \to r\mathrm{R}\epsilon$ that moves the tape head 1 position. Since this transition moves the tape head past a, the tape configuration before the transition must consist of a followed by the tape configuration after the transition, i.e., $wb = ab$ and so $w = a$. Furthermore, step 3 in

the construction of G created rules $\mathrm{V}_{qaAr\epsilon} \to a$ and $\mathrm{V}_{q\epsilon Ar\epsilon} \to a$ from the transition. Thus we have a transition with no lookahead, and $\mathrm{V}_{qaAr\epsilon} \Rightarrow^* a = w$ and $\mathrm{V}_{q\epsilon Ar\epsilon} \Rightarrow^* w$.

For the induction step, suppose that the lemma holds whenever P has a computation $(q, wb, A) \vdash^* (r, b, \epsilon)$ with at most $k \geq 1$ transitions. Consider a computation $(q, wb, A) \vdash^* (r, b, \epsilon)$ with $k + 1$ transitions. Because every transition in a pushdown automaton removes a symbol from the stack, the first of these transitions must replace A with some other symbol or symbols in order for there to be something for the subsequent transitions to remove. As in the base case, such a first transition can move the tape head either 0 positions or 1 position:

1. Suppose P moves the tape head 0 positions, i.e., the first transition is of the form $qaA \to s_1 \mathrm{N} B_n \dots B_2 B_1$. The configuration resulting from this transition is $(s_1, wb, B_1 B_2 \dots B_n)$, and the automaton has lookahead symbol a. Corresponding to the transition, construction step 2 created rules
$$\begin{array}{l} \mathrm{V}_{qaArc} \to \mathrm{V}_{s_1 a B_1 s_2 b_2} \mathrm{V}_{s_2 b_2 B_2 s_3 b_3} \dots \mathrm{V}_{s_n b_n B_n r c} \quad \text{and} \\ \mathrm{V}_{q\epsilon Arc} \to \mathrm{V}_{s_1 a B_1 s_2 b_2} \mathrm{V}_{s_2 b_2 B_2 s_3 b_3} \dots \mathrm{V}_{s_n b_n B_n r c} \end{array} \tag{3}$$
in G, for all states s_1 through s_n and all lookahead values c and b_2 through b_n in $\Sigma \cup \{\square, \epsilon\}$. These rules can begin two derivations in G, namely
$$\begin{array}{l} \mathrm{V}_{qaArc} \Rightarrow \mathrm{V}_{s_1 a B_1 s_2 b_2} \mathrm{V}_{s_2 b_2 B_2 s_3 b_3} \dots \mathrm{V}_{s_n b_n B_n r c} \quad \text{and} \\ \mathrm{V}_{q\epsilon Arc} \Rightarrow \mathrm{V}_{s_1 a B_1 s_2 b_2} \mathrm{V}_{s_2 b_2 B_2 s_3 b_3} \dots \mathrm{V}_{s_n b_n B_n r c} \end{array} \tag{4}$$
In order to eventually reach (r, b, ϵ), the automaton must clear symbols B_1 through B_n from the stack. As it clears each, it also moves through part of w, so we can write w as $w = w_1 w_2 \dots w_n$ such that w_i is the part of w that the automaton moves its tape head past during the shortest computation that clears B_i from the stack. More formally, for each i, there are states s_i and s_{i+1} (with $s_{n+1} = r$) such that
$$(s_i, w_i w_{i+1} \dots w_n, B_i B_{i+1} \dots B_n) \vdash^* (s_{i+1}, w_{i+1} \dots w_n, B_{i+1} \dots B_n) \tag{5}$$
via a shortest computation. Because construction step 2 creates rules (3) for *all* states, there must be one such pair of rules in which, for every index i, the states that subscript $\mathrm{V}_{s_i b_i B_i s_{i+1} b_{i+1}}$ exactly match the states in the computation
$$(s_i, w_i w_{i+1} \dots w_n b, B_i B_{i+1} \dots B_n) \vdash^* (s_{i+1}, w_{i+1} \dots w_n b, B_{i+1} \dots B_n).$$
The same is also true of the lookahead symbols, because construction step 2 creates rules (3) for all combinations of states *and lookahead symbols*, so we can simply pick b_i to be the leftmost symbol in $w_i w_{i+1} \dots w_n b$. This choice ensures that if the computation that reads past w_i ends with

a lookahead symbol, that symbol must be b_{i+1}. So we now know that P clears the B_i from the stack in a series of computations of form (5), and that there is a pair of rules of form (3) whose variables are subscripted with exactly the states and (potential) lookahead symbols of those computations. Each of these computations uses at most k transitions. Furthermore, for each of these computations, the Oblivious Lemma says that the same computation takes P from configuration $(s_i, w_i b_{i+1}, B_i)$ to $(s_{i+1}, b_{i+1}, \epsilon)$. Thus, by the induction hypothesis G must have derivations $\mathrm{V}_{s_i b_i B_i s_{i+1} b_{i+1}} \Rightarrow^* w_i$ if the computation by P uses b_{i+1} as lookahead, and $\mathrm{V}_{s_1 b_i B_i s_{i+1} \epsilon} \Rightarrow^* w_i$ if it doesn't. Thus the derivations begun in (4) continue by rewriting the variables on the right, i.e.,

$$\begin{array}{ll} \mathrm{V}_{qaArc} & \Rightarrow \mathrm{V}_{s_1 a B_1 s_2 b_2} \mathrm{V}_{s_2 b_2 B_2 s_3 b_3} \ldots \mathrm{V}_{s_n b_n B_n r c} \\ & \Rightarrow^* w_1 \mathrm{V}_{s_2 b_2 B_2 s_3 b_3} \ldots \mathrm{V}_{s_n b_n B_n r c} \\ & \ldots \\ & \Rightarrow^* w_1 w_2 \ldots w_n = w \\ \mathrm{V}_{q\epsilon Arc} & \Rightarrow \mathrm{V}_{s_1 a B_1 s_2 b_2} \mathrm{V}_{s_2 b_2 B_2 s_3 b_3} \ldots \mathrm{V}_{s_n b_n B_n r c} \\ & \Rightarrow^* w_1 \mathrm{V}_{s_2 b_2 B_2 s_3 b_3} \ldots \mathrm{V}_{s_n b_n B_n r c} \\ & \ldots \\ & \Rightarrow^* w_1 w_2 \ldots w_n = w \end{array} \tag{6}$$

The computation for the final step is $(s_n, w_n b, B_n) \vdash^* (r, b, \epsilon)$ and the corresponding derivation is $\mathrm{V}_{s_n b_n B_n r c} \Rightarrow^* w_n$. If the computation uses b as lookahead, then from the instance of (3) in which $c = b$ there are complete derivations of w

$$\begin{array}{ll} \mathrm{V}_{qaArb} \Rightarrow^* w & \text{and} \\ \mathrm{V}_{q\epsilon Arb} \Rightarrow^* w. & \end{array}$$

On the other hand, if the final computation does not use b as lookahead, there is an instance of (3) in which $c = \epsilon$, and the complete derivations of w are

$$\begin{array}{ll} \mathrm{V}_{qaAr\epsilon} \Rightarrow^* w & \text{and} \\ \mathrm{V}_{q\epsilon Ar\epsilon} \Rightarrow^* w. & \end{array}$$

In all cases, we have $\mathrm{V}_{qaArb} \Rightarrow^* w$ and $\mathrm{V}_{q\epsilon Arb} \Rightarrow^* w$ in G if the computation $(q, wb, A) \vdash^* (r, b, \epsilon)$ uses b as a lookahead symbol, and $\mathrm{V}_{qaAr\epsilon} \Rightarrow^* w$ and $\mathrm{V}_{q\epsilon Ar\epsilon} \Rightarrow^* w$ if it doesn't.

2. Alternatively, suppose P's first transition moves the tape head 1 position right, i.e., it uses a transition of the form $qaA \to s_1 \mathrm{R} B_n \ldots B_2 B_1$. Since this transition moves the tape head past a, a must be the first symbol of w and we can write $w = a w_1 w_2 \ldots w_n$ where, as in case 1, w_i is the substring of w that P reads past while clearing B_i from the stack. From this transition, construction rule 1 created rules in G of the forms

$$\begin{array}{l} \mathrm{V}_{qaArc} \to a \mathrm{V}_{s_1 \epsilon B_1 s_2 b_2} \mathrm{V}_{s_2 b_2 B_2 s_3 b_3} \ldots \mathrm{V}_{s_n b_n B_n r c} \\ \mathrm{V}_{q\epsilon Arc} \to a \mathrm{V}_{s_1 \epsilon B_1 s_2 b_2} \mathrm{V}_{s_2 b_2 B_2 s_3 b_3} \ldots \mathrm{V}_{s_n b_n B_n r c} \end{array} \tag{7}$$

for all states $s_1 \dots s_n$, and all lookahead values c and $b_2 \dots b_n$ in $\Sigma \cup \{\Box, \epsilon\}$. These rules provide the beginning of two derivations in G, namely

$$\mathrm{V}_{qaArc} \Rightarrow a\mathrm{V}_{s_1\epsilon B_1 s_2 b_2}\mathrm{V}_{s_2 b_2 B_2 s_3 b_3} \dots \mathrm{V}_{s_n b_n B_n rc}$$
$$\mathrm{V}_{q\epsilon Arc} \Rightarrow a\mathrm{V}_{s_1\epsilon B_1 s_2 b_2}\mathrm{V}_{s_2 b_2 B_2 s_3 b_3} \dots \mathrm{V}_{s_n b_n B_n rc}$$

Reasoning as in case 1, P clears each B_i from the stack through a computation of the form

$$(s_i, w_i w_{i+1} \dots w_n, B_i B_{i+1} \dots B_n) \vdash^* (s_{i+1}, w_{i+1} \dots w_n, B_{i+1} \dots B_n)$$

and there are rules of form (7) that use the same states and lookahead symbols in their variables, so that corresponding to each of these computations, $\mathrm{V}_{s_i b_i B_i s_{i+1} b_{i+1}} \Rightarrow^* w_i$. The collective result is thus, as it was in case 1, that $\mathrm{V}_{qaArc} \Rightarrow^* w$ and $\mathrm{V}_{q\epsilon Arc} \Rightarrow^* w$. Finally, and again as argued in case 1, we can have $c = b$ if P used b as lookhead at the end of its computation, and $c = \epsilon$ if not.

This completes the inductive argument that if $(q, wb, A) \vdash^* (r, b, \epsilon)$, then $\mathrm{V}_{qaArb} \Rightarrow^* w$ and $\mathrm{V}_{q\epsilon Arb} \Rightarrow^* w$ in G if the computation in P ends with lookahead symbol b and $\mathrm{V}_{qaAr\epsilon} \Rightarrow^* w$ and $\mathrm{V}_{q\epsilon Ar\epsilon} \Rightarrow^* w$ in G if the computation does not end with a lookahead symbol. □

Finally, we can prove the main theorem by tying together lemmas 4 and 5:

Corollary 3. For all strings $w \in \Sigma^*$, $w \in L(G)$ if and only if $w \in L(P)$.

Proof. For the first direction, suppose $w \in L(G)$. Then from construction step 5, w must be derived from $\mathrm{V}_{q_0\epsilon\$r\epsilon}$ or $\mathrm{V}_{q_0\epsilon\$r\Box}$, for some state r. For the first of these derivations, lemma 4 says that $(q_0, w, \$) \vdash^* (r, \epsilon, \epsilon)$, i.e., P accepts w. Similarly, for the second derivation, lemma 4 says that $(q_0, w\Box, \$) \vdash^* (r, \Box, \epsilon)$, which is also an accepting computation.

For the other direction, suppose $w \in L(P)$. Then there is a computation in P whereby $(q_0, w\Box, \$) \vdash^* (r, \Box, \epsilon)$ for some state r. Then by lemma 5, we have that $\mathrm{V}_{q_0\epsilon\$r\Box} \Rightarrow^* w$ in G if P uses the blank as lookahead, and $\mathrm{V}_{q_0\epsilon\$r\epsilon} \Rightarrow^* w$ if P does not use the blank. By construction step 5, S can rewrite to both $\mathrm{V}_{q_0\epsilon\$r\Box}$ and $\mathrm{V}_{q_0\epsilon\$r\epsilon}$, no matter what state r is, and so G derives w. □

Although the proof was long, we have now seen that for any pushdown automaton, it is possible to construct a context free grammar whose language is the same as the automaton's. Thus every pushdown automaton accepts a context free language.

4 Exercises

1. One of the fine points about Maheshwari and Smid's pushdown automata is that they only accept if the tape head is on the *first* blank when the stack becomes empty. In other words, an automaton that finishes reading some input but then moves right beyond the first blank does not accept that input. In terms of our construction, this means that G should never generate strings with blanks at the end. What in the construction ensures this property?

2. Instead of allowing pushdown automata to use lookahead, authors such as [1, 3] allow them to use "epsilon transitions," i.e., transitions that do not read the tape at all (nor do such transitions advance the tape). Show that for every pushdown automaton with lookahead (but not epsilon transitions) there is a pushdown automaton with epsilon transitions (but not lookahead) that accepts the same language. Hints: consult [1] or [3] for more detailed definitions of pushdown automata with epsilon transitions. Consider simulating lookahead by "remembering" an input symbol in the automaton's state.

References

[1] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Automata Theory, Languages, and Computation.* Pearson, 3rd edition, 2007.

[2] Anil Maheshwari and Michiel Smid. Introduction to theory of computation. (http://cglab.ca/∼michiel/TheoryOfComputation/), School of Computer Science, Carleton University, March 2017.

[3] Michael Sipser. *Introduction to the Theory of Computation.* Cengage Learning, 3rd edition, 2013.