Lecture 10

Closure Properties for Context-Free Languages

In this lecture, we will show that context-free languages have various nice closure properties. However, they don't quite satisfy all the closure properties that regular languages satisfy: in particular, CFLs are not closed under intersection and complement (we will prove this in the next lecture, where we introduce tools for showing that a language is not context-free).

10.1 The regular operations

We first show that the context-free languages are closed under the regular operations (union, concatenation, and star).

Union

To show that the context-free languages are closed under union, let A and B be context-free languages over an alphabet Σ, and let $G_A = (V_A, \Sigma, R_A, S_A)$ and $G_B = (V_B, \Sigma, R_B, S_B)$ be context-free grammars that generate A and B respectively. By renaming the variables if necessary, we assume that V_A is disjoint from V_B, and that neither variable set contains the variable S. We now construct a new CFG by writing

$$G = (V_A \cup V_B \cup \{S\}, \Sigma, R_A \cup R_B \cup \{S \to S_A, S \to S_B\}, S).$$

In other words, the new CFG G has all the variables and rules of G_A and G_B, plus a new start variable S together with the rule $S \to S_A | S_B$. We claim that $L(G) = A \cup B$, and hence $A \cup B$ is context-free.

To see this, first let $x \in L(G)$. This means that there is some derivation $z_1, z_2, \ldots, z_n$ for x in G, where $z_1 = S$, $z_n = x$, and $z_i \Rightarrow_G z_{i+1}$ for all $i = 1, 2, \ldots, n-1$. Since the only transition rules from S are $S \Rightarrow S_A$ and $S \Rightarrow S_B$, we must have $z_2 = S_A$ or $z_2 = S_B$. Since all the variables in V_A and V_B are disjoint, the derivation $z_2, z_3, \ldots, z_n$ is a derivation of x in either G_A or G_B. Hence we have either $x \in A$ or $x \in B$, so $x \in A \cup B$.

For the converse, let $x \in A \cup B$. Then either $x \in A$ or $x \in B$. This means we have a derivation $z_1, z_2, \ldots, z_n$ of x in either G_A or G_B, so either $z_1 = S_A$ or $z_1 = S_B$. Adding $z_0 = S$ gives a derivation $z_0, z_1, \ldots, z_n$ of x in G, and hence $x \in L(G)$. Thus we conclude that $A \cup B = L(G)$, and $A \cup B$ is therefore context-free.

Concatenation

As above, let A and B be context-free languages over an alphabet Σ, and let $G_A = (V_A, \Sigma, R_A, S_A)$ and $G_B(V_B, \Sigma, R_B, S_B)$ be CFGs for A and B respectively. As before, we will assume that V_A and V_B are disjoint and do not contain the variable S. This time, we want to construct a CFG for the language AB. This CFG will be

$$G = (V_A \cup V_B \cup \{S\}, \Sigma, R_A \cup R_B \cup \{S \to S_A S_B\}, S).$$

In other words, the new CFG G has all the variables and rules of G_A and G_B, plus a new start variable S with the rule $S \to S_A S_B$. We claim that $L(G) = AB$, and hence AB is context-free.

Let $w \in L(G)$. Then x has a derivation $z_1, z_2, \ldots, z_n$ with $z_n = w$ and $z_1 = S$, so $z_2 = S_A S_B$. In this derivation, the variable S_A eventually turns into some string $x \in \Sigma^*$ and the variable S_B eventually turns into some string $y \in \Sigma^*$, with $w = xy$. Since the variables of G_A and of G_B are disjoint, it must mean we have a derivation of x in G_A and a derivation of y in G_B, so $x \in A$ and $y \in B$, and hence $w = xy \in AB$.

For the converse, let $w \in AB$. Then $w = xy$ for some $x \in A$ and $y \in B$. This means there is a derivation $z_1, z_2, \ldots, z_n$ of x in G_A and another derivation $z'_1, z'_2, \ldots, z'_m$ of y in G_B. We can use the rule $S \Rightarrow S_A S_B$, together with these two derivations which convert S_A to x and S_B to y, to get a derivation of xy in G. Hence $w = xy \in L(G)$. We conclude that $L(G) = AB$, so AB is context-free, and hence CFLs are closed under concatenation.

Star

We now show that the context-free languages are closed under the Kleene star operation. Let A be a context-free language with CFG $G_A = (V_A, \Sigma, R_A, S_A)$, and assume that V_A does not contain the variable S. We construct a CFG for A^* as follows.

$$G = (V_A \cup \{S\}, \Sigma, R_A \cup \{S \to SS_A, S \to \epsilon\}, S).$$

In other words, G has all the variables and rules of G_A, plus a new start variable S with the rules $S \to SS_A|\epsilon$. We now show that $L(G) = A^*$, which implies that A^* is context-free.

Let $x \in L(G)$. Then there is a derivation $z_1, z_2, \ldots, z_n$ for x in G, with $z_1 = S$ and $z_n = x$. We can assume this derivation is left-most, so that the left most variable is replaced each time. Since we start with $z_1 = S$ and the only possible rules are $S \to SS_A$ and $S \to \epsilon$, the left-most variable stays S until we use the rule $S \to \epsilon$; in other words, the strings $z_1, z_2, z_3, \ldots$ start out as S, SS_A, SS_AS_A, and keep going until $S \to \epsilon$ is used, giving $z_{k+2} = S_A^k$ for some $k \leq n-2$. Then each of the k copies of the variable S_A in z_{k+2} derives a string in Σ^*; let these strings be $x_1, x_2, \ldots, x_k$. We have $x = x_1 x_2 \ldots x_k$, and since each of these derivations is a derivation in G_A, we have $x_i \in A$ for all $i = 1, 2, \ldots, k$. This means that $x \in A^k$ for some finite k, and hence $x \in A^*$.

Conversely, let $x \in A^*$. Since $A^* = \bigcup_{k \in \mathbb{N}} A^k$, it follows that $x \in A^k$ for some finite $k \in \mathbb{N}$. Let $x_1, x_2, \ldots, x_k \in A$ be such that $x = x_1 x_2 \ldots x_k$. Then we have a derivation of x_i in G_A for each $i \in \{1, 2, \ldots, k\}$. We can then use the rules

$$S \Rightarrow_G SS_A \Rightarrow_G \cdots \Rightarrow_G SS_A^k \Rightarrow S_A^k$$

followed by the derivations of $x_1, x_2, \ldots, x_k$ in G_A to get a derivation of x in G. Hence $x \in L(G)$, so we conclude that $L(G) = A^*$ and thus the CFLs are closed under star.

10.2 Every regular language is context-free

Note that the languages $\varnothing$, $\{\epsilon\}$, and $\{c\}$ for each $c \in \Sigma$ are all context-free, because we can easily write CFGs for them (for the language $\varnothing$, our CFG will not have any rules; for $\{\epsilon\}$, we can use the rule $S \to \epsilon$; and for the languages $\{c\}$, we can use the rule $S \to c$). Since all regular languages can be constructed from these starting languages using the regular operations, and since CFLs are closed under the regular operations, we conclude that every regular language is context-free.

We will now give an alternative proof of the fact that every regular language is context-free. This proof will directly convert a DFA for a regular language A into a CFG for A. Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA recognizing A. For each $q \in Q$, we will have one variable X_q. The start variable will be X_{q_0}. For each $q \in F$, we will include the rule $X_q \to \epsilon$ in our CFG. Finally, for $(q, c) \in Q \times \Sigma$, we will include the rule $S_q \to cS_{\delta(q,c)}$.

We claim that this CFG G that we defined generates the language A. As usual, we will handle the two directions separately: first we will show that $L(G) \subseteq A$, and then we will show that $A \subseteq L(G)$.

Let $x \in L(G)$. Then there is a derivation of x in G, say $z_1, z_2, \ldots, z_n$ with $z_1 = X_{q_0}$ and $z_n = x$. Note that each rule of G has at most one variable on the right hand side, and hence the number of variables in z_i is exactly 1 (except for the last string $z_n = x$, which has no variables). Moreover, this variables occurs as the rightmost symbol of z_i. For each $i \in \{1, 2, \ldots, n-1\}$, let p_i be the state of M that corresponds to the variable in z_i. Then we get a sequence of states $p_1, p_2, \ldots, p_{n-1}$ with $p_1 = q_0$. Moreover, since z_n has no variables, the transition from z_{n-1} to z_n must use the rule $X_{p_{n-1}} \to \epsilon$, which means that $p_{n-1} \in F$. Finally, note that the transition from z_i to z_{i+1} adds a character c_i to the left of the variable X_{p_i} and moves to the variable $X_{p_{i+1}}$, which means that we must have $\delta(p_i, c_i) = p_{i+1}$. These characters c_i eventually form the whole string x, so $x = c_1 c_2 \ldots c_{n-2}$. This implies that the sequence $p_1, p_2, \ldots, p_{n-1}$ is exactly the sequence of states in Q that M visits when run on x. Since $p_{n-1} \in F$, M accepts x, so $x \in A$.

Conversely, let $x \in A$. Then M accepts when run on x. Let $p_1, p_2, \ldots, p_n$ be the sequence of states M visits when run on x, so that $p_1 = q_0$ and $p_{i+1} = \delta(p_i, c_i)$ for all $i = 1, 2, \ldots, n-1$, where $x = c_1 c_2 \ldots c_{n-1}$. Since M accepts x, we must have $p_n \in F$. Then the derivation

$$X_{p_1} \Rightarrow c_1 X_{p_2} \Rightarrow c_1 c_2 X_{p_3} \Rightarrow \cdots \Rightarrow c_1 c_2 \ldots c_{n-1} X_{p_n} \Rightarrow c_1 c_2 \ldots c_{n-1}$$

is a valid derivation of x in the context-free grammar G we described. This means that $x \in L(G)$. We conclude that $L(G) = A$, so A is context-free, as desired.

10.3 Intersection with a regular language

The intersection of two context-free languages need not be context-free, as we will show in the next lecture. However, the intersection of a context-free language with a *regular* language will always be context-free. Let's prove this.

Let A be a context-free language, and let B be a regular language. Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA for B, and let $G_A = (V_A, \Sigma, R_A, S_A)$ be a CFG for A. We assume that G_A is in Chomsky normal form. We wish to construct a CFG for $A \cap B$.

Our new CFG G will have one variable for each triple in $V_A \times Q \times Q$; that is, for each variable of G_A, it will have $|Q|^2$ variables, for a total of $|V_A| \cdot |Q|^2$. It will also have an additional start variable S, making the final total $|V_A| \cdot |Q|^2 + 1$. The notation we will use is as follows: for each $X \in V_A$, we will use $X_{p,q}$ to denote the variable of G which corresponds to $(X, p, q) \in V_A \times Q \times Q$. We will use S for the additional start variable in G.

Our goal will be to make the variable $X_{p,q}$ generate all strings in Σ^* that can be generated by X in G_A and which *also* cause M to go from p to q; that is, for any $w \in \Sigma^*$, we want $X_{p,q} \Rightarrow_G^* w$ to hold if and only if both $X \Rightarrow_{G_A}^* w$ and $\delta^*(p, w) = q$. Once we achieve this, we will also add the rules $S \to (S_A)_{q_0,p}$ for each $p \in F$; this will mean that S can generate any string which can be generated by S_A in G_A and which also causes M to transition from q_0 to an accept state p of M.

Since G_A is in Chomsky normal form, its rules take only three different forms: the possible rule $S_A \to \epsilon$, rules $X \to YZ$ with Y and Z not equalling the start variable, or rules $X \to c$ with $c \in \Sigma$. If the rule $S_A \to \epsilon$ is present in G_A and if q_0 is an accept state, we add the rule $S \to \epsilon$ to G; otherwise, we do not add this rule (as ϵ will not be in $A \cap B$). For each rule $X \to c$ in G_A and for each $p \in Q$, we add the rule $X_{p,\delta(p,c)} \to c$ to the grammar G. Finally, for each rule $X \to YZ$ in G_A and for each $p_1, p_2, p_3 \in Q$, we add the rule $X_{p_1,p_2} \to Y_{p_1,p_3} Z_{p_3,p_2}$ to the grammar G.

This completes the definition of G. Note that G is almost in Chomsky normal form, in that once we leave the start variable S, we can never return to it (it does not occur on the right hand side of any rule), and no other variables can become ϵ. In particular, this means that the variables $X_{p,q}$ of G never generate ϵ.

We will now show that a variable $X_{p,q}$ in G generates a string $w \in \Sigma^* \setminus \{\epsilon\}$ if and only if $X \Rightarrow_{G_A}^* w$ and $\delta^*(p, w) = q$. In other words, we show that for all $w \in \Sigma^* \setminus \{\epsilon\}$ and for all $X \in V_A$, $p, q \in Q$, we have

$$X_{p,q} \Rightarrow_G^* w \quad \Leftrightarrow \quad (X \Rightarrow_{G_A}^* w \text{ and } \delta^*(p, w) = q).$$

We first show that the left hand side implies the right hand side. Indeed, suppose not, and let w be the smallest counterexample. Since $X_{p,q} \Rightarrow_G^* w$, we have a derivation $z_1, z_2, \ldots, z_n$ in G with $z_1 = X_{p,q}$ and $z_n = w$. If the first rule used from z_1 to z_2 has the form $X_{p,q} \Rightarrow c$ for some $c \in \Sigma$, then we have $w = c$, and we have the rule $X \to c$ in G_A as well as $q = \delta(p, c)$. This means that $X \Rightarrow_{G_A}^* w$ and that $\delta^*(p, w) = q$, as desired. The remaining case is when the first rule from z_1 to z_2 has the form $X_{p,q} \to Y_{p,p'} Z_{p',q}$, where $X \to YZ$ is a rule of G_A. In this case, $Y_{p,p'}$ becomes some nonempty string y and $Z_{p',q}$ becomes some nonempty strong z in the derivation of w, with $w = yz$. Note that y and z are shorter than w and are not the empty string, and w was the smallest counterexample. Note also that $Y_{p,p'} \Rightarrow_G^* y$ and $Z_{p',q} \Rightarrow_G^* z$. This implies that $Y \Rightarrow_{G_A}^* y$ and $\delta^*(p, y) = p'$, and also that $Z \Rightarrow_{G_A}^* z$ and $\delta^*(p', z) = q$. Since $X \to YZ$ is a rule of G_A, we conclude that $X \Rightarrow_{G_A} YZ \Rightarrow_{G_A}^* yz$, so $X \Rightarrow_{G_A}^* w$. Also, since $\delta^*(p, y) = p'$ and $\delta^*(p', z) = q$, we have $\delta^*(p, w) = \delta^*(p, yz) = q$. This gives a contradiction, as w was not a counterexample.

We now show the other direction, that the right hand side implies the left hand side. Once again, suppose not, and consider the shortest counterexample w. We have $X \Rightarrow_{G_A}^* w$ and $\delta^*(p, w) = q$, and w is not the empty string. Consider the derivation of w from X in G_A. Since G_A is in Chomsky normal form and since w is not empty, we cannot use a rule with ϵ on the right hand side; hence the derivation of w either starts with $X \to c$ for some $c \in \Sigma$ or else with $X \to YZ$ where neither Y nor Z are equal to S_A. In the former case, we have $w = c$, so $\delta(p, c) = q$, and hence the rule $X_{p,q} \to c$ exists in G, so $X_{p,q} \Rightarrow_G w$. In the latter case, we use $X \Rightarrow_{G_A} YZ$ in the derivation of w, so we must have $w = yz$ with $Y \Rightarrow_{G_A}^* y$ and $Z \Rightarrow_{G_A}^* z$, and neither y nor z are the empty string. Thus both y and z are shorter than w, so they are not counterexamples. Let $p' = \delta^*(p, y)$. Then since $\delta^*(p, w) = q$ and $w = yz$, we must have $\delta^*(p', z) = q$. Since $p' = \delta^*(p, y)$ and $Y \Rightarrow_{G_A}^* y$, and since y is not a counterexample, we have $Y_{p,p'} \Rightarrow_G^* y$. Similarly, since $q = \delta^*(p', z)$ and $Z \Rightarrow_{G_A}^* z$, and since z is not a counterexample, we have $Z_{p',q} \Rightarrow_G^* z$. Finally, since the rule $X \to YZ$ is in G_A, the rule $X_{p,q} \to Y_{p,p'} Z_{p',q}$ must be in G, so we have $X_{p,q} \Rightarrow_G Y_{p,p'} Z_{p',q} \Rightarrow_G^* yz = w$. This contradicts the assumption that w was a counterexample.

We've now shown that a nonempty string w is generated by $X_{p,q}$ in G if and only if it is generated by X in G_A and it causes M to go from p to q. Since we have the rules $S \to (S_A)_{q_0,p}$ in G for

each $p \in F$ (and these are the only rules with S on the left hand side except for the possible rule $S \to \epsilon$), a nonempty string w is generated by G (starting from S) if and only if it is generated by some $(S_A)_{q_0,p}$ for an accept state p, which happens if and only if $S_A \Rightarrow_{G_A}^* w$ and $\delta^*(q_0, w) \in F$, or in other words, if and only if $w \in L(G_A)$ and $w \in L(M)$. This means that $w \in L(G)$ if and only if w is in both A and B, at least assuming w is not the empty string.

Finally, note that the empty string w is generated by G if and only if the rule $S \to \epsilon$ was included in G. The way we defined G, this happens if and only if ϵ is in both A and B. Hence $L(G) = A \cap B$. Since A and B were arbitrary, we conclude that the intersection of a regular language and a context-free language is context-free.

10.4 Reverse

We will now show that the context-free languages are closed under the operations reverse, prefix, suffix, and substring.

We will start with reverse. Let A be context-free, and let G_A be a CFG for A. We assume that G_A is in Chomsky normal form. Let G be a CFG with the same variables as G_A, and with the same rules of the form $X \to c$ or $S \to \epsilon$ for $c \in \Sigma$, but for each rule $X \to YZ$ in G_A we include the rule $X \to ZY$ in G instead. We claim that $L(G) = A^R$.

To show this, we will actually show something slightly stronger: we will show that for each variable X and each string $w \in \Sigma^*$, we have $X \Rightarrow_G^* w$ if and only if $X \Rightarrow_{G_A}^* w^R$. When X is the start variable S, this implies that $w \in L(G)$ if and only if $w^R \in L(G_A) = A$, which means $L(G) = A^R$.

We will prove this claim by induction on the length of w. When $w = \epsilon$, we the statement $X \Rightarrow_G^* w$ holds if and only if $X = S$ and $S \to \epsilon$ is present in G; moreover, $X \Rightarrow_{G_A}^* w$ holds if and only if $X = S$ and $S \to \epsilon$ is present in G_A. Since $S \to \epsilon$ is a rule of G exactly when it is a rule of G_A, the desired claim follows when $w = \epsilon$.

Next, suppose $|w| = 1$. Then $X \Rightarrow_G^* w$ if and only if $X \to w$ is a rule of G, and similarly $X \Rightarrow_{G_A}^* w$ holds if and only if $X \to w$ is a rule of G_A. Since $X \to w$ is a rule of G exactly when it is a rule of G_A when $w \in \Sigma$, and since $w^R = w$ when $|w| = 1$, the desired claim holds.

Finally, suppose $|w| > 1$, and suppose by induction that the claim holds for all shorter strings. If $X \Rightarrow_G^* w$, then the derivation of w from X starts with a rule $X \to YZ$ of G; since Y and Z cannot be S, they cannot become the empty string, and hence $w = yz$ for some nonempty strings y and z with $Y \Rightarrow_G^* y$ and $Z \Rightarrow_G^* z$. By our induction hypothesis, $Y \Rightarrow_{G_A}^* y^R$ and $Z \Rightarrow_{G_A}^* z^R$, so $ZY \Rightarrow_{G_A}^* z^R y^R = w^R$. Since $X \to YZ$ is a rule of G, $X \to ZY$ must be a rule of G_A, and hence $X \Rightarrow_{G_A}^* w^R$. Conversely, suppose that $X \Rightarrow_{G_A}^* w^R$. Then the derivation must start with $X \Rightarrow_{G_A} YZ$ for some rule $X \to YZ$ in G_A. Then Y and Z cannot be S and cannot generate the empty string; hence $w^R = yz$ for some shorter strings y and z, with $Y \Rightarrow_{G_A}^* y$ and $Z \Rightarrow_{G_A}^* z$. By the induction hypothesis, we get $Y \Rightarrow_G^* y^R$ and $Z \Rightarrow_G^* z^R$, so $ZY \Rightarrow_G^* z^R y^R = w$. By the construction of G, the rule $X \to ZY$ must be in G (since the rule $X \to YZ$ was in G_A). Thus we have $X \Rightarrow_G^* w$, as desired. This completes the proof that context-free languages are closed under reverse.

10.5 Prefix, suffix, and substring

We will now show that the context-free languages are closed under prefix, suffix, and substring. Recall that for any language A, we have $\text{Suffix}(A) = \text{Prefix}(A^R)^R$. Since we know that context-free languages are closed under reverse, if we show that they are closed under prefix it will imply they are

also closed under suffix. Similarly, for any language A, we have Substring(A) = Prefix(Suffix(A)). This means that if we show the context-free languages are closed under prefix, it will also imply they are closed under substring.

It remains to show that the context-free languages are closed under prefix. To this end, let A be a context-free language, and let G be a CFG for A. Assume that G is in Chomsky normal form. Note that if $A = \varnothing$, then Prefix$(A) = \varnothing$, so Prefix(A) is context-free. From this point on, suppose that $A \neq \varnothing$. We will also assume that G does not have a *useless variable*, that is, a variable which cannot generate any string in Σ^*. It's not hard to see that if such a variable exists in G, we could delete it and delete all rules that use it (either on the left hand side or the right hand side), and this will not affect the language of the grammar G. Hence we assume that all variables of G can generate at least one string.

We will now construct a CFG G' for Prefix(A). The CFG G' will have two variables for each variable X of G, denoted X and X_0. The idea will be for the variables X in G' to generate the strings that are generated by X in G, while the variables X_0 in G' generate the prefixes of the strings that are generated by X in G.

For each rule $X \to c$ in G (where $c \in \Sigma$), we add the rules $X \to c$ and $X_0 \to c|\epsilon$ to G'. For each rule $X \to YZ$ in G, we add the rules $X \to YZ$ and $X_0 \to YZ_0|Y_0$ to G'. add the rule $S \to \epsilon$ to G' if it is present in G, and add the rule $S_0 \to \epsilon$ either way. We set S_0 to be the start variable of G'.

Note that the rules of G' with variables X of G (rather than X_0) on the left hand side all have variables of G on the right hand side, and these rules are all exactly the rules of G. In other words, G' has all the variables of G with the exact same rules governing them, it simply also has additional variables X_0 with additional rules for them (and a different start variable). In particular, it follows that $X \Rightarrow_G^* w$ if and only if $X \Rightarrow_{G'}^* w$ for all $w \in \Sigma^*$.

We now claim that for any string $w \in \Sigma^*$, we have $X_0 \Rightarrow_{G'}^* w$ if and only if $X \Rightarrow_G^* ws$ for some $s \in \Sigma^*$. This means that the variable X_0 generates precisely the prefixes of the strings generated by X, and in particular, S_0 generates the prefixes of the strings generated by S, i.e. the language Prefix(A).

To see this, we first show that for every variable X, every string w generated by X_0 is a prefix of some string generated by X. We will do this by induction on the length of the derivation of w from X_0. In the base case, if $X_0 \Rightarrow_{G'} w$, then $w = \epsilon$ or $w = c$ for some $c \in \Sigma$, and the rule $X \to c$ must be present in G; in that case, c can be derived from X in G, and w is a prefix of c. Now let w have a longer derivation from X_0, and suppose the claim holds for all shorter derivations. The beginning of the derivation of w must look like $X_0 \Rightarrow_{G'} YZ_0$ or $X_0 \Rightarrow_{G'} Y_0$ where the rule $X \to YZ$ is in G. In the former case, the derivation continues with YZ_0 turning into w, so $w = yz$ with $Y \Rightarrow_G^* y$ and $Z_0 \Rightarrow_{G'}^* z$. By the induction hypothesis, z is a prefix of a string z' generated by Z (since the derivation of z from Z_0 is shorter than that of w from X_0). Hence we have $X \Rightarrow_G YZ \Rightarrow_G^* yz'$, and $w = yz$ is a prefix of yz' since z is a prefix of z. In the latter case, the derivation continues with Y_0 turning into w, so $Y_0 \Rightarrow_{G'}^* w$, and by the induction hypothesis $Y \Rightarrow_G^* ws$ for some string s; hence $X \Rightarrow_G YZ \Rightarrow_G^* wst$, where t is some string generated by Z (this exists as Z is not a useless variable). Thus w is a prefix of a string wst generated by X.

We now show the other direction, that each prefix of a string generated by a variable X is generated by X_0. We do this by induction on the length of the string. If the string w is ϵ or $c \in \Sigma$, then since G is in Chomsky normal form, the derivation of the string w from X must have the form $X \Rightarrow_G w$, so the rule $X \to w$ is in G. This means the rule $X_0 \to w|\epsilon$ is in G', so X_0 generates both w and ϵ, which are all the prefixes of w. Now consider any other string w with $|w| \geq 2$ generated by X, and suppose that the claim holds for all shorter strings. Since G is in Chomsky normal form, the derivation of w from X must start with $X \Rightarrow_G YZ$, where Y and Z are not the start variable (and hence cannot generate ϵ). Hence we have $w = yz$ where y and z are shorter than w, and

$Y \Rightarrow_G^* y$, $Z \Rightarrow_G^* z$. By the induction hypothesis, Y_0 generates all prefixes of y and Z_0 generates all prefixes of z. Note that each prefix of $w = yz$ is either a prefix of y, or else has the form yx where x is a prefix of z. Since the rule $X \to YZ$ is in G, the rule $X_0 \to YZ_0|Y_0$ is in G'. This means we have $X_0 \Rightarrow_{G'} Y_0$ and $X_0 \Rightarrow_{G'} YZ_0 \Rightarrow_{G'}^* yZ_0$, so X_0 can generate all the prefixes of y and also all strings of the form yx where x is a prefix of z. This means that X_0 can generate all prefixes of w, as desired.