Automata Theory and Formal Grammars: Lecture 6

Context Free Languages

Portions ©2000 Rance Cleaveland ©2004 James Riely

Context Free Languages

Last Time

- Decision procedures for FAs
- Minimum-state DFAs

Today

- The Myhill-Nerode Theorem
- The Pumping Lemma
- Context-free grammars and languages
- Closure properties of CFLs
- Relating regular languages and CFLs

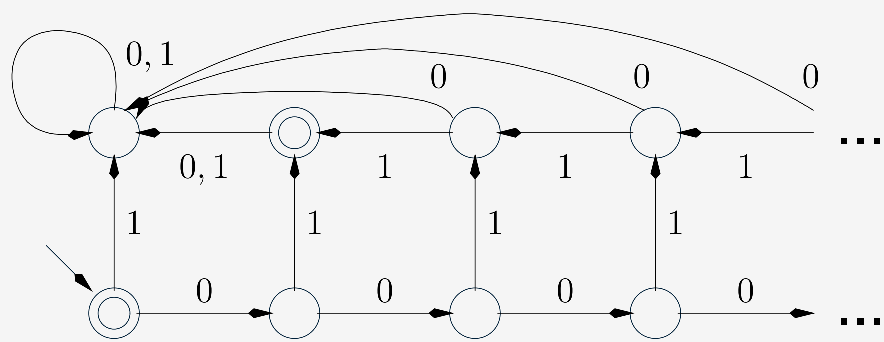
Non-Regular Languages

Languages That Are Not Regular

So far we have only seen regular languages. Do nonregular ones exist?

$\boxed{\text{Yes!}}$ Consider $L = \{\, 0^n 1^n \mid n \geq 0 \,\}$.

- What would a "FA" look like for this language?



- What can you say about the strings 0^i and 0^j if $i \neq j$?
  If $i \neq j$ then $0^i \not\stackrel{L}{\bowtie} 0^j$!

In this case $\stackrel{L}{\bowtie}$ has an infinite number of equivalence classes!

The Myhill-Nerode Theorem

Theorem (Myhill-Nerode) Let $L \subseteq \Sigma^*$ be a language. Then L is regular if and only if $\overset{L}{\bowtie}$ has a finite number of equivalence classes.

So how do you prove that a language L is not regular using Myhill-Nerode?

- Must show that $\overset{L}{\bowtie}$ has an infinite number of equivalence classes.
- Suffices to give an *infinite* set $S \subseteq \Sigma^*$ whose elements are *pairwise distinguishable* with respect to L: for every $x, y \in S$ with $x \neq y$, $x \overset{L}{\not\bowtie} y$.

Why does this condition suffice?

- If S is pairwise distinguishable, then every element of S must belong to a different equivalence class of $\overset{L}{\bowtie}$.
- Since S is infinite, there must be an infinite number of equivalence classes!

Example: Proving Nonregularity of $\{ 0^n 1^n \mid n \geq 0 \}$

$\boxed{\text{Theorem}}$ $L = \{ 0^n 1^n \mid n \geq 0 \}$ is not regular.

$\boxed{\text{Proof}}$ On the basis of the Myhill-Nerode Theorem, it suffices to give an infinite set $S \subseteq \{0,1\}^*$ that is pairwise distinguishable with respect to L. Consider

$$S = \{ 0^i \mid i \geq 1 \}.$$

Clearly S is infinite.

We now must show that S is pairwise distinguishable. So consider strings $x = 0^i$ and $y = 0^j$ where $i \neq j$; we must show that $x \not\bowtie_L y$, which requires that we find a z such that $xz \in L$ and $yz \notin L$ (or vice versa). Consider $z = 1^i$. Then $xz = 0^i 1^i \in L$, but $yz = 0^j 1^i \notin L$. Thus $x \not\bowtie_L y$, and S is pairwise distinguishable.

Another Example: Even-Length Palindromes

$\boxed{\text{Recall}}$ If $x \in \Sigma^*$ then x^r is the "reverse" of x.

$\boxed{\text{E.g.}}$ $abb^r = bba$.

A *palindrome* is a word that is the same backwards as well as forwards.

- $abba$
- 01110
- RADAR

Any *even-length* palindrome can be written as $x \cdot x^r$ for some string x.

$\boxed{\text{E.g.}}$ $abba = ab \cdot ba = ab \cdot (ab)^r$.

Even-length palindromes over $\{a, b\}$ form a nonregular language.

Proving Even-Length Palindromes To Be Nonregular

Theorem Let $E = \{\, x \cdot x^r \mid x \in \{a,b\}^* \,\}$. Then E is not regular.

Proof On the basis of the Myhill-Nerode Theorem it suffices to come up with an infinite set $S \subseteq \{a,b\}^*$ that is pairwise distinguishable with respect to E. Consider

$$S = \{\, a^i b \mid i \geq 0 \,\}.$$

Clearly S is infinite.

To show pairwise distinguishability, consider $x = a^i b$ and $y = a^j b$ where $i \neq j$; we must show $x \overset{E}{\not\bowtie} y$, i.e. we must find a z with $xz \in L$ and $yz \notin L$, or vice versa. Consider

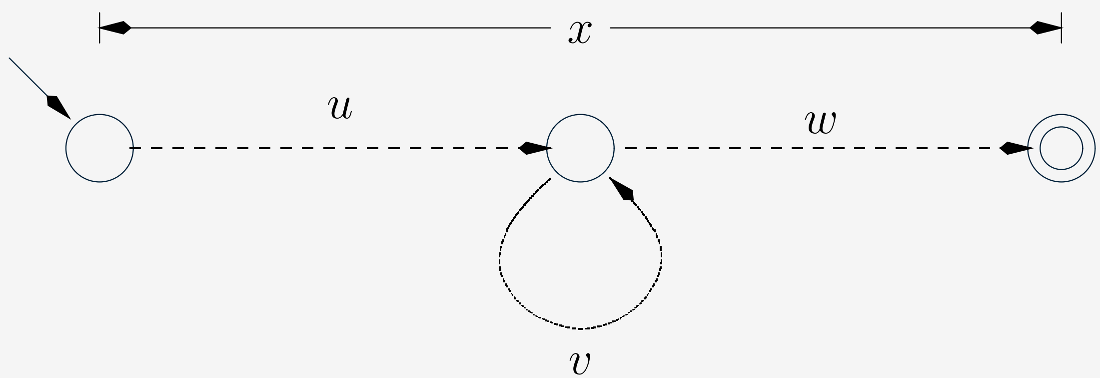
$$z = x^r = ba^i.$$

By definition $xz \in L$. However, $yz = a^j bba^i \notin L$ since $j \neq i$.

The Pumping Lemma: Another Way of Proving Nonregularity

By way of introduction, consider the following.

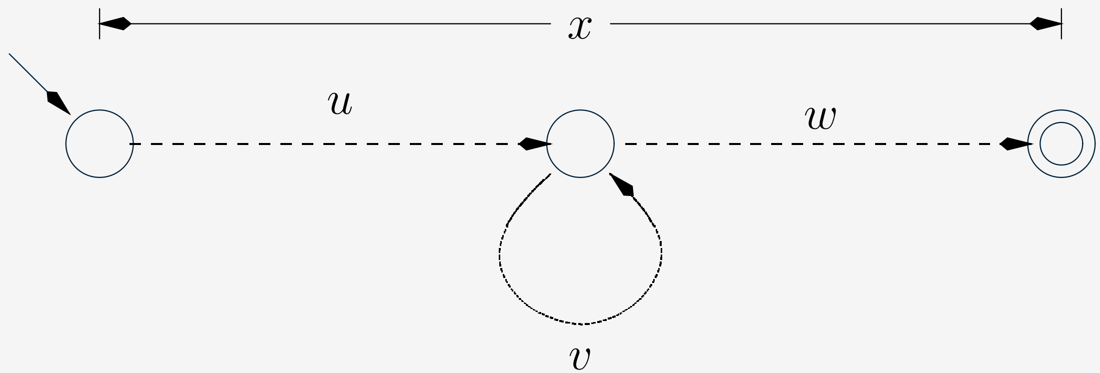
- If a language L is regular, there is a minimum-state DFA accepting L. Let n be the number of states in this DFA.
- What happens if $x \in L$ is at least n symbols long?



Some state must be visited twice, i.e. "cycled through"!

Portions ©2000 Rance Cleaveland ©2004 James Riely

The Pumping Lemma (cont.)



Notes

- $v > 0$ since the cycle has to contain something.
- The first repeated state satisfies: $|uv| \leq n$ (why?).
- $uv^m w \in L$ all $m \geq 0$!

Formalizing the Pumping Lemma

Lemma (Pumping Lemma) If $L \subseteq \Sigma^*$ is regular, then there exists $n > 0$ such that for any $x \in L$, if $|x| \geq n$ then there exist $u, v, w \in \Sigma^*$ such that

$$
\begin{aligned}
x &= uvw && (1)\\
|uv| &\leq n && (2)\\
|v| &> 0 && (3)\\
uv^m w &\in L \quad \text{for any } m \geq 0 && (4)
\end{aligned}
$$

This lemma can be used to prove nonregularity! Look at its logical structure.

$$
\begin{aligned}
L \text{ is regular } \implies\ & \exists\, n > 0.\\
& \forall\, x \in L. |x| \geq n \implies\\
& \qquad \exists\, u, v, w \in \Sigma^*.\ x = uvw \wedge |uv| \leq n \wedge |v| > 0 \wedge\\
& \qquad\qquad \forall\, m \geq 0. uv^m w \in L
\end{aligned}
$$

Using the Pumping Lemma to Prove Nonregularity

Recall form of Pumping Lemma:

$$
\begin{aligned}
L \text{ is regular} \implies\ & \exists\, n > 0. \\
& \forall\, x \in L. |x| \geq n \implies \\
& \qquad \exists\, u, v, w \in \Sigma^*.\ x = uvw \wedge |uv| \leq n \wedge |v| > 0 \wedge \\
& \qquad\qquad \forall\, m \geq 0. uv^m w \in L
\end{aligned}
$$

What is contrapositive? $\neg(\exists n > 0 \ldots) \implies L$ is not regular!
If we drive the negation inside the antecedent we get:

$$
\begin{aligned}
& \forall n > 0. \exists\, x \in L.\ |x| \geq n \wedge \\
& \qquad \forall\, u, v, w \in \Sigma^*. (x = uvw\ \wedge\ |uv| \leq n\ \wedge\ |v| > 0) \implies \\
& \qquad\qquad \exists\, m \geq 0. uv^m w \notin L
\end{aligned}
$$

So if we can prove this statement of a language L, then L is not regular!

Example: Proving $\{ ww \mid w \in \{a,b\}^* \}$ Is Not Regular

$\boxed{\text{Theorem}}$ $L = \{ ww \mid w \in \{a,b\}^* \}$ is not regular.

$\boxed{\text{Proof}}$ On the basis of the Pumping Lemma it suffices to prove the following.

$$\forall n > 0. \exists\, x \in L.\ |x| \geq n\ \wedge$$
$$\forall\, u, v, w \in \Sigma^*. (x = uvw\ \wedge\ |uv| \leq n\ \wedge\ |v| > 0) \implies$$
$$\exists\, m \geq 0. uv^m w \notin L$$

So fix $n > 0$ and consider $\boxed{x = a^n b^n a^n b^n}$. Clearly $x \in L$ and $|x| > n$.

Now fix $u, v, w \in \Sigma^*$ and assume that $x = uvw$, $|uv| \leq n$, and $|v| > 0$. We have the following picture.

$$\underbrace{a\cdots a}_{u}\,\underbrace{a\cdots a}_{v}\,\underbrace{a\cdots a\ \overbrace{b\cdots b}^{n}\ \overbrace{a\cdots a}^{n}\ \overbrace{b\cdots b}^{n}}_{w}$$

Portions © 2000 Rance Cleaveland © 2004 James Riely

Proof (cont.)

That is, $v = a^i$ some $i > 0$. Now let $m = 2$, and consider

$$uv^m w = uv^2 w = a^{n+i} b^n a^n b^n.$$

This word is not an element of L; consequently, L cannot be regular.

Context-Free Grammars

Context-Free Grammars and Languages

Regular languages have a nice theory:

- Regular expressions give a "syntax" for defining them.
- FAs provide the computational means for processing them.

However, some "simple" languages are not regular, e.g.
$L = \{ 0^n 1^n \mid n \geq 0 \}$.

- No FA exists for L.
- On the other hand, it's easy to give a recursive definition of L.
  - $\varepsilon \in L$
  - If $w \in L$ then $0w1 \in L$.

Observations

- Some "easy to process languages" like $L = \{ 0^n 1^n \mid n \geq 0 \}$ are nevertheless not recognizable using FAs alone.
- So there must be computing devices that are "better" than FAs when it comes to recognizing languages.
- There must also be "more general" classes of languages than regular languages that are still amenable to automatic analysis.

Context-free languages represent the next, broader class of languages we will study. They are defined using *context-free grammars*.

Context-Free Grammars

... provide a notation for defining languages "recursively".

Example A context-free grammar for $\{ 0^n 1^n \mid n \geq 0 \}$.

$$\begin{array}{rcl} S & \longrightarrow & \varepsilon \\ & \mid & 0S1 \end{array}$$

- S is a *nonterminal* (think "variable").
- The grammar has two *productions* saying how variable S may be rewritten.
- One generates words by applying productions beginning from the *start symbol* (always a nonterminal, here S):

$$S \Rightarrow 0S1 \Rightarrow 00S11 \Rightarrow 00\varepsilon 11 = 0011$$

Defining Context-Free Grammars

$\boxed{\text{Definition}}$ A *context-free grammar* (CFG) is a quadruple $\langle V, \Sigma, S, P \rangle$, where:

- V is a finite set of *variables* (aka *nonterminals*).
- Σ is an alphabet, with $V \cap \Sigma = \emptyset$. Elements of Σ are sometimes called *terminals*.
- $S \in V$ is a distinguished *start symbol*.
- P is a finite set of *productions* of the form $A \longrightarrow \alpha$, where $A \in V$ and $\alpha \in (V \cup \Sigma)^*$.

Notational Conventions for CFGs

- $A \longrightarrow \alpha_1 \mid \cdot \mid \alpha_n$ is shorthand for n productions of form $A \longrightarrow \alpha_i$.
- Start symbol is first one written down.

$\boxed{\text{E.g.}}$ In CFG

$$\begin{array}{rcl} S & \longrightarrow & \varepsilon \\ & \mid & 0S1 \end{array}$$

$V = \{S\}$, $\Sigma = \{0, 1\}$, S is start symbol, and $P = \{S \longrightarrow \varepsilon, S \longrightarrow 0S1\}$.

Other CFG Examples

Palindromes over $\Sigma = \{a, b\}$

$$
\begin{aligned}
S &\rightarrow \varepsilon \mid a \mid b \\
&\mid \quad aSa \mid bSb
\end{aligned}
$$

Sample word: $S \Rightarrow aSa \Rightarrow abSba \Rightarrow ababa$

Nonpalindromes over $\Sigma = \{a, b\}$

$$
\begin{aligned}
S &\rightarrow aSa \mid bSb \mid aAb \mid bAa \\
A &\rightarrow \varepsilon \mid aA \mid bA
\end{aligned}
$$

Sample word: $S \Rightarrow aSa \Rightarrow aaAba \Rightarrow aa\varepsilon ba = aaba$

Languages of CFGs

CFGs are be used to generate strings of terminals and nonterminals.

- Productions are used as "rewrite rules" to replace variables by strings.
- So what should the language of a CFG be?
  The sequences of terminals that can be generated from the start variable.

How do we make this precise?

- Given a grammar G we'll define a "rewrite relation" $\Rightarrow_G$: $\alpha \Rightarrow_G \beta$ should hold if α can be "rewritten" into β by applying one production.
- Then $w \in \Sigma^*$ is in the language of G if $S \Rightarrow_G \alpha \Rightarrow_G \cdots \Rightarrow_G w$.

Defining $\Rightarrow_G$

Example Let G be: $S \longrightarrow \varepsilon \mid 0S1$.

We want to define $\Rightarrow_G$ so that $0S1 \Rightarrow_G 00S11$.

Definition Let $G = \langle V, \Sigma, S, P \rangle$ be a CFG, and let $\alpha, \beta \in (V \cup \Sigma)^*$.

Then $\alpha \Rightarrow_G \beta$ if there exist $\alpha_1, \alpha_2, \gamma \in (V \cup \Sigma)^*$ and $A \in V$ such that:

- $\alpha = \alpha_1 \cdot A \cdot \alpha_2$
- $\beta = \alpha_1 \cdot \gamma \cdot \alpha_2$
- $A \longrightarrow \gamma$ is a production in P.

$$\underbrace{\overbrace{BaC\cdots b}^{\alpha_1}\ \boxed{A}\ \overbrace{BAc\cdots B}^{\alpha_2}}_{\alpha} \Rightarrow_G \underbrace{\overbrace{BaC\cdots b}^{\alpha_1}\ \boxed{\gamma}\ \overbrace{BAc\cdots B}^{\alpha_2}}_{\beta}$$

Generating Words in CFGs

$\Rightarrow_G$ defines the valid "one-step" derivations in a CFG. We can use this to define "multi-step" derivations via the relation $\Rightarrow_G^*$.

Example Let G be: $S \longrightarrow \varepsilon \mid 0S1$.
Then we want $S \Rightarrow_G^* 0011$ to hold.

Definition Let $G = \langle V, \Sigma, S, P \rangle$ be a CFG, and let $\alpha, \beta \in (V \cup \Sigma)^*$.
Then $\alpha \Rightarrow_G^* \beta$ if there exists $n \geq 0$ and $\alpha_0, ...\alpha_n \in (V \cup \Sigma)^*$ such that:

- $\alpha = \alpha_0$
- $\beta = \alpha_n$
- For all $i < n$ $\alpha_i \Rightarrow_G \alpha_{i+1}$.

In other words, $\alpha = \alpha_0 \Rightarrow_G \alpha_1 \Rightarrow_G \cdot \Rightarrow_G \alpha_n = \beta$.

Examples

Let G be the nonpalindrome CFG:

$$\begin{array}{lcl} S & \rightarrow & aSa \mid bSb \mid aAb \mid bAa \\ A & \rightarrow & \varepsilon \mid aA \mid bA \end{array}$$

1. Does $S \Rightarrow_G abaa$?
2. Does $aSAa \Rightarrow_G^* aabAa$?
3. Does $S \Rightarrow_G^* S$?
4. Does $S \Rightarrow_G^* A$?

The Language of a CFG

The language of a CFG G can now be defined using $\Rightarrow_G^*$.

$\boxed{\text{Definition}}$ Let $G = \langle V, \Sigma, S, P \rangle$ be a CFG. Then the *language* of G, $\mathcal{L}(G) \subseteq \Sigma^*$, is defined as follows.

$$\mathcal{L}(G) = \{\, w \in \Sigma^* \mid S \Rightarrow_G^* w \,\}$$

Context-free languages (CFLs) are those for which one can give CFGs.

$\boxed{\text{Definition}}$ A language $L \subseteq \Sigma^*$ is *context-free* if there is a CFG G with $L = \mathcal{L}(G)$.

Another CFG/CFL Example

A CFG for the valid arithmetic expressions over the natural numbers.

$$
\begin{array}{rcl}
S & \longrightarrow & N \\
& | & SOS \\
& | & (S) \\
N & \longrightarrow & D \mid PR \\
D & \longrightarrow & 0 \mid P \\
P & \longrightarrow & 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \\
R & \longrightarrow & D \mid DR \\
O & \longrightarrow & + \mid - \mid * \mid /
\end{array}
$$

Regular Languages and CFLs

Regular Languages and CFLs

$\boxed{\text{Theorem}}$ Every regular language is context-free.

How can we prove this? By giving any one of several different translations:

1. Regular expressions $\Rightarrow$ CFGs
2. FAs $\Rightarrow$ CFGs
3. NFAs $\Rightarrow$ CFGs

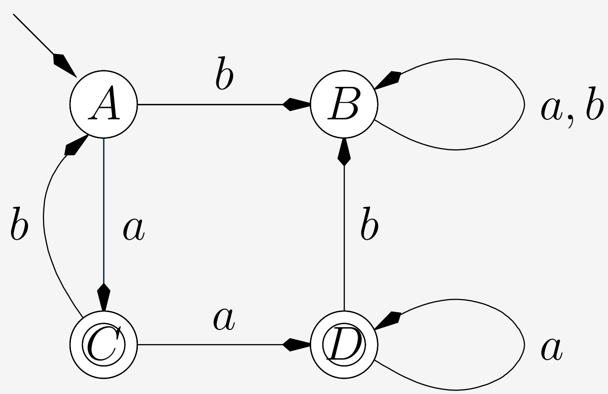
We will pursue (2).

Portions © 2000 Rance Cleaveland © 2004 James Riely

Translating FAs into CFGs

How do we do this? By turning:

- states into variables;
- transitions into productions; and
- acceptance into ε-productions.



$$
\begin{array}{rcl}
A & \longrightarrow & aC \mid bB \\
B & \longrightarrow & aB \mid bB \\
C & \longrightarrow & aD \mid bA \mid \varepsilon \\
D & \longrightarrow & aD \mid bB \mid \varepsilon
\end{array}
$$

Note $\delta^*(A, aab) = B$, and $A \Rightarrow_G^* aabB$.

Formalizing the Translation

Given a FA $M = \langle Q, \Sigma, q_0, \delta, A \rangle$, we want to define CFG $G_M = \langle V, \Sigma, S, P \rangle$ so that $\mathcal{L}(M) = \mathcal{L}(G_M)$. Assume without loss of generality that $Q \cap \Sigma = \emptyset$.

- $V = Q$
- $S = q_0$
- $P = \{\, q \longrightarrow a \cdot \delta(q, a) \mid q \in Q \,\} \cup \{\, q \longrightarrow \varepsilon \mid q \in A \,\}$

To prove that $\mathcal{L}(M) = \mathcal{L}(G_M)$ we can first argue that:

For every $x \in \Sigma^*, q, q' \in Q$,

$$\delta^*(q, x) = q' \text{ iff } q \Rightarrow^*_{G_M} x \cdot q'.$$

Then $x \in \mathcal{L}(M)$ iff $x \in \mathcal{L}(G_M)$! (Why?)

Closure Properties of CFLs

Closure Properties of CFLs

What we know:

- Every regular langauge is a CFL.
- Regular languages are closed with respect to: $\cdot, ^*, \cup, \cap$, etc.

Are CFLs automatically closed with respect to these operations also?

$\boxed{\text{No!}}$ Regular languages constitute a *proper subset* of CFLs, and the closure properties do not immediately “transfer.”

Nevertheless, we do have the following.

$\boxed{\text{Theorem}}$ The set of context-free languages is closed with respect to $\cup$, $\cdot$ and *.

Proofs rely on *grammar constructions*.

Proving CFLs Closed with Respect to $\cup$

We need to show how to combine two CFGs G_1 and G_2:

$$\begin{array}{ccccccc} S_1 & \longrightarrow & \cdots & \qquad & S_2 & \longrightarrow & \cdots \\ & \vdots & & & & \vdots & \\ & \boxed{G_1} & & & & \boxed{G_2} & \end{array}$$

into a single CFG G_U such that $\mathcal{L}(G_U) = \mathcal{L}(G_1) \cup \mathcal{L}(G_2)$. I.e. if S_U is start symbol of G_U then $S_U \Rightarrow^*_{G_U} x$ iff $S_1 \Rightarrow^*_{G_1} x$ or $S_2 \Rightarrow^*_{G_2} x$.

Idea

Why not introduce a new variable S_U as follows?

$$\begin{array}{rcl} S_U & \longrightarrow & S_1 \mid S_2 \\ S_1 & \longrightarrow & \cdots \\ & \vdots & \\ S_2 & \longrightarrow & \cdots \\ & \vdots & \end{array}$$

- If $S_1 \Rightarrow^*_{G_1} w$ then $S \Rightarrow_{G_U} S_1 \Rightarrow^*_{G_U} x$.
- If $S \Rightarrow^*_{G_U} x$ then $S \Rightarrow_{G_U} S_i \Rightarrow^*_{G_i} x$ for $i = 1$ or $i = 2$.

For the latter to hold we need to ensure that G_1, G_2 don't interfere with one another (i.e. share variables).

Formal Construction of G_U

Let $G_1 = \langle V_1, \Sigma, S_1, P_1 \rangle$ and $G_2 = \langle V_2, \Sigma, S_2, P_2 \rangle$; without loss of generality, assume that $V_1 \cap V_2 = \emptyset$. We build $G_U = \langle V_U, \Sigma, S_U, P_U \rangle$ as follows.

1. Choose a new variable $S_U \notin V_1 \cup V_2 \cup \Sigma$ to be the start symbol of G_U.
2. Take $V_U = V_1 \cup V_2 \cup \{S_U\}$
3. Set $P_U = P_1 \cup P_2 \cup \{S_U \longrightarrow S_1, S_U \longrightarrow S_2\}$

We can then argue that $\mathcal{L}(G_U) = \mathcal{L}(G_1) \cup \mathcal{L}(G_2)$ by first establishing:

Fact $S_1 \Rightarrow_{G_U} \alpha$ iff $S_1 \Rightarrow_{G_1} \alpha$ for any $\alpha \in (V_U \cup \Sigma)^*$.

Then $S_U \Rightarrow^*_{G_U} x$ iff $S_1 \Rightarrow^*_{G_1} x$ or $S_2 \Rightarrow^*_{G_2} x$, for any $x \in \Sigma^*$!

Proving CFLs Closed with Respect to $\cdot$

We need to show how to combine two CFGs G_1 and G_2:

$$\begin{array}{ccc} S_1 & \longrightarrow & \cdots \\ & \vdots & \\ & \boxed{G_1} & \end{array} \qquad\qquad \begin{array}{ccc} S_2 & \longrightarrow & \cdots \\ & \vdots & \\ & \boxed{G_2} & \end{array}$$

into a single CFG G_C such that $\mathcal{L}(G_C) = \mathcal{L}(G_1) \cdot \mathcal{L}(G_2)$. I.e. if S_C is start symbol of G_C then $S_C \Rightarrow^*_{G_C} x$ iff $S_1 \Rightarrow^*_{G_1} x_1$, $S_2 \Rightarrow^*_{G_2} x_2$, and $x = x_1 \cdot x_2$, for some x_1, x_2.

Idea

We can do something similar to what we did for $\cup$!

$$
\begin{array}{rcl}
S_C & \longrightarrow & S_1 \cdot S_2 \\
S_1 & \longrightarrow & \cdots \\
 & \vdots & \\
S_2 & \longrightarrow & \cdots \\
 & \vdots &
\end{array}
$$

- If $S_1 \Rightarrow^*_{G_1} x$ and $S_2 \Rightarrow^*_{G_2} y$ then $S \Rightarrow_{G_C} S_1 \cdot S_2 \Rightarrow^*_{G_C} xy$.
- If $S \Rightarrow^*_{G_C} x$ then $S \Rightarrow_{G_C} S_1 \cdot S_2 \Rightarrow^*_{G_C} x$, meaning $S_1 \Rightarrow_{G_1} x_1$ and $S_2 \Rightarrow_{G_2} x_2$ for some x_1, x_2 with $x_1 \cdot x_2 = x$.

For the latter to hold we need to ensure that G_1, G_2 don't share variables....

Formal Construction of G_C

Approach is similar to that for G_U: pick a new start symbol $S_C \notin V_1 \cup V_2 \cup \Sigma$, and construct $G_C = \langle V, \Sigma, S_C, P_C \rangle$ where:

- $V_C = V_1 \cup V_2 \cup \{S_C\}$.
- $P_C = P_1 \cup P_2 \cup \{S_C \longrightarrow S_1 \cdot S_2\}$

Proof of correctness follows similar lines to G_U case.

Proving CFLs Closed with Respect to *

To build G_K from G so that $\mathcal{L}(G_K) = (\mathcal{L}(G))^*$ we follow the same line of attack as for $\cup, \cdot$!

$$\begin{array}{ccc} S & \longrightarrow & \cdots \\ & \vdots & \\ & \boxed{G} & \end{array} \qquad \begin{array}{ccc} S_K & \longrightarrow & \varepsilon \mid S \cdot S_K \\ S & \longrightarrow & \cdots \\ & \vdots & \\ & \boxed{G_K} & \end{array}$$