CMPU 240 · Theory of Computation

The Limits of Context-Free Languages

31 March 2026

Where are we?

To show a language is context-free, we can

Design a context-free grammar that generates it ***or***

Design a pushdown automaton that recognizes it.

Closure properties of CFLs

To show a language is context-free, we can

Design a context-free grammar that generates it ***or***

Design a pushdown automaton that recognizes it ***or***

Use closure properties for context-free languages.

Recall that if L_1 and L_2 are *regular languages*, then

$\overline{L_1}$ is regular

$L_1 \cup L_2$ is regular

$L_1 \cap L_2$ is regular

$L_1 \circ L_2$ is regular

L_1^* is regular

L_1^R is regular

How many of these closure properties still hold for *context-free languages*?

THEOREM The class of context-free languages is closed under the regular operations.

THEOREM The class of context-free languages is closed under the regular operations.

PROOF SKETCH Let $G_1 = (V_1, \Sigma, P_1, S_1)$ and $G_2 = (V_2, \Sigma, P_2, S_2)$ be context-free grammars. We can construct new grammars for each regular operation:

THEOREM The class of context-free languages is closed under the regular operations.

PROOF SKETCH Let $G_1 = (V_1, \Sigma, P_1, S_1)$ and $G_2 = (V_2, \Sigma, P_2, S_2)$ be context-free grammars. We can construct new grammars for each regular operation:

For a grammar that generates $L(G_1) \cup L(G_2)$, we add new production rules

$$S \rightarrow S_1$$
$$S \rightarrow S_2$$

THEOREM The class of context-free languages is closed under the regular operations.

PROOF SKETCH Let $G_1 = (V_1, \Sigma, P_1, S_1)$ and $G_2 = (V_2, \Sigma, P_2, S_2)$ be context-free grammars. We can construct new grammars for each regular operation:

For a grammar that generates $L(G_1) \circ L(G_2)$, we add a new production rule

$$S \rightarrow S_1 S_2.$$

THEOREM The class of context-free languages is closed under the regular operations.

PROOF SKETCH Let $G_1 = (V_1, \Sigma, P_1, S_1)$ and $G_2 = (V_2, \Sigma, P_2, S_2)$ be context-free grammars. We can construct new grammars for each regular operation:

For a grammar that generates $L(G_1)^*$, we add new production rules

$$S \to S_1 S \mid \varepsilon.$$

Reversal

Suppose that L is a context-free language.

Is L^R a context-free language?

Reversal

Suppose that L is a context-free language.

Is L^{R} a context-free language?

Yes! Proof idea:

$S \rightarrow A$ `bites` B

$A \rightarrow$ `man`

$B \rightarrow$ `dog`

Reversal

Suppose that L is a context-free language.

Is L^{R} a context-free language?

Yes! Proof idea:

$S \rightarrow A$ `bites` B	$S \rightarrow B$ `setib` A
$A \rightarrow$ `man`	$A \rightarrow$ `nam`
$B \rightarrow$ `dog`	$B \rightarrow$ `god`

Reversal

PROOF SKETCH Given a CFG $G = (V, \Sigma, P, S)$, construct grammar $G^R = (V, \Sigma, P^R, S)$:

Define $P^R = \{A \to \alpha^R \text{ where } A \to \alpha \in P\}$.

That is, perform normal string reversal over the body of each production rule.

We've seen that if L_1 and L_2 are context-free languages, then

$L_1 \cup L_2$ is context-free

$L_1 \circ L_2$ is context-free

L_1^* is context-free

L_1^R is context-free

What about intersection, complement, or difference?

We've seen that if L_1 and L_2 are context-free languages, then

$L_1 \cup L_2$ is context-free

$L_1 \circ L_2$ is context-free

L_1^* is context-free

L_1^R is context-free

What about intersection, complement, or difference?

There isn't an obvious way to do these using CFG productions…

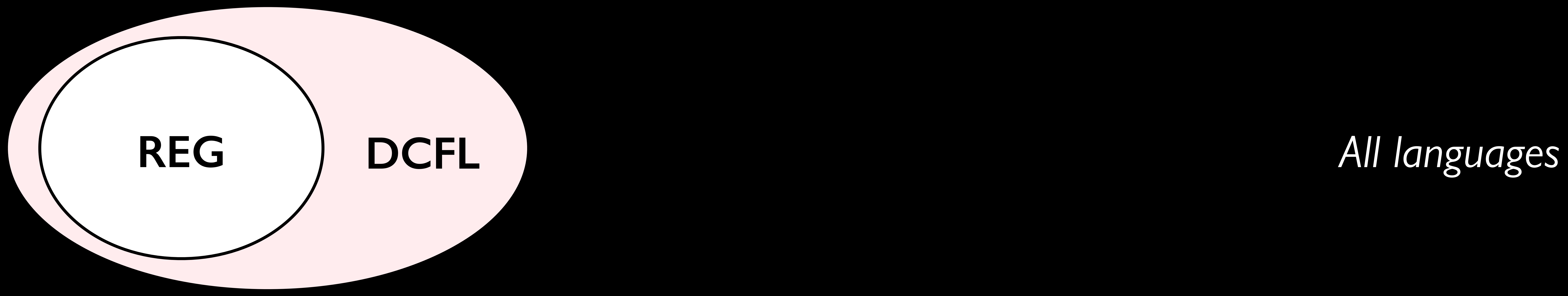
If we want to prove *non*-closure for these operations, we first need to know that there are languages that aren't context-free!

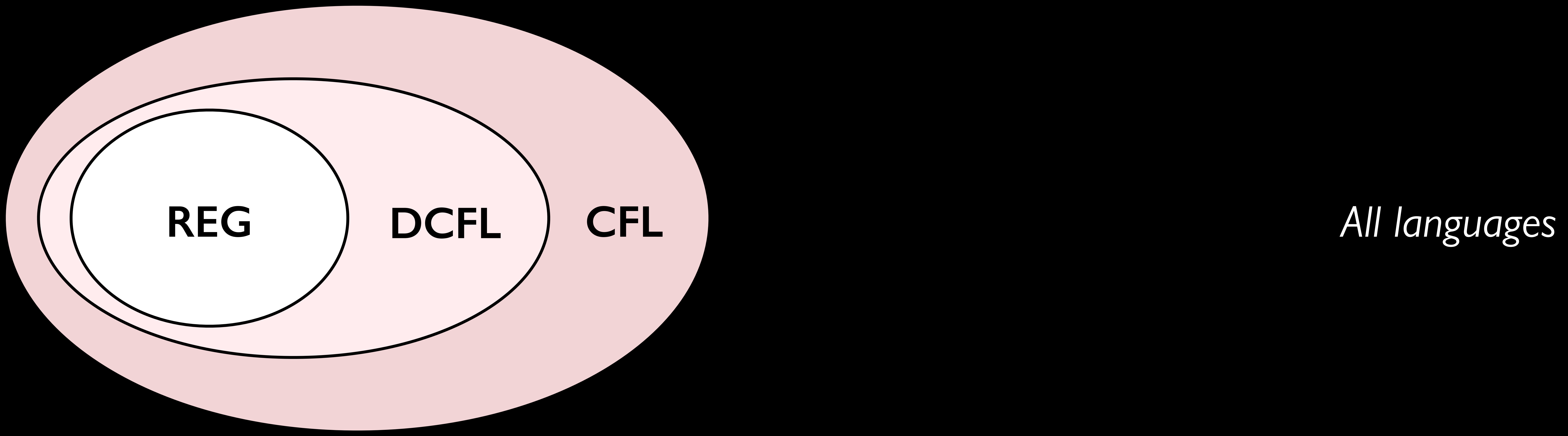
Where are we *now*?

Are some problems inherently harder than others?

REG

All languages


REG
DCFL
All languages


REG
DCFL
CFL
All languages


What sort of languages are out here?
REG
DCFL
CFL
All languages

The Pumping Lemma for Regular Languages

If L is a regular language, there must be a DFA D that recognizes L.

A sufficiently long string $s \in L$ must visit some state in D twice.

- This means s went through a loop in the DFA D.
- By replicating the characters that went through the loop in D, we can "pump" a portion of s to produce new strings in the language.

The Pumping Lemma for Regular Languages

For any regular language L,

there exists a positive integer p such that

for any string $s \in L$ such that $|s| \geq p$,

there exist strings x, y, and z such that

$s = xyz$	*s can be broken into three pieces,*
$\lvert xy \rvert \leq p$	*where the first two pieces occur at the start of the string,*
$y \neq \varepsilon$	*the middle part isn't empty, and*
$xy^iz \in L$	*the middle piece can be repeated zero or more times.*

Intuition for the Pumping Lemma

The model of computation used has a finite description.

For sufficiently long strings, the model of computation must repeat some step of the computation to recognize the string.

Under the right circumstances, we can iterate this repeated step zero or more times to produce more strings.

The Pumping Lemma for CFLs

$$S \rightarrow [P]$$

$$P \rightarrow RR \mid \mathrm{a}$$

$$R \rightarrow (P) \mid \mathrm{b}$$

$S \rightarrow [P]$

$P \rightarrow RR \mid a$

$R \rightarrow (P) \mid b$

$\boxed{S}$

$$S \rightarrow [P]$$

$$P \rightarrow RR \mid a$$

$$R \rightarrow (P) \mid b$$

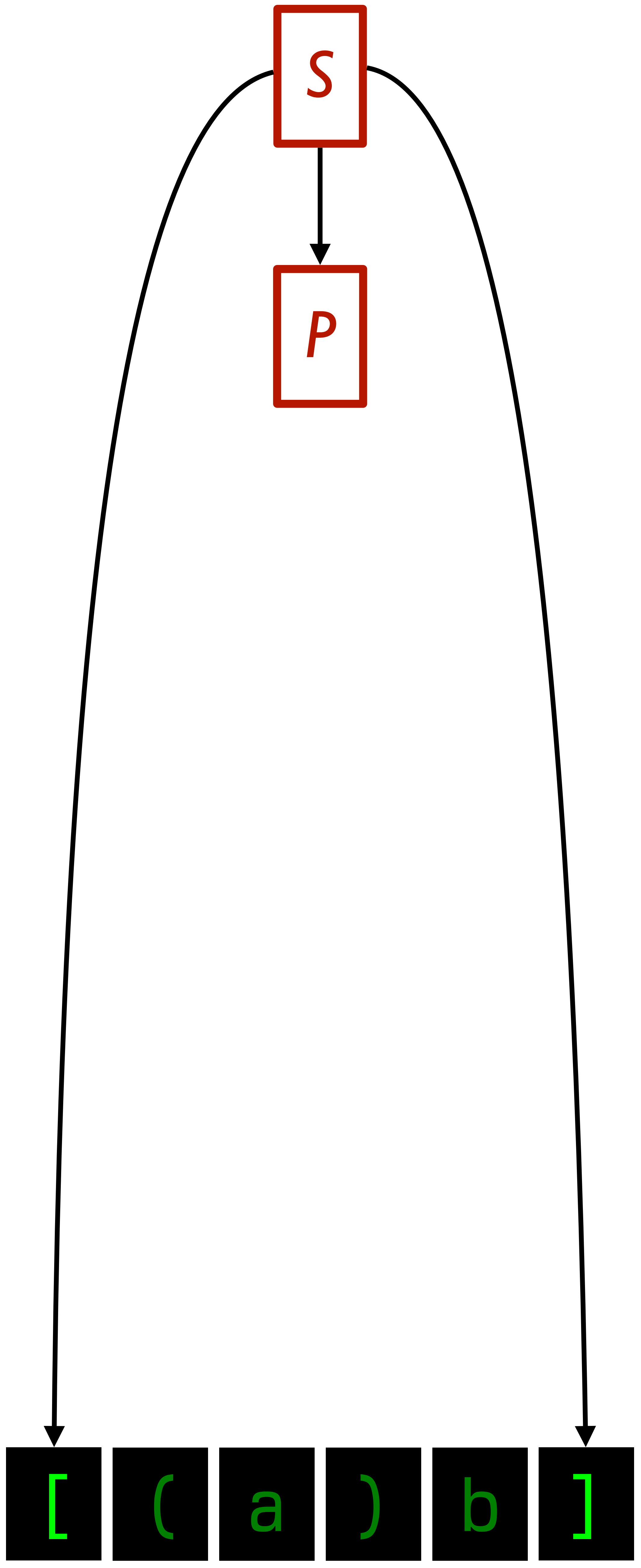
$\boxed{S}$

[	(	a	)	b	]

$S \rightarrow [P]$

$P \rightarrow RR \mid a$

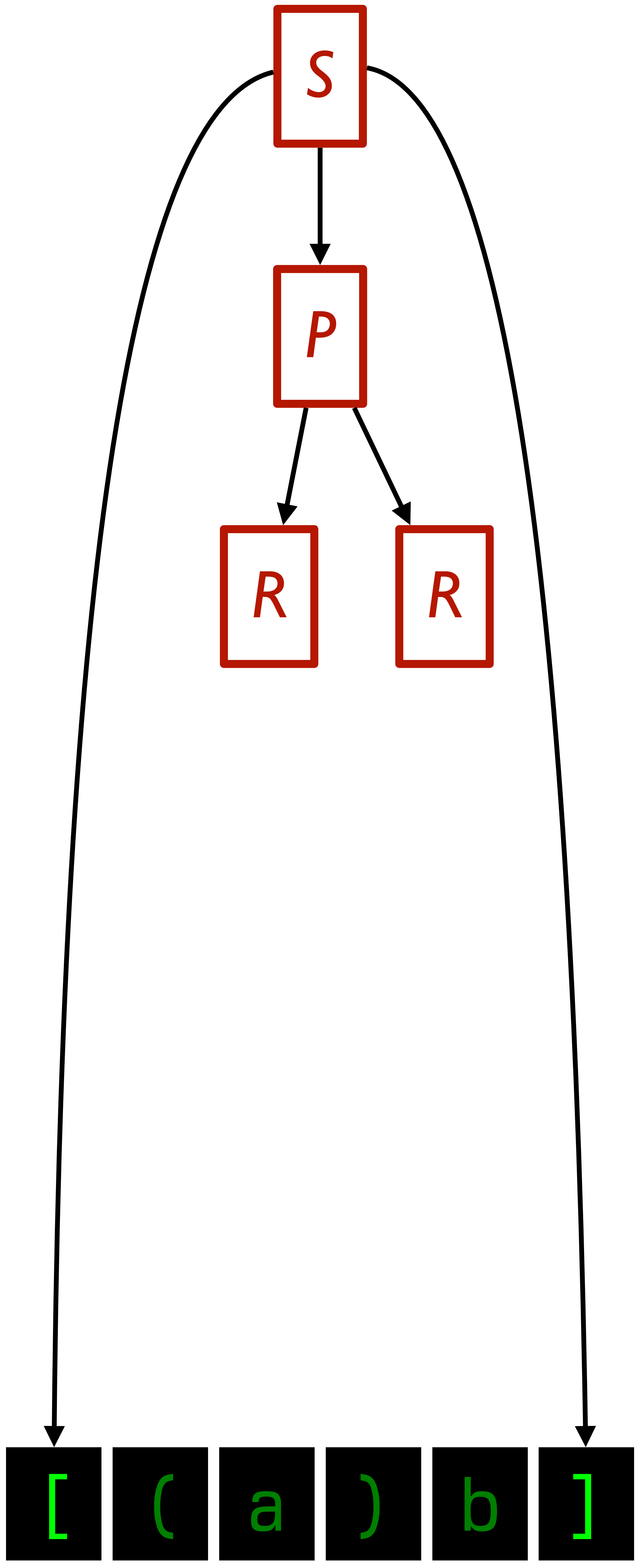
$R \rightarrow (P) \mid b$



$S \rightarrow [P]$

$P \rightarrow RR \mid a$

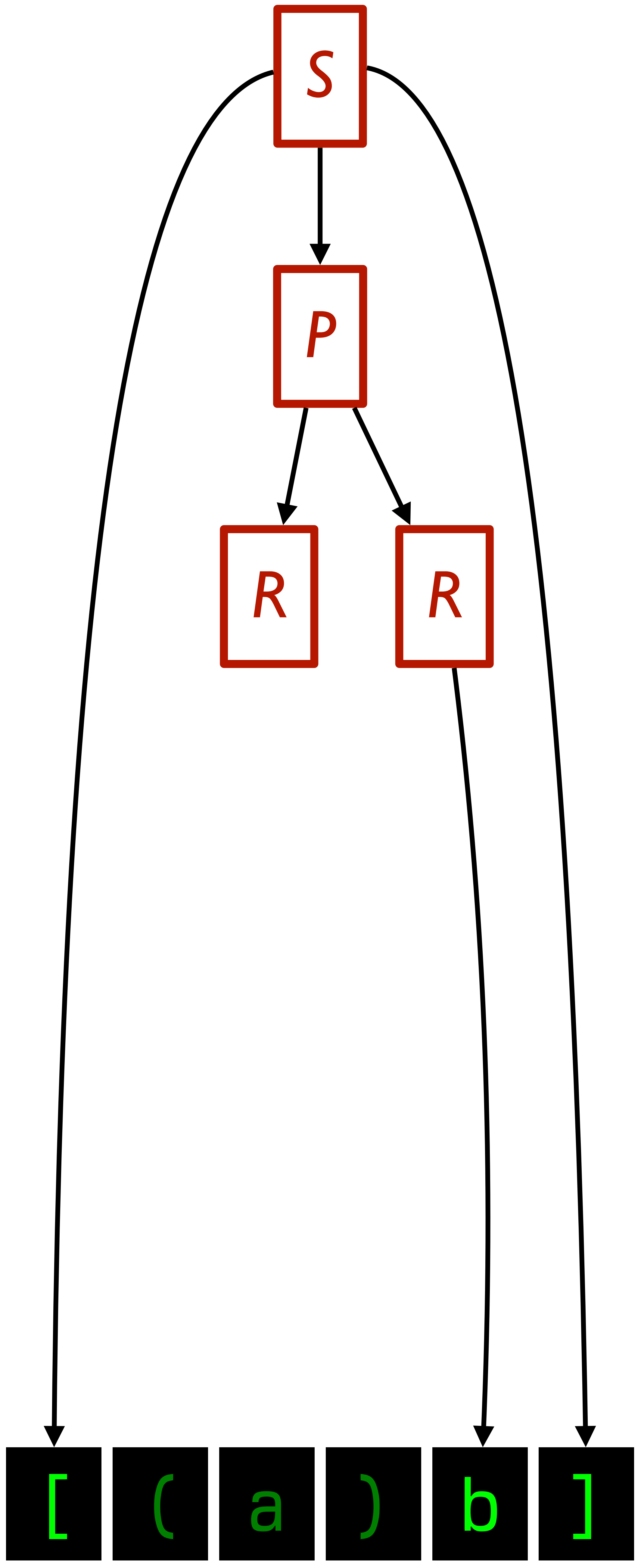
$R \rightarrow (P) \mid b$



$S \rightarrow [P]$

$P \rightarrow RR \mid a$

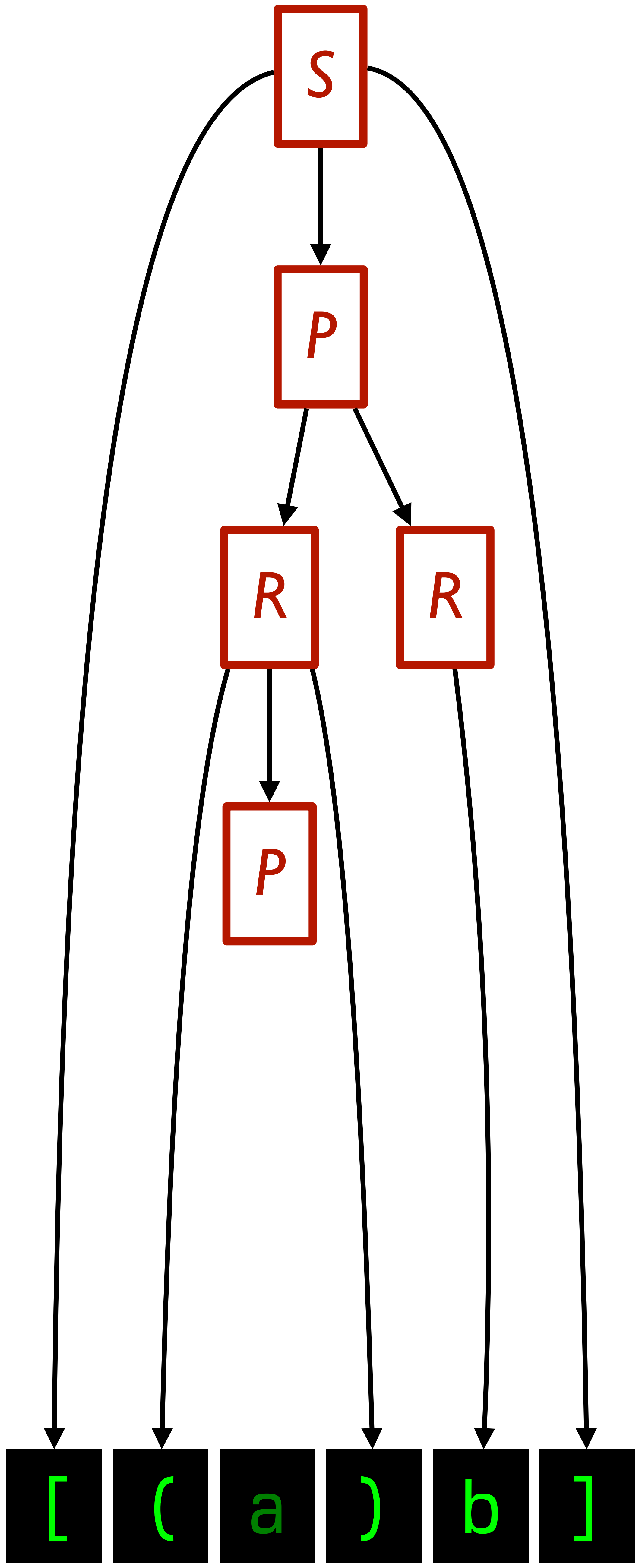
$R \rightarrow (P) \mid b$



$S \rightarrow [P]$

$P \rightarrow RR \mid \mathrm{a}$

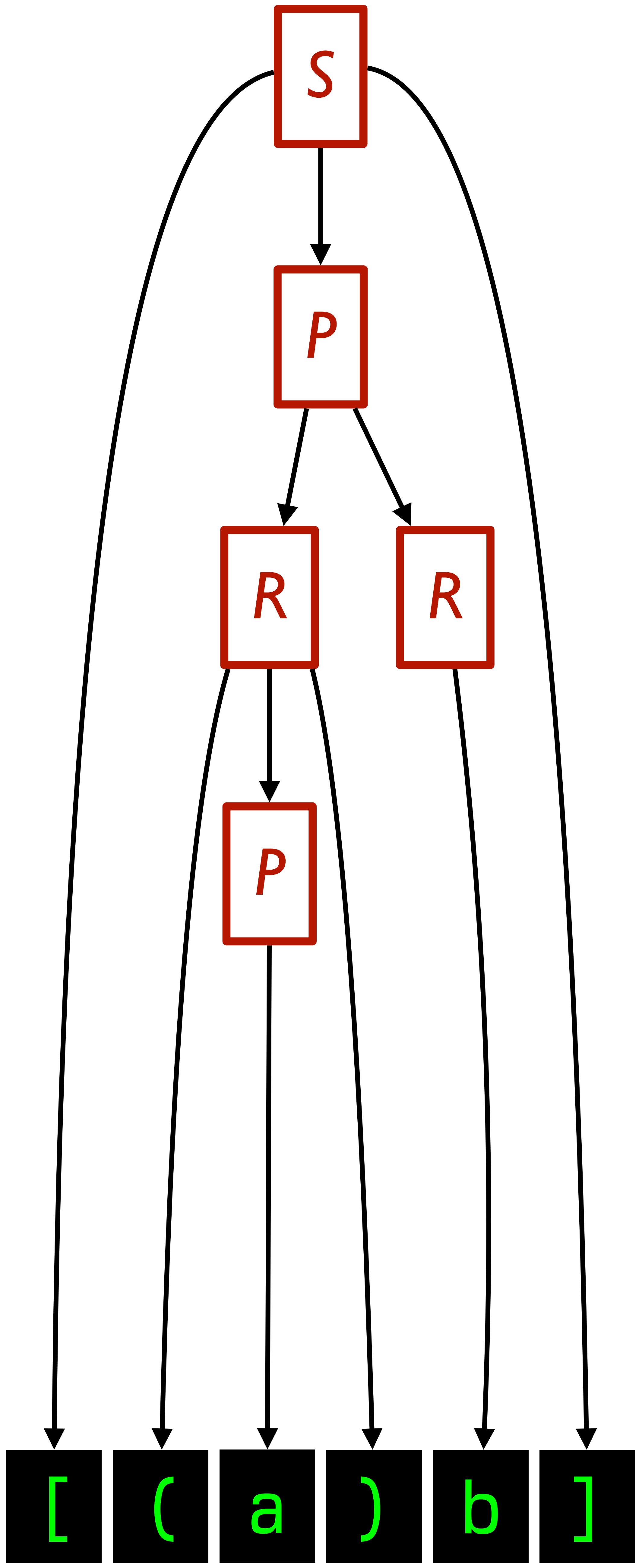
$R \rightarrow (P) \mid \mathrm{b}$



$S \rightarrow [P]$

$P \rightarrow RR \mid a$

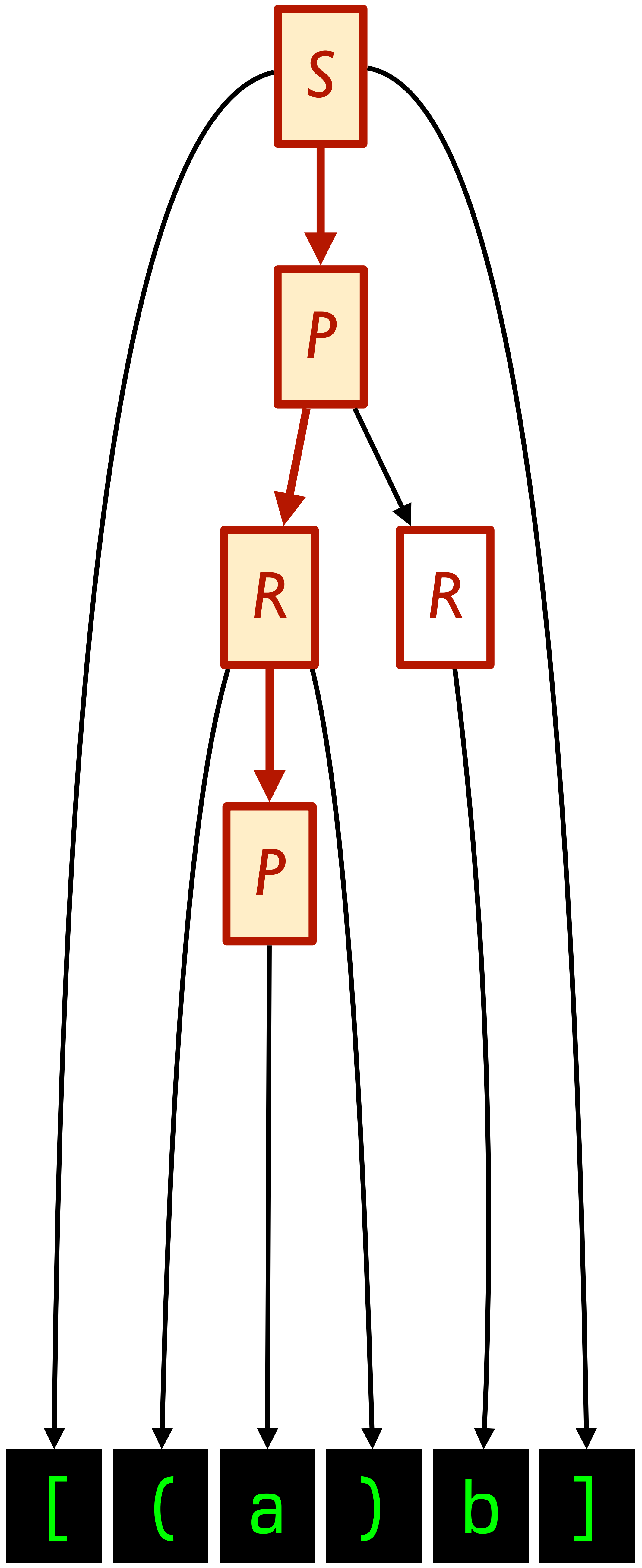
$R \rightarrow (P) \mid b$



$$S \to [P]$$

$$P \to RR \mid a$$

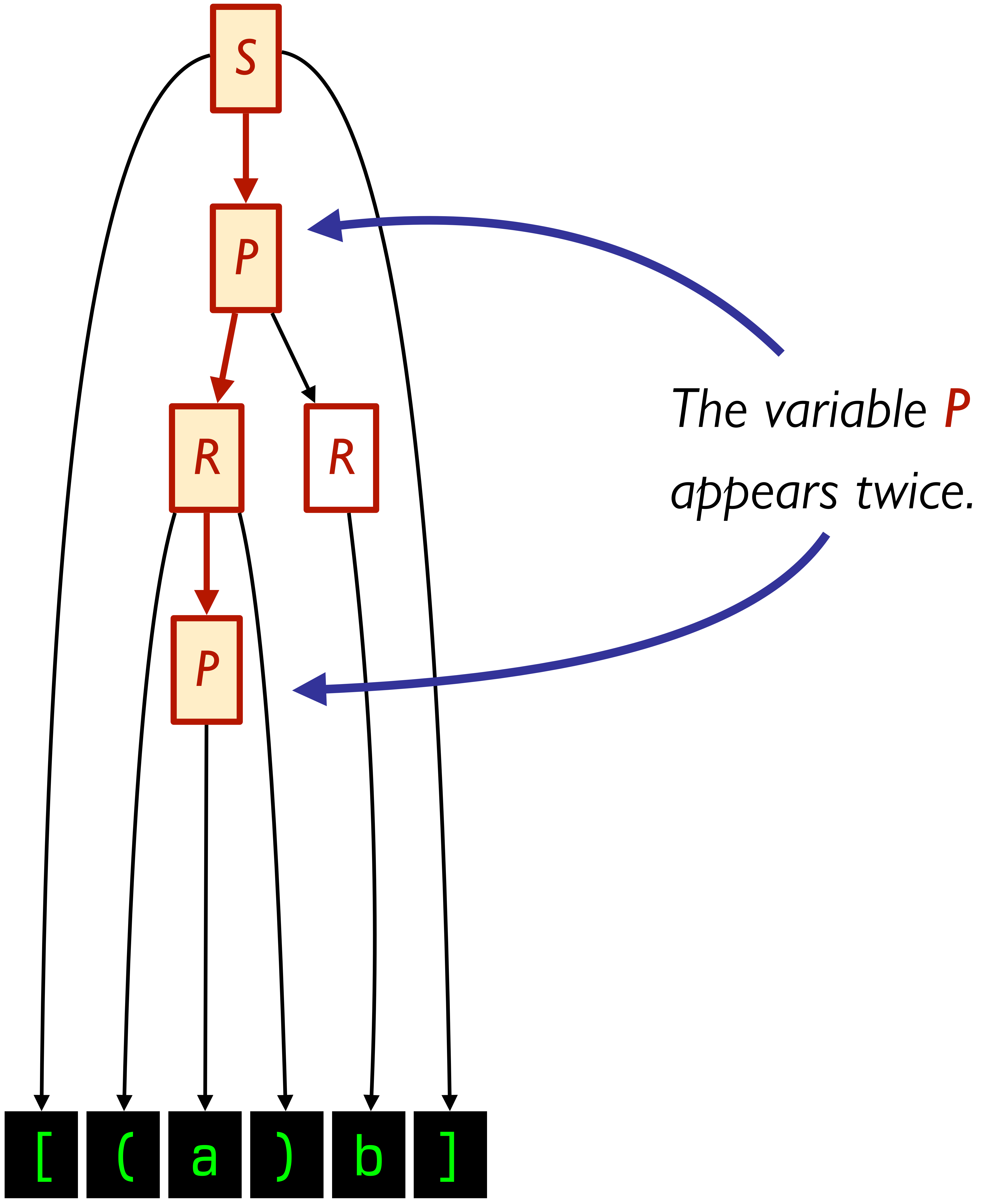
$$R \to (P) \mid b$$



$S \rightarrow [P]$

$P \rightarrow RR \mid \mathtt{a}$

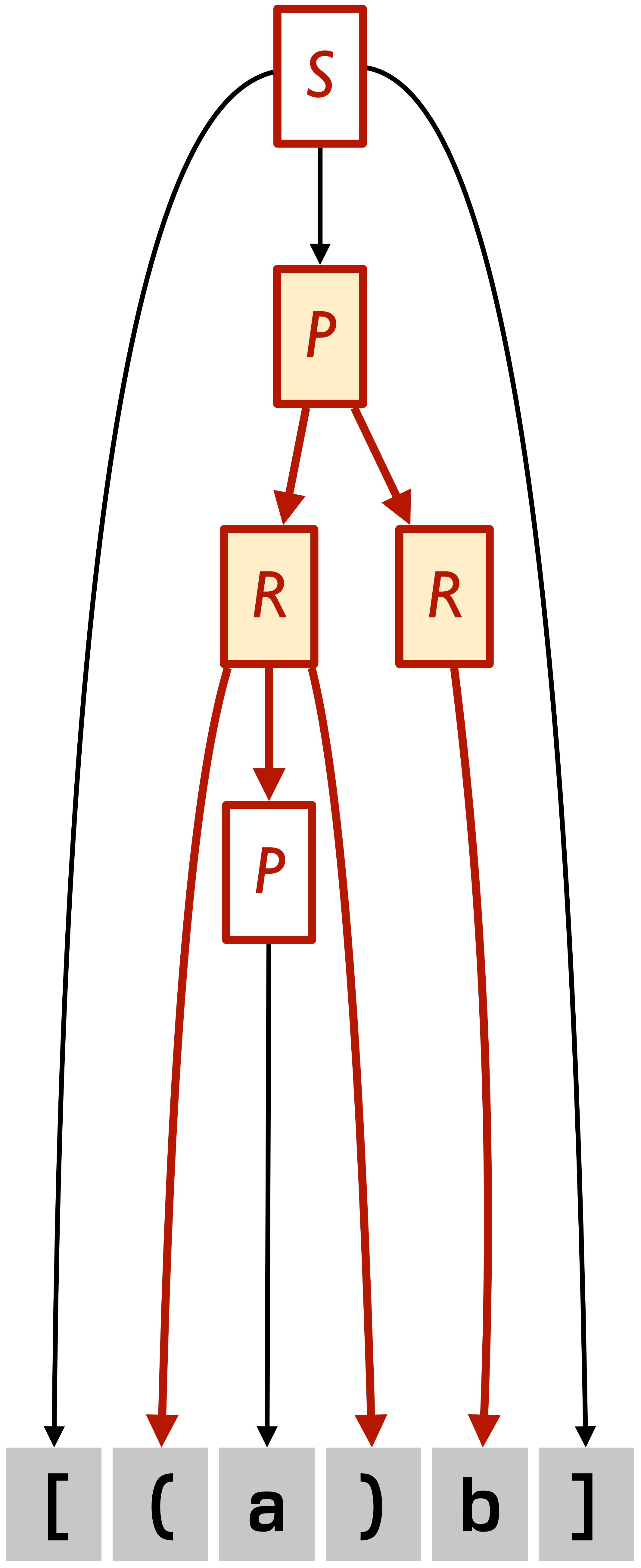
$R \rightarrow (P) \mid \mathtt{b}$



$S \rightarrow [P]$

$P \rightarrow RR \mid \texttt{a}$

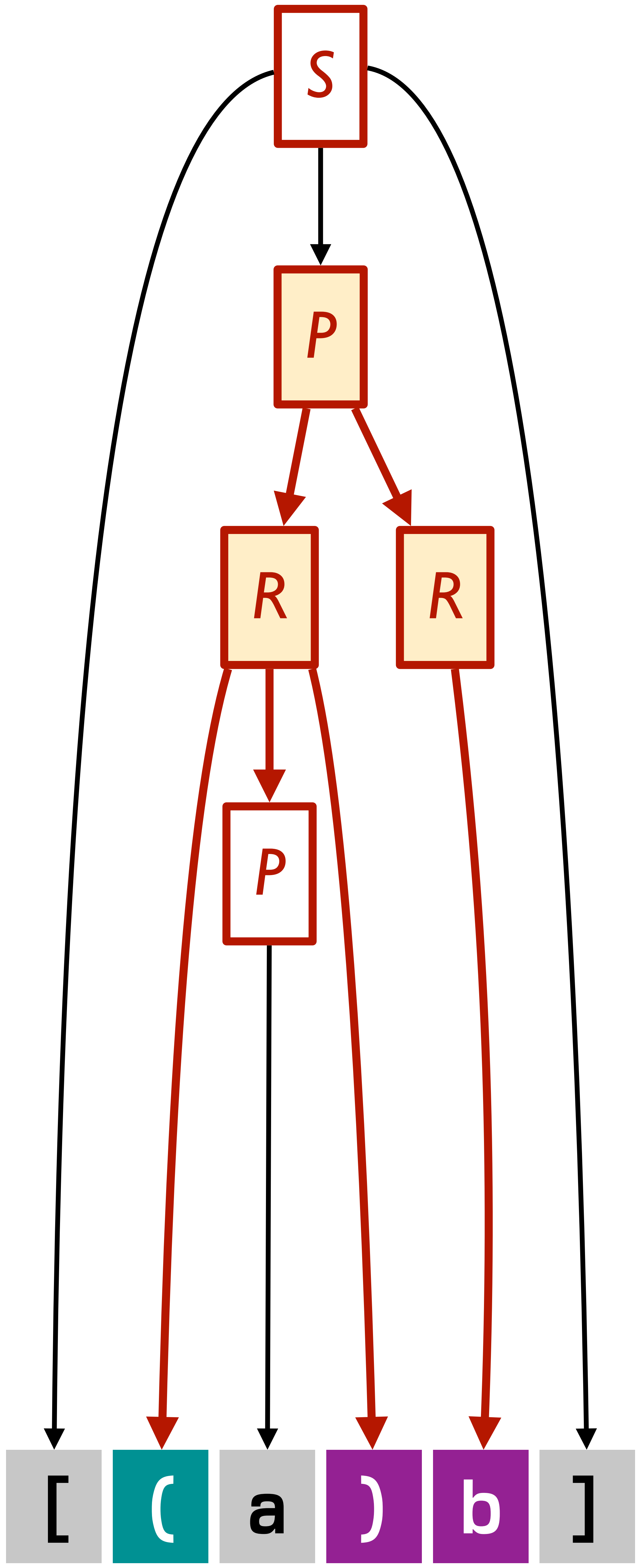
$R \rightarrow (P) \mid \texttt{b}$



$$S \rightarrow [P]$$

$$P \rightarrow RR \mid a$$

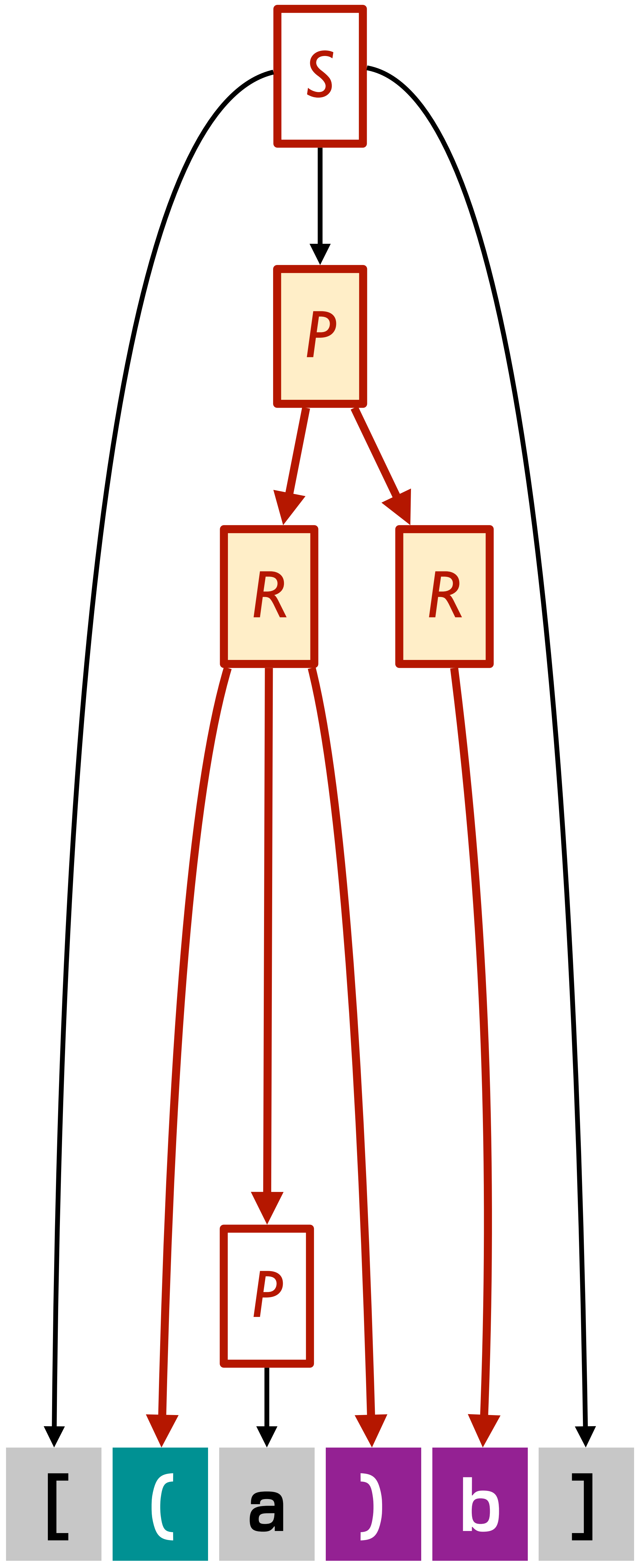
$$R \rightarrow (P) \mid b$$



$S \to [P]$

$P \to RR \mid a$

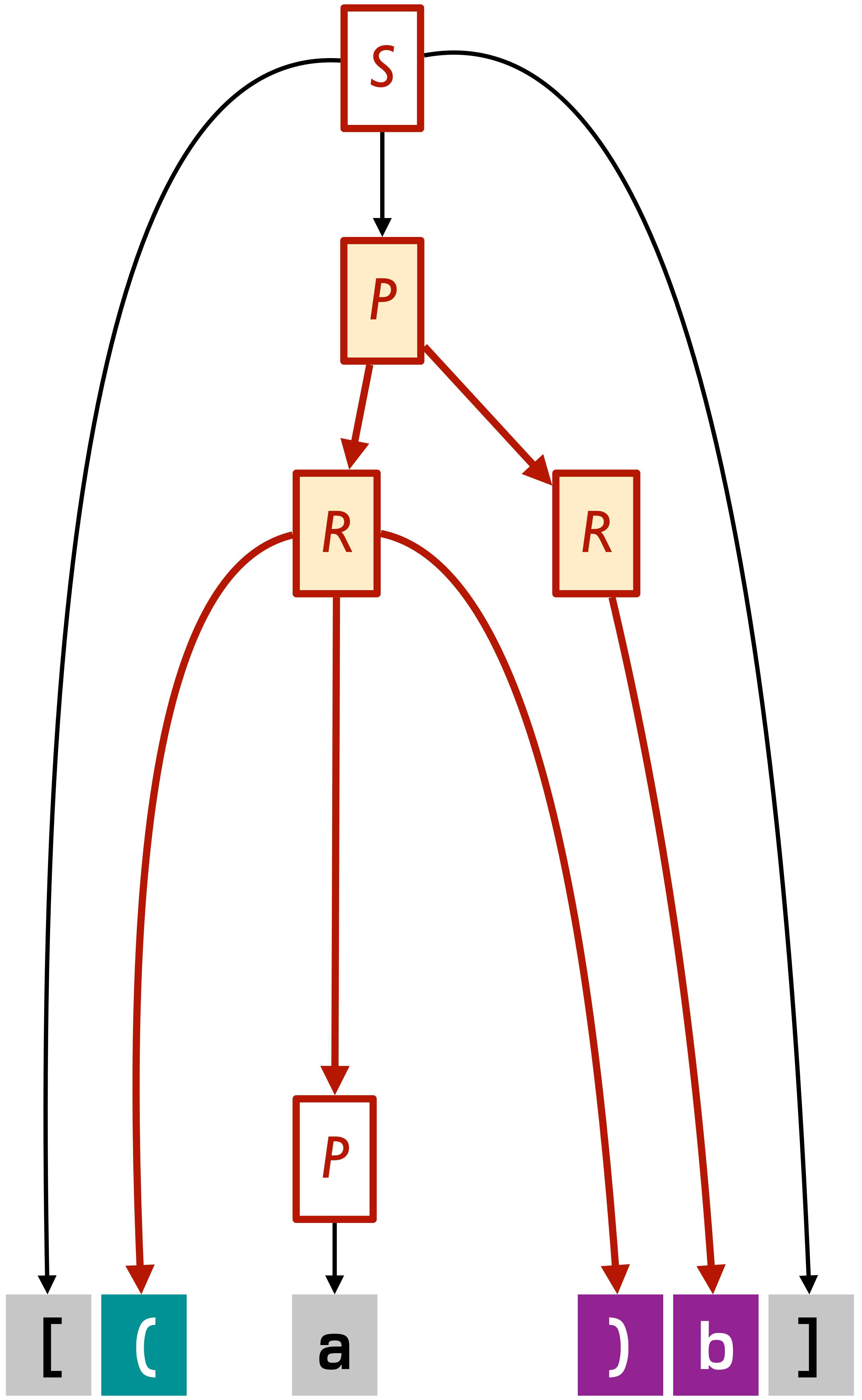
$R \to (P) \mid b$



$S \rightarrow [P]$

$P \rightarrow RR \mid \mathrm{a}$

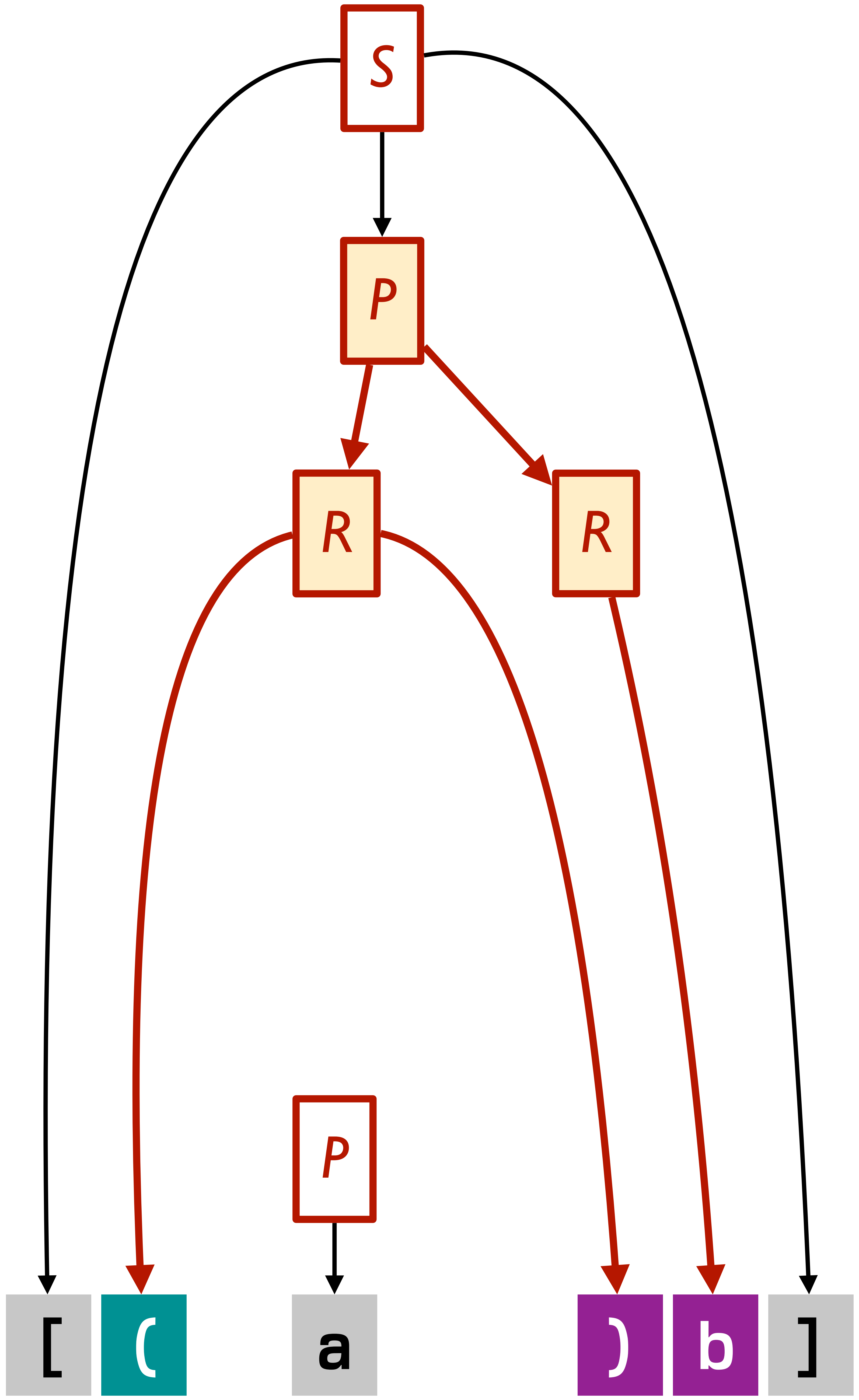
$R \rightarrow (P) \mid \mathrm{b}$



$S \rightarrow [P]$

$P \rightarrow RR \mid a$

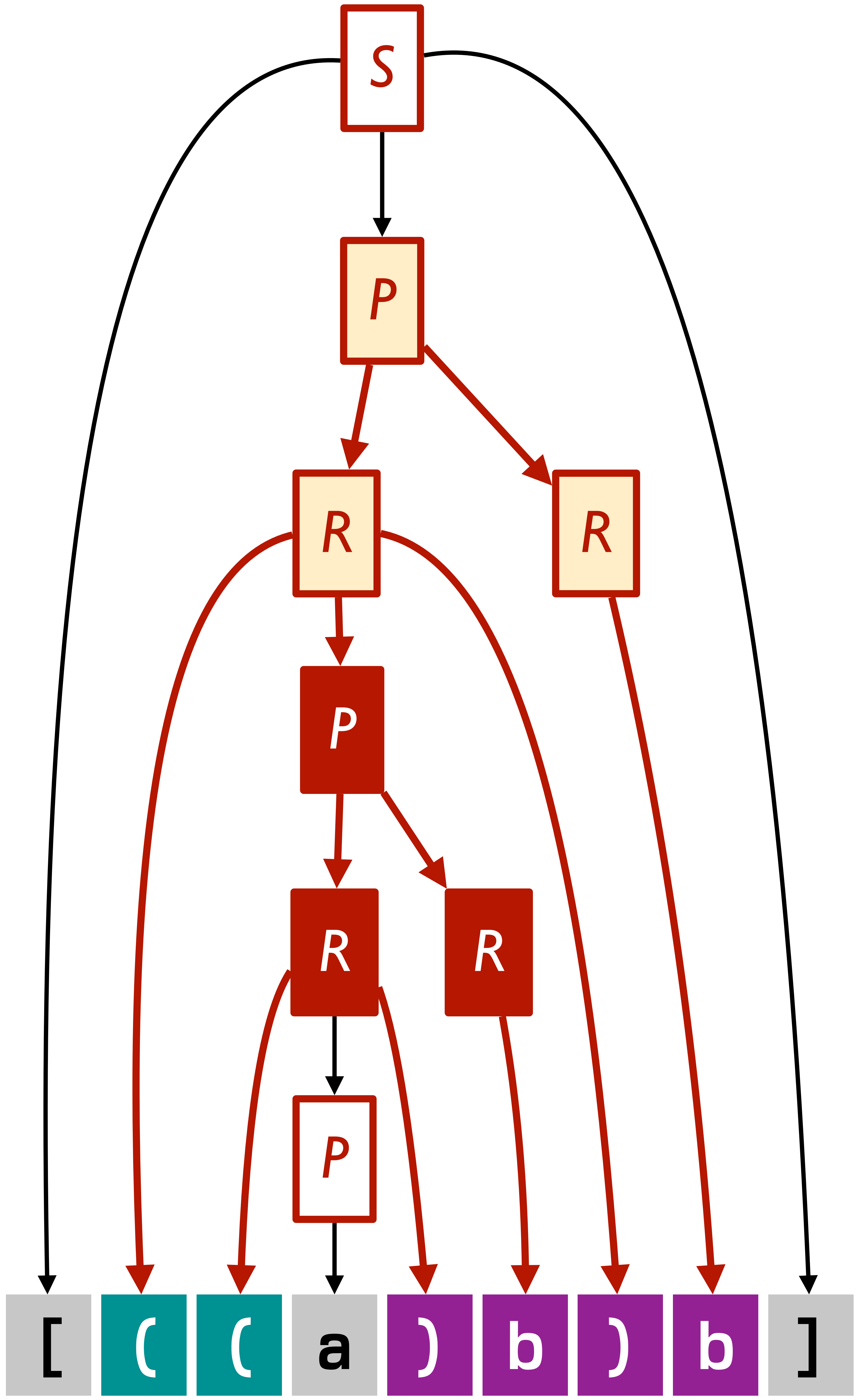
$R \rightarrow (P) \mid b$



$$S \longrightarrow [P]$$

$$P \longrightarrow RR \mid \mathtt{a}$$

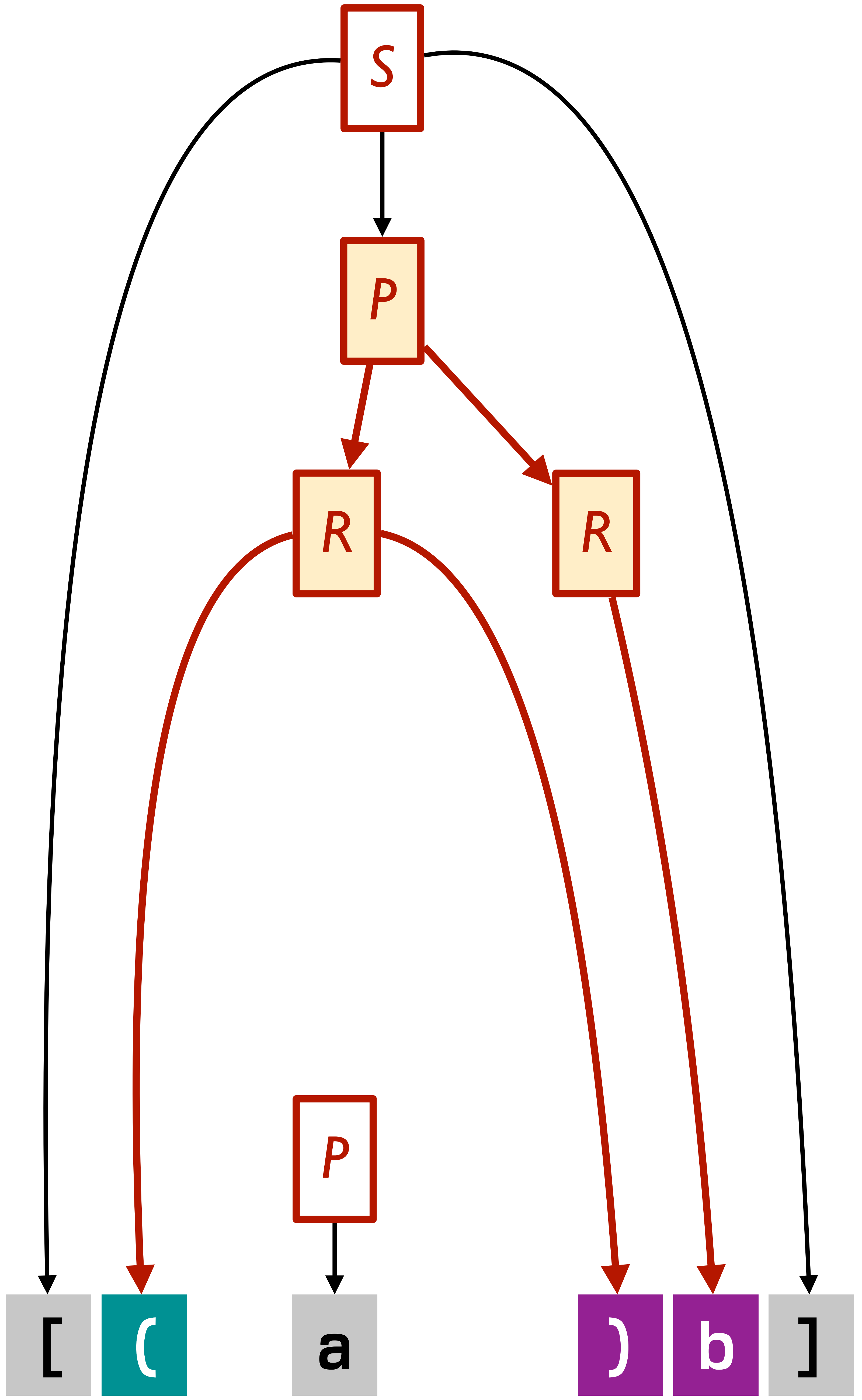
$$R \longrightarrow (P) \mid \mathtt{b}$$



$S \to [P]$

$P \to RR \mid a$

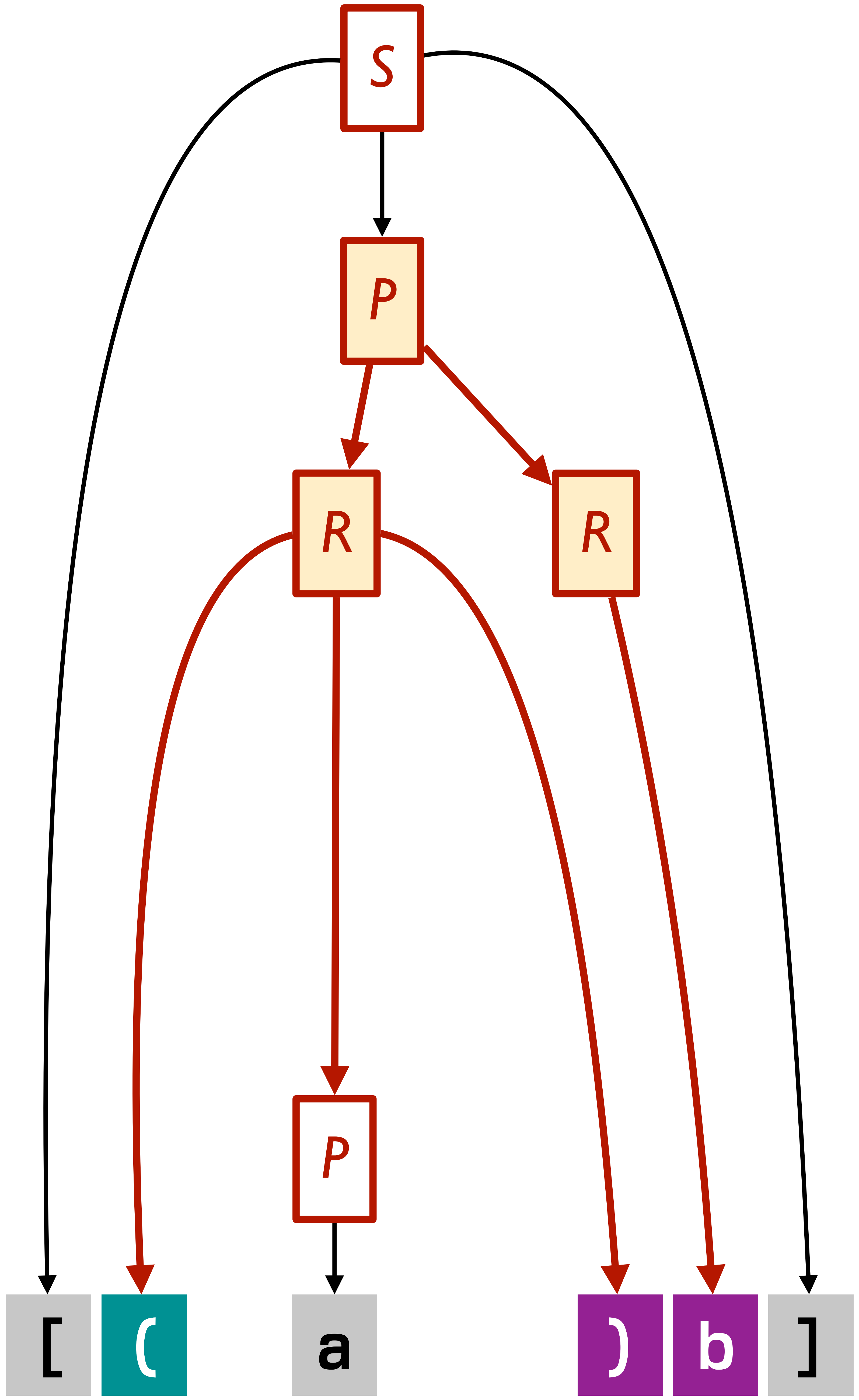
$R \to (P) \mid b$



$$S \to [P]$$

$$P \to RR \mid a$$

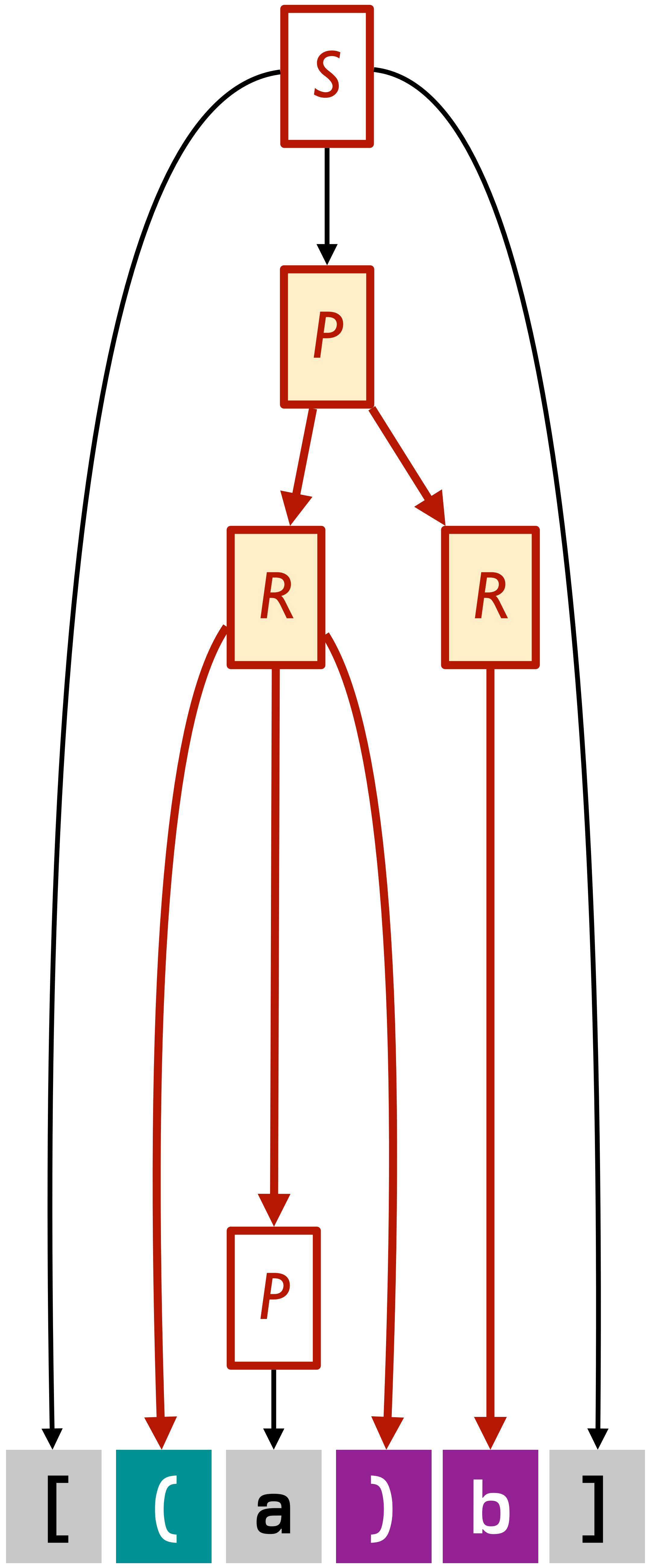
$$R \to (P) \mid b$$



$$S \rightarrow [P]$$

$$P \rightarrow RR \mid a$$

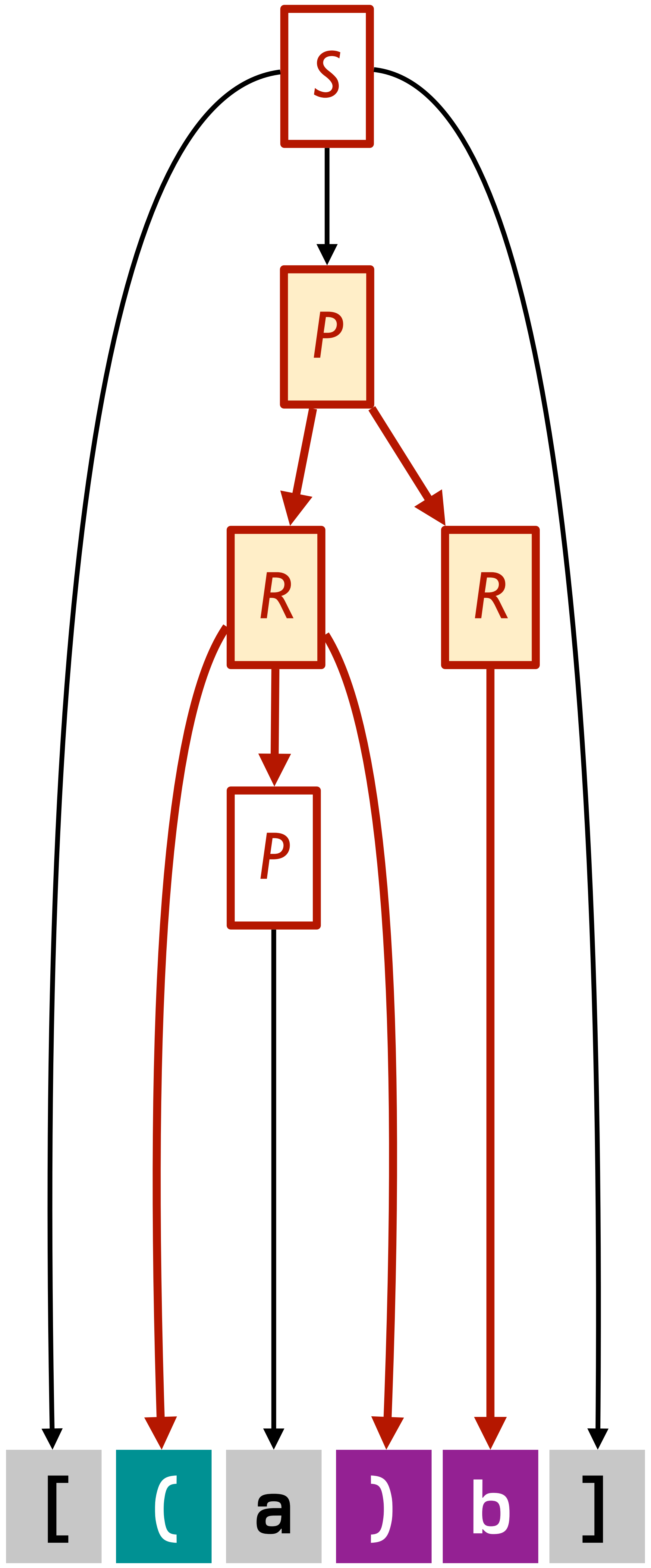
$$R \rightarrow (P) \mid b$$



$S \rightarrow [P]$

$P \rightarrow RR \mid a$

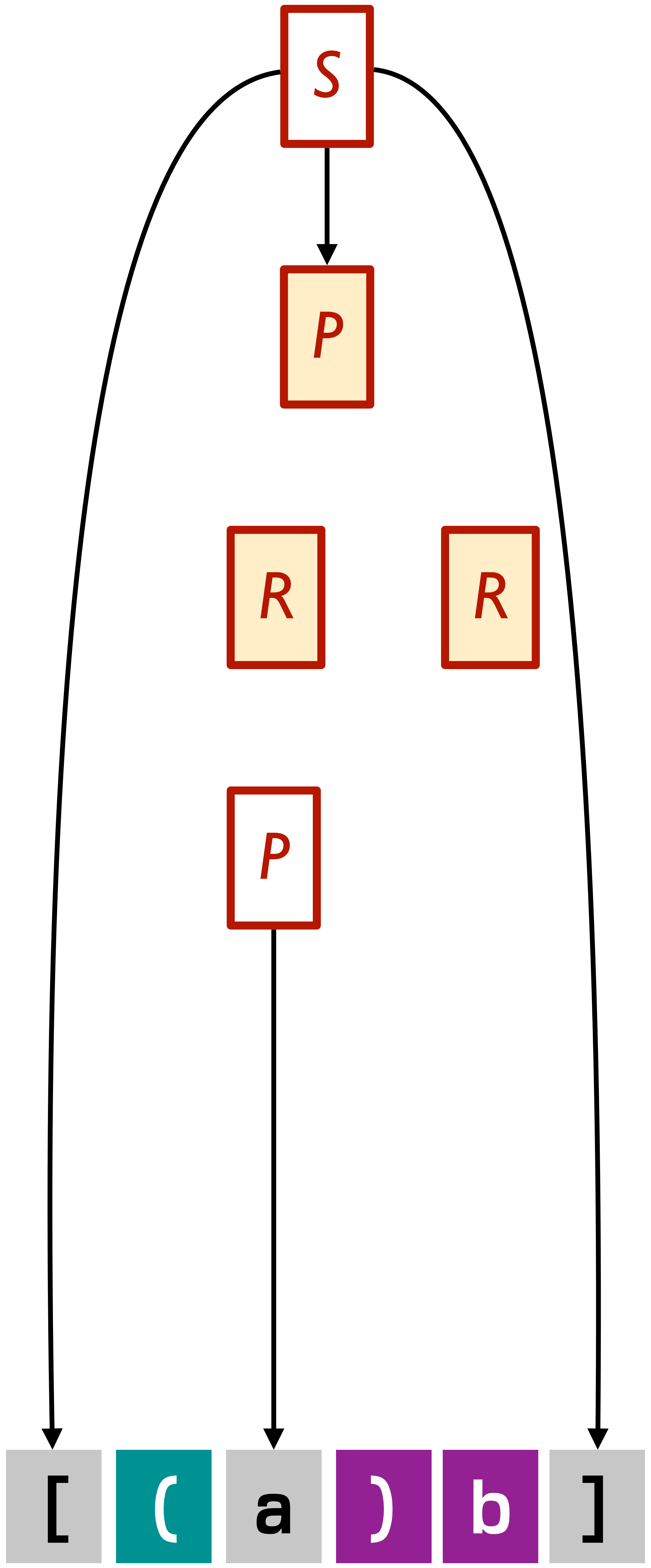
$R \rightarrow (P) \mid b$



$S \rightarrow [P]$

$P \rightarrow RR \mid a$

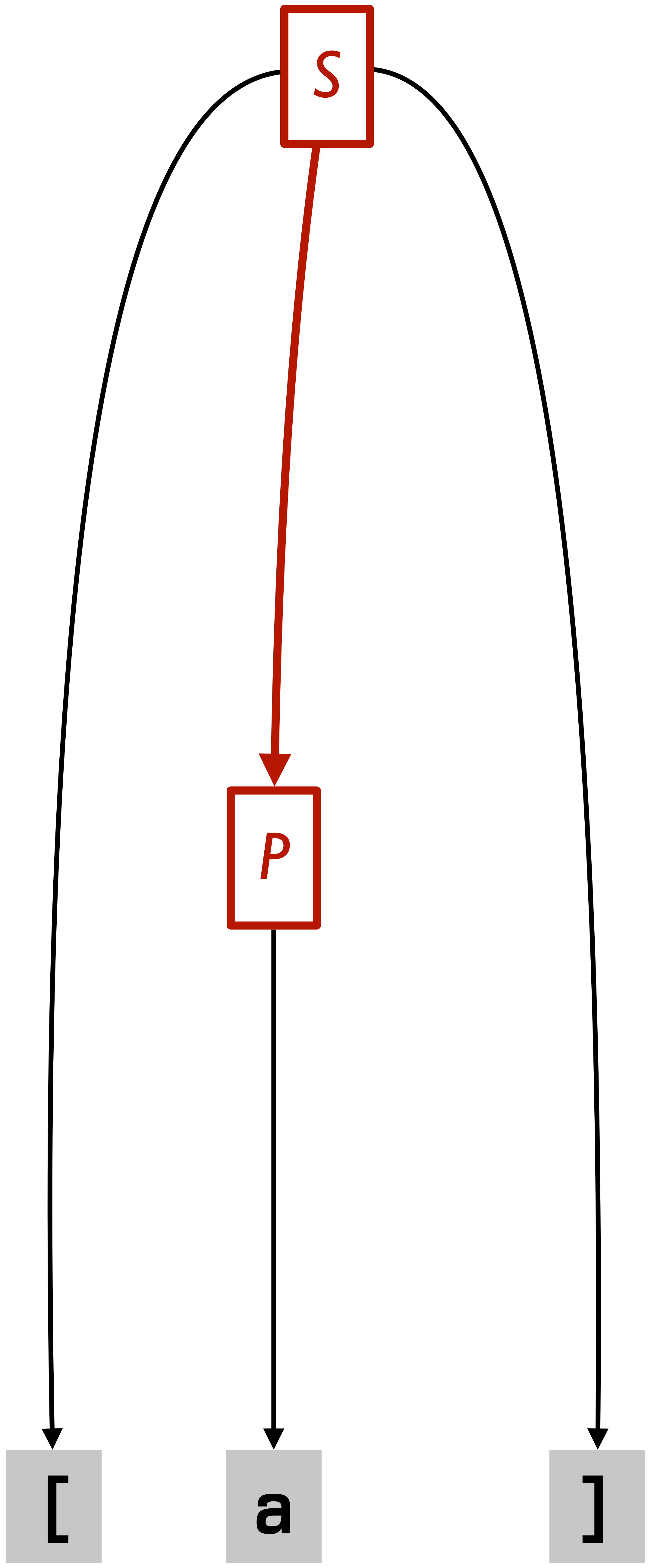
$R \rightarrow (P) \mid b$



$S \rightarrow [P]$

$P \rightarrow RR \mid a$

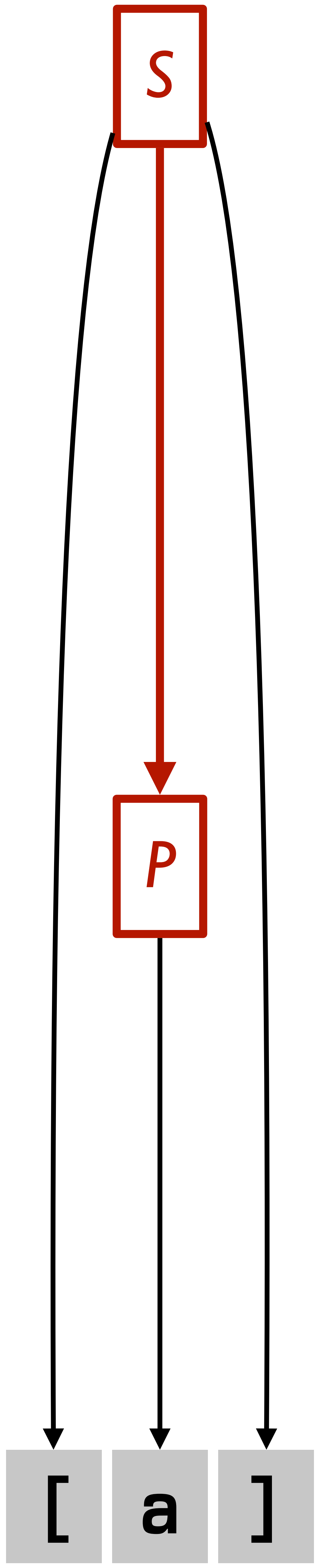
$R \rightarrow (P) \mid b$

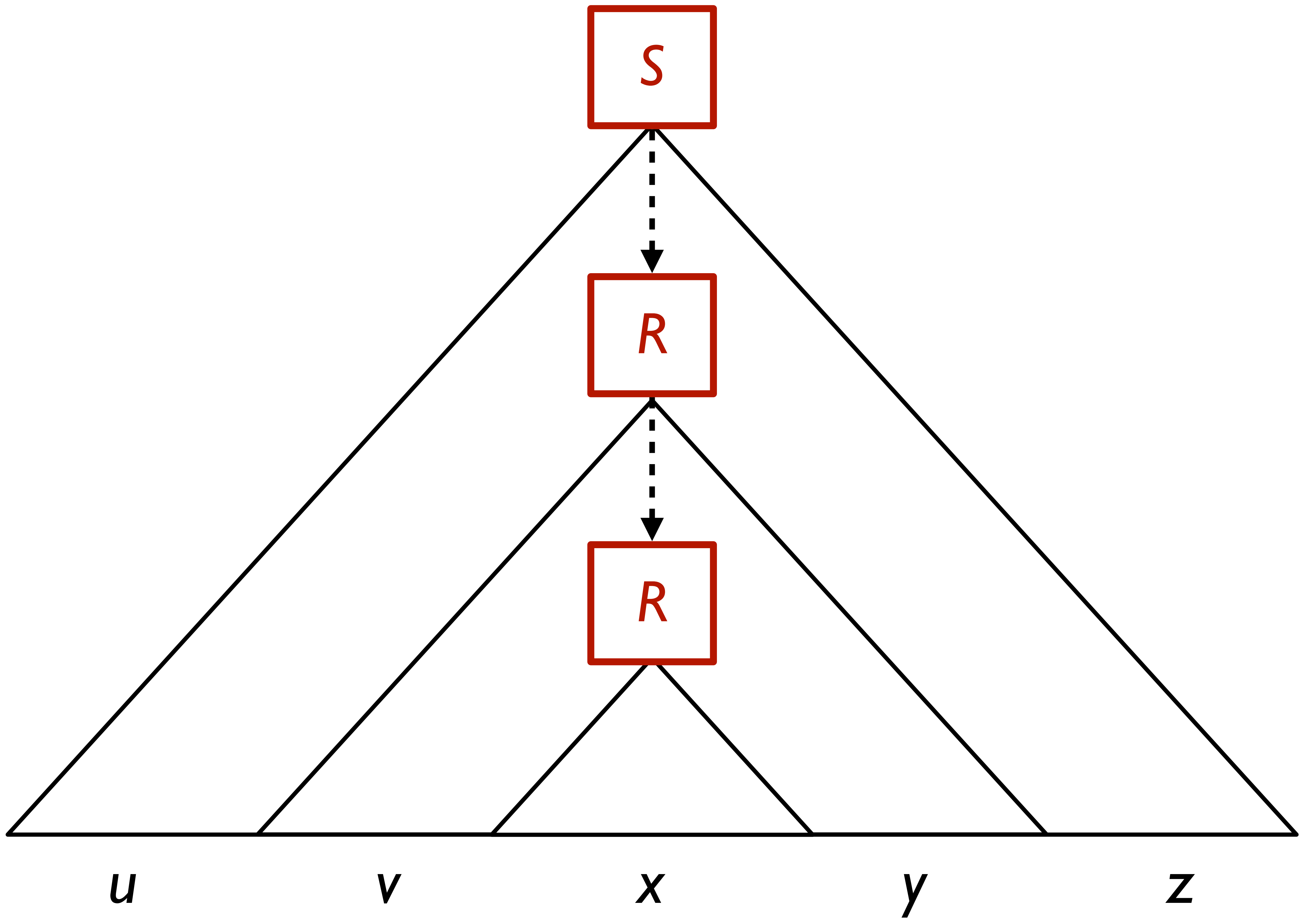


$S \rightarrow [P]$

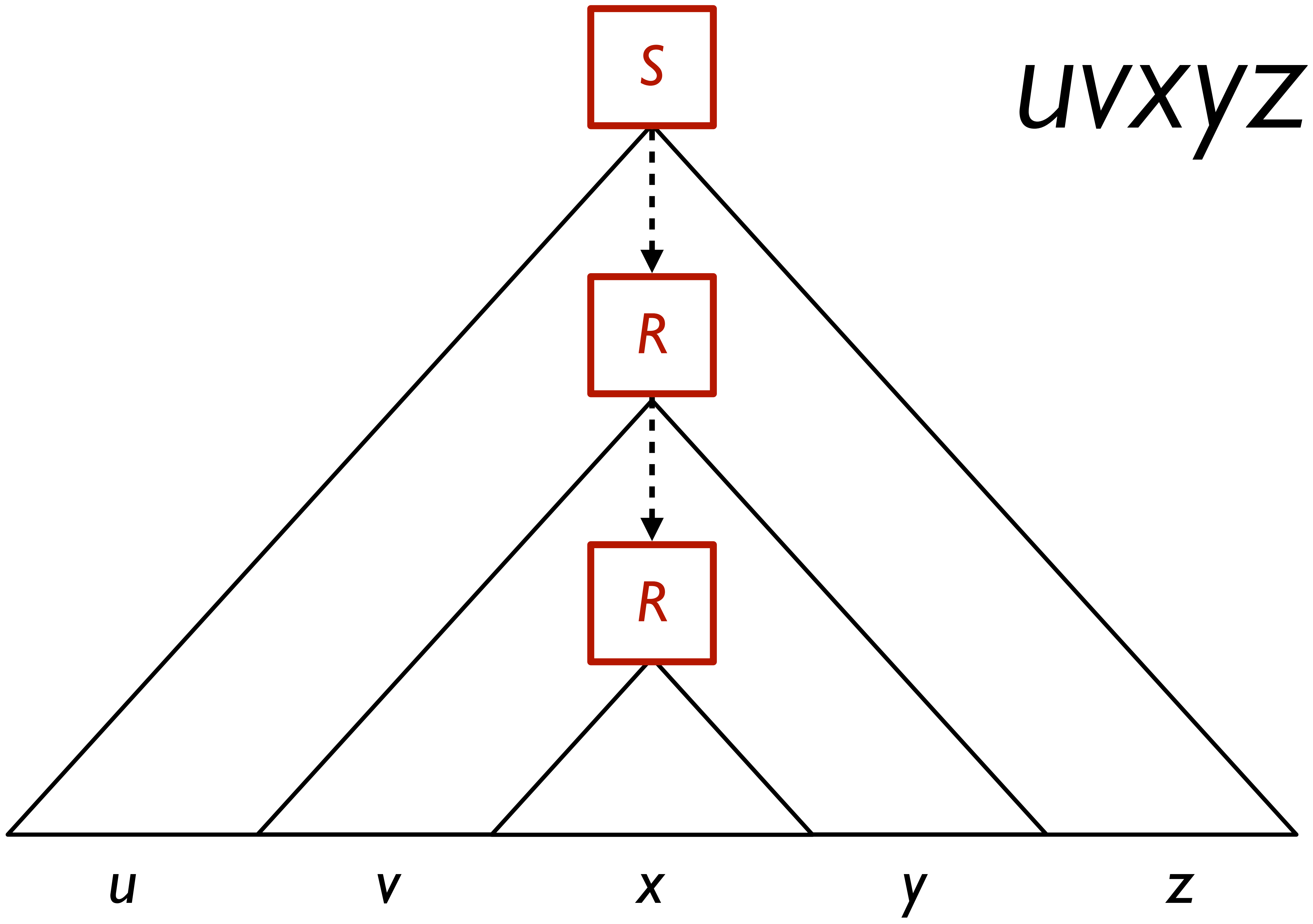
$P \rightarrow RR \mid a$

$R \rightarrow (P) \mid b$

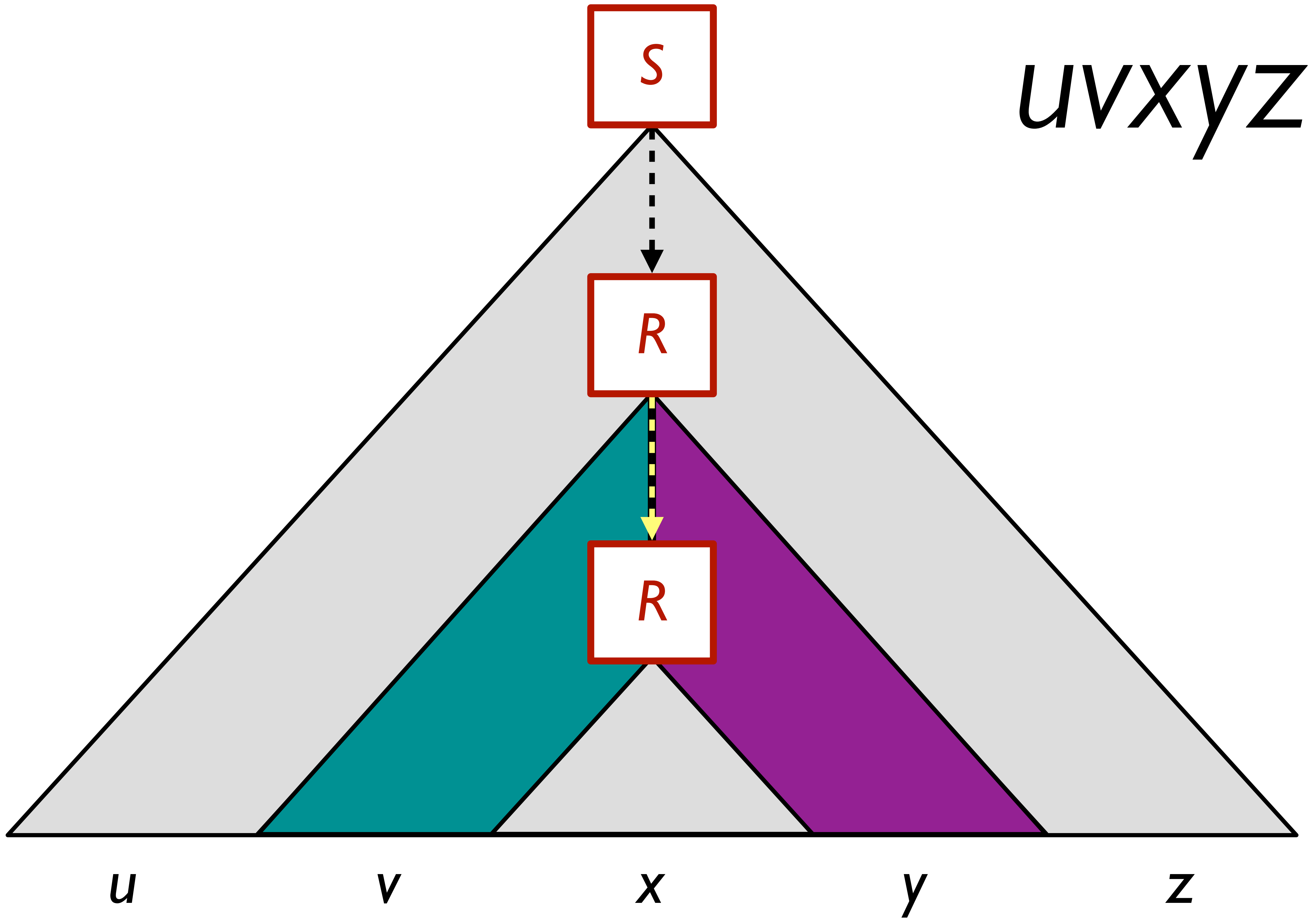



S
R
R
u
v
x
y
z

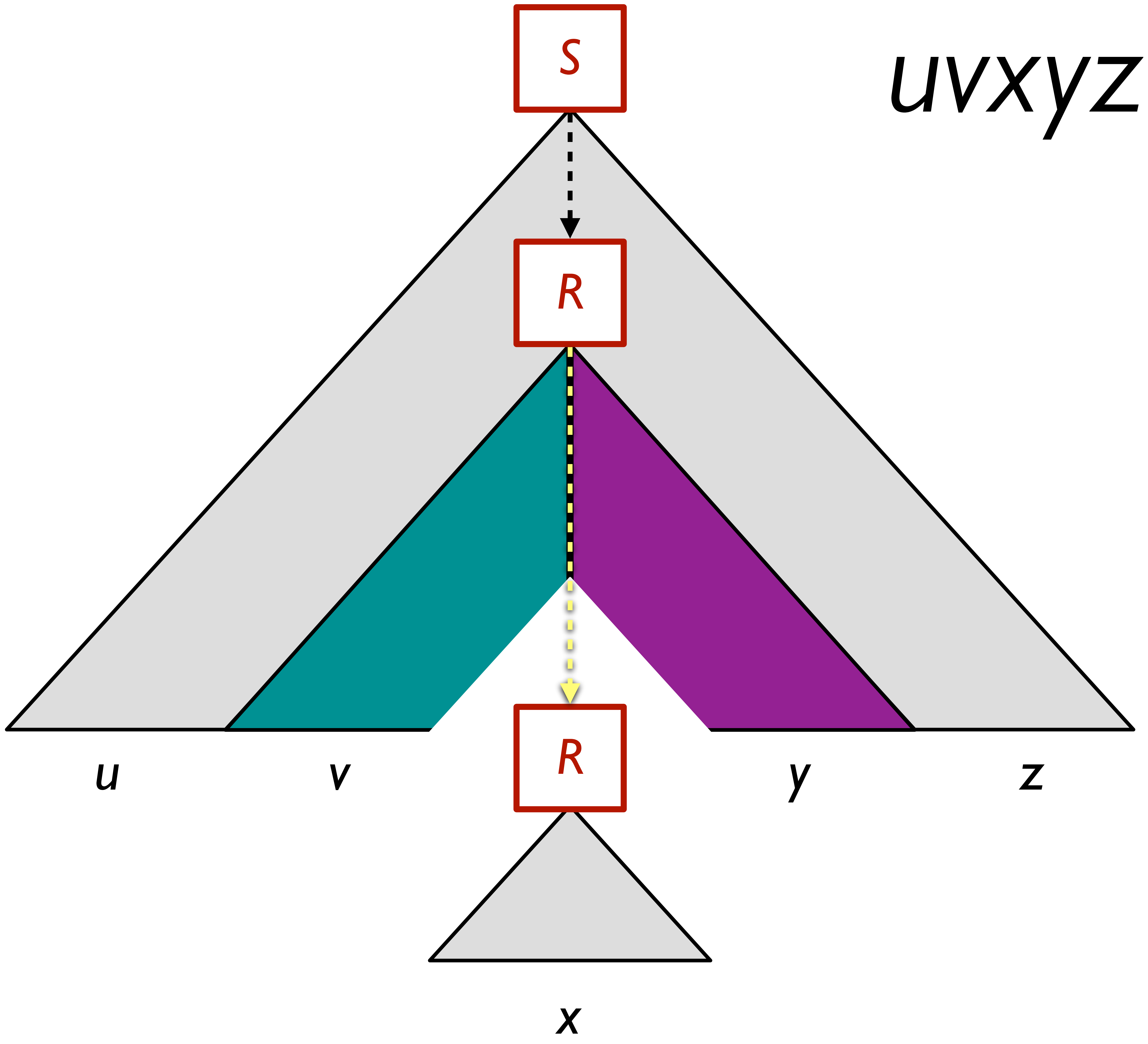
$uvxyz \in L$



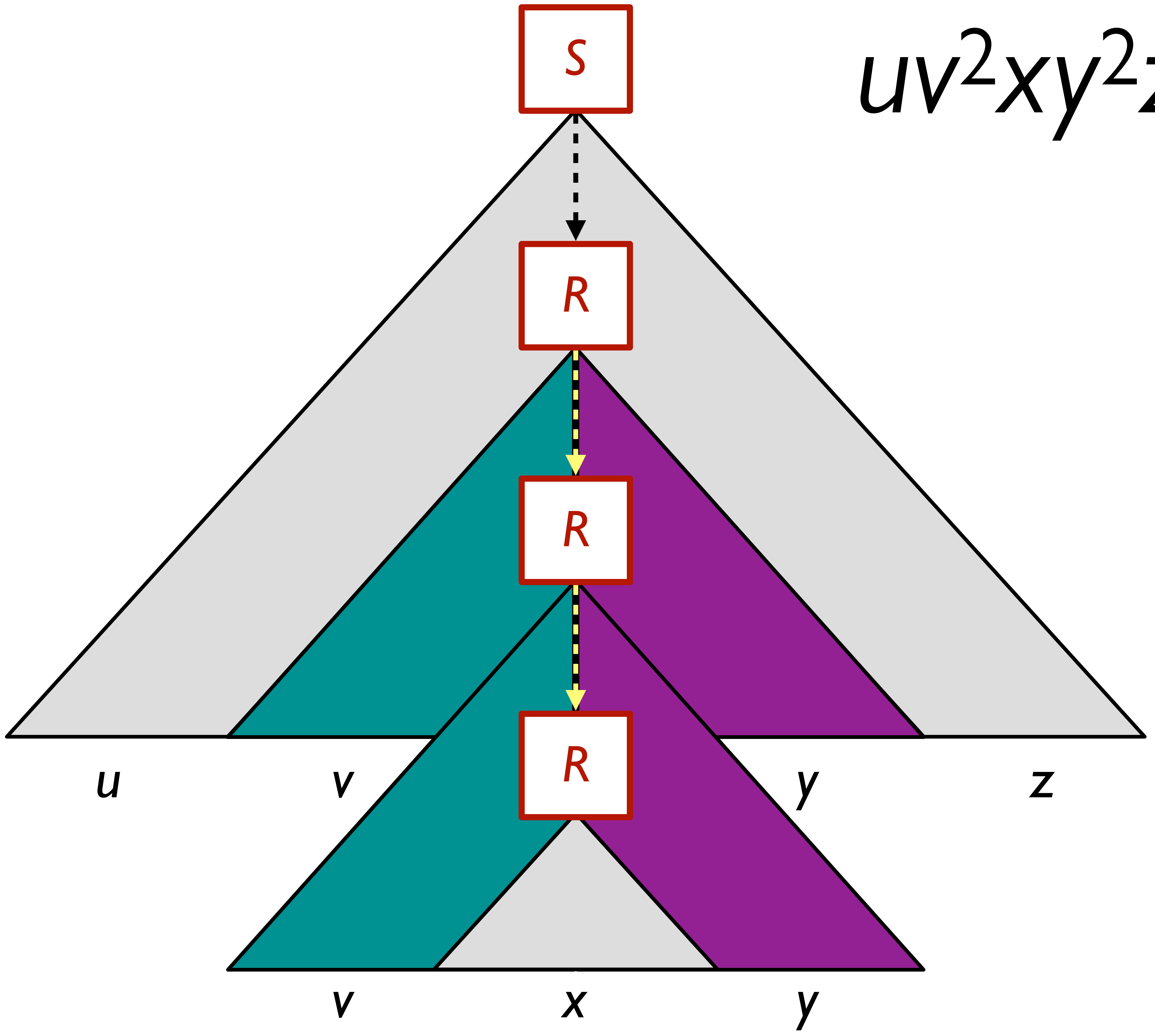
$uvxyz \in L$



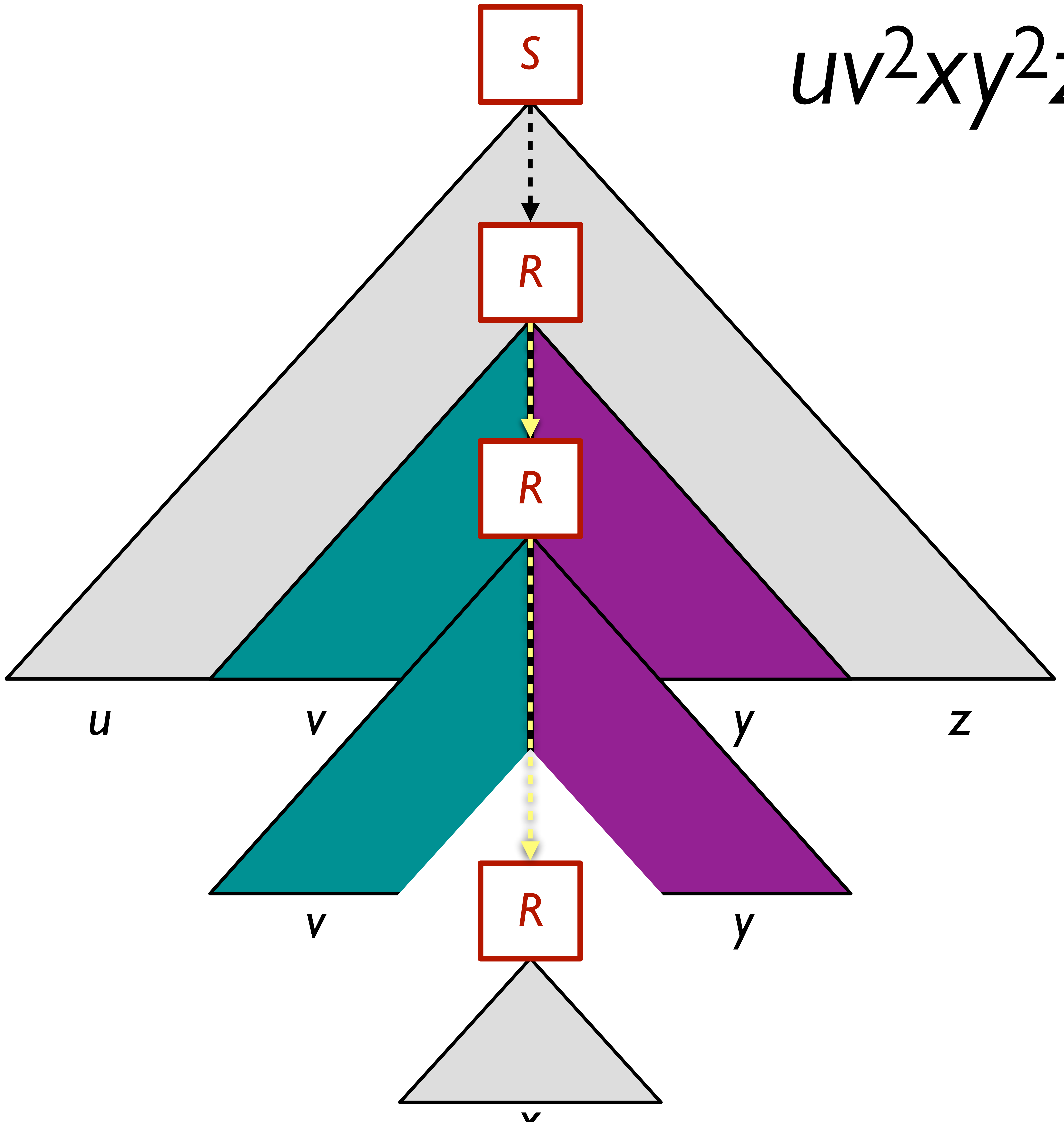
$uvxyz \in L$



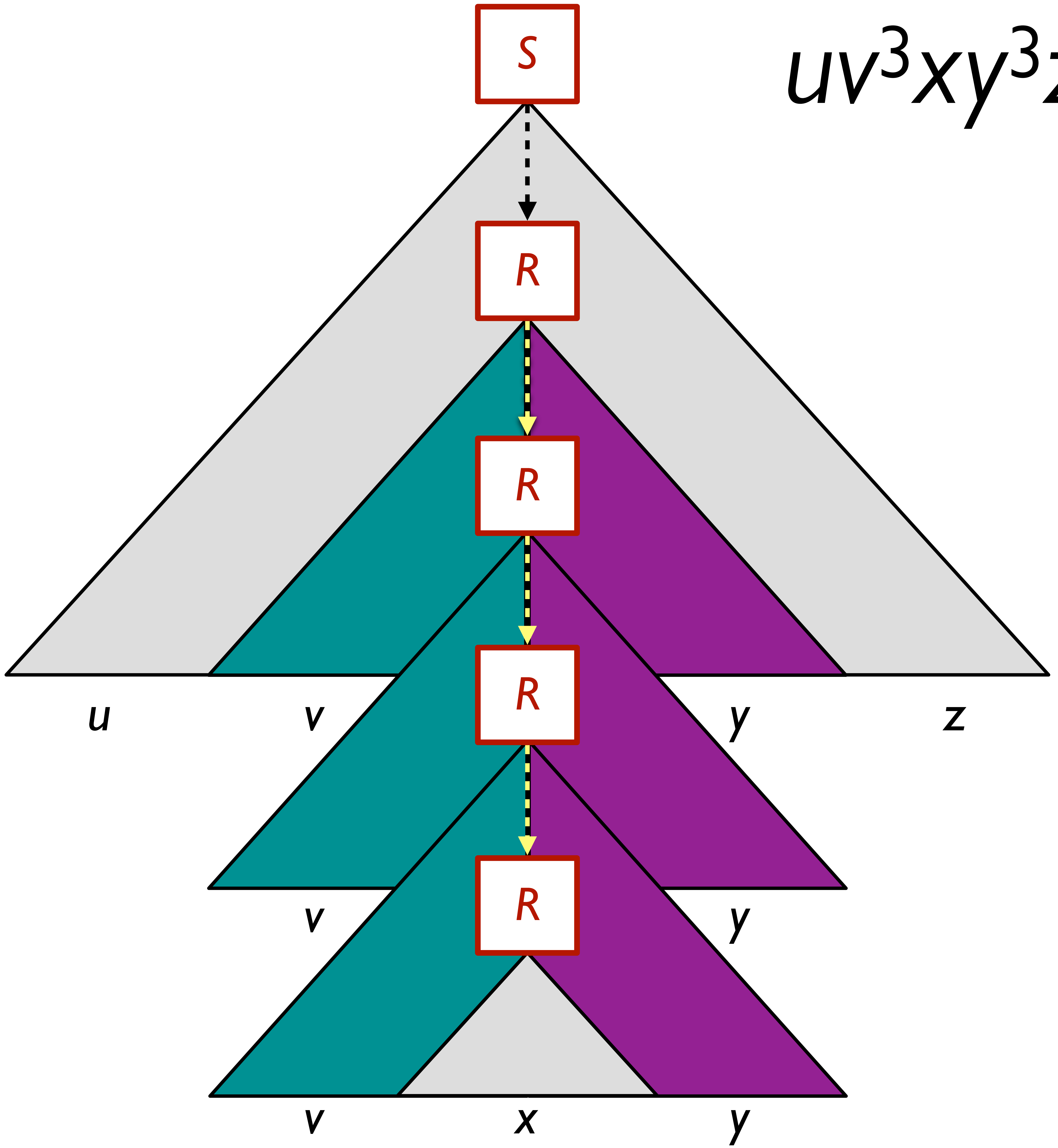
$uv^2xy^2z \in L$



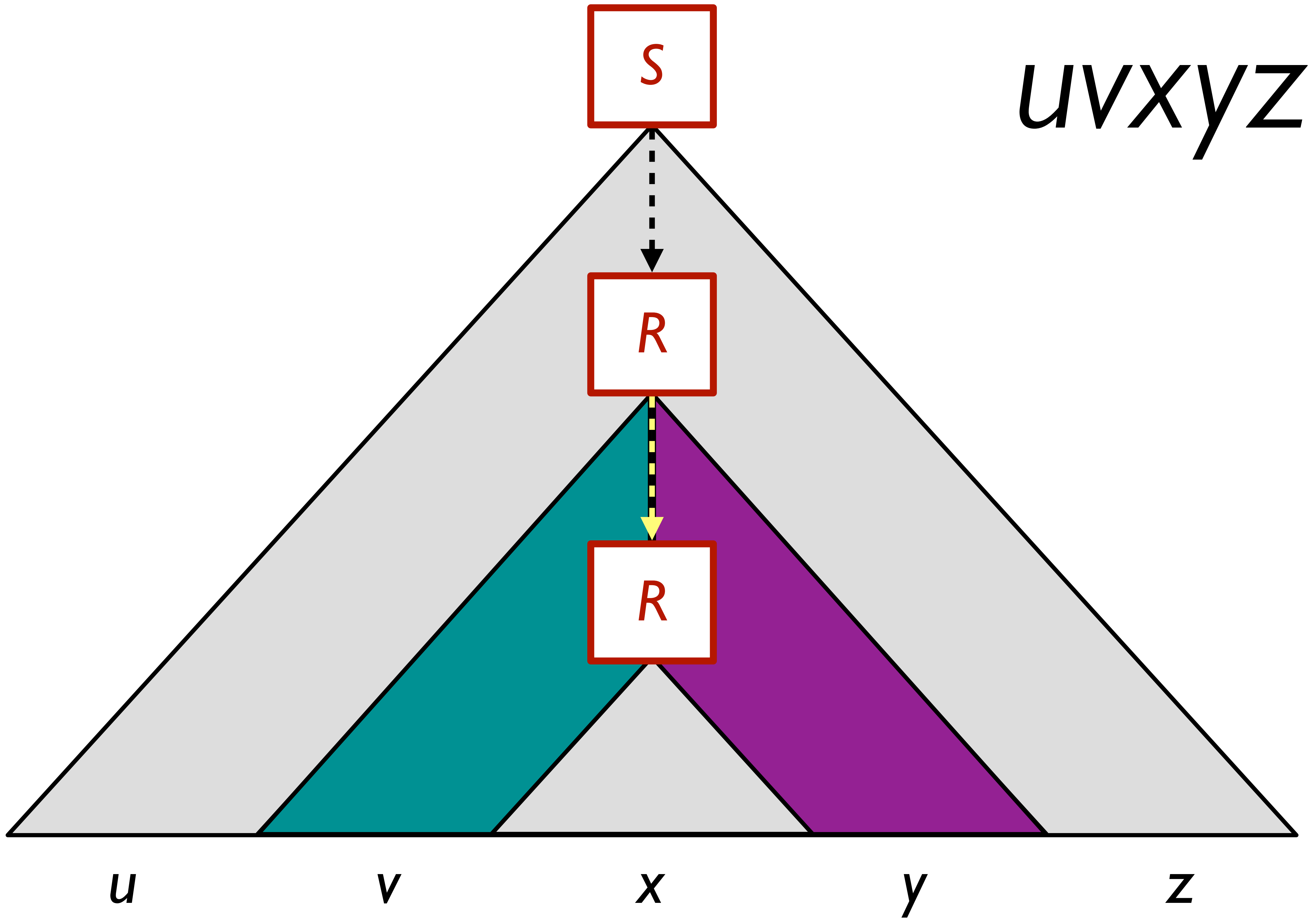
$uv^2xy^2z \in L$

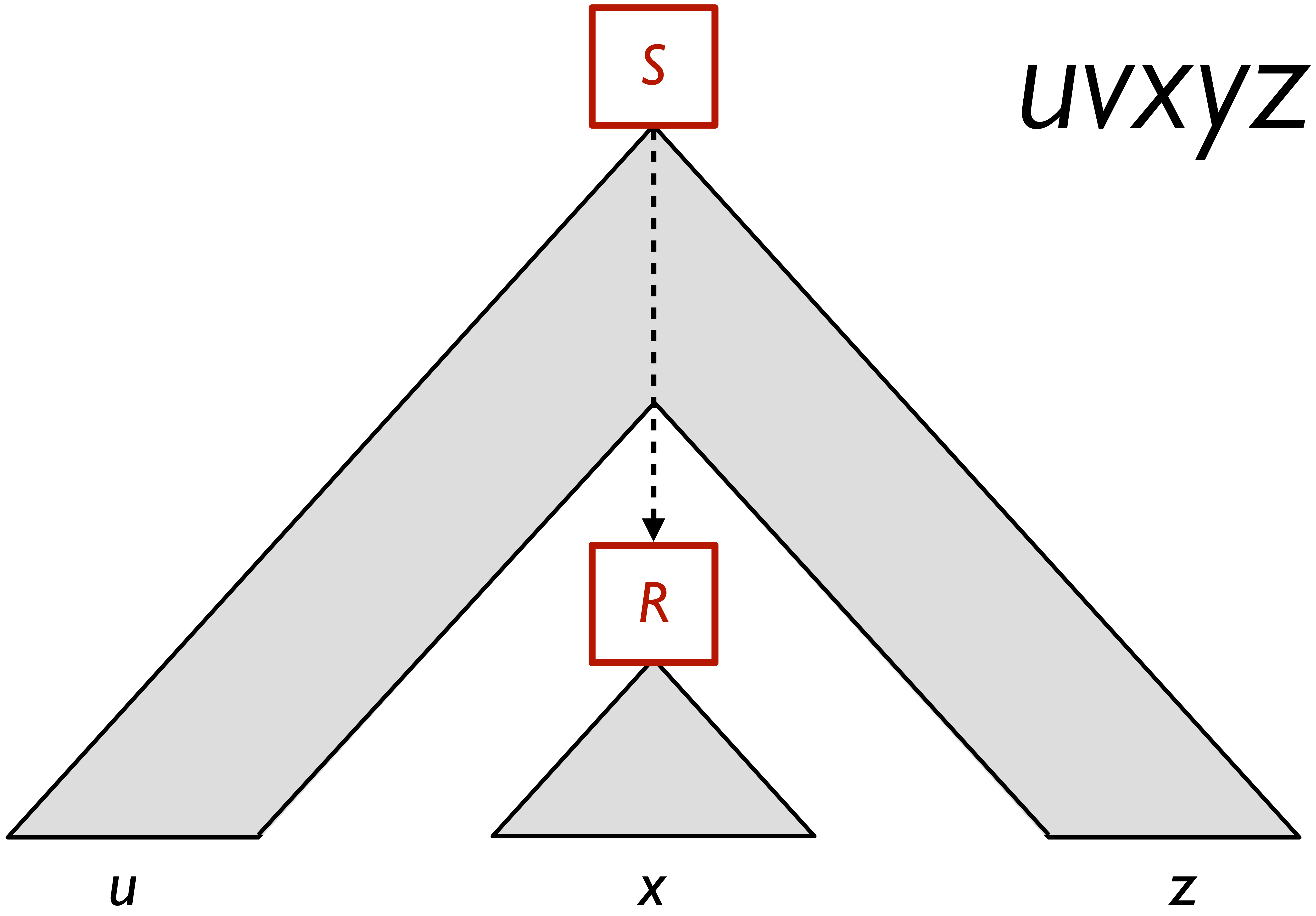


$uv^3xy^3z \in L$



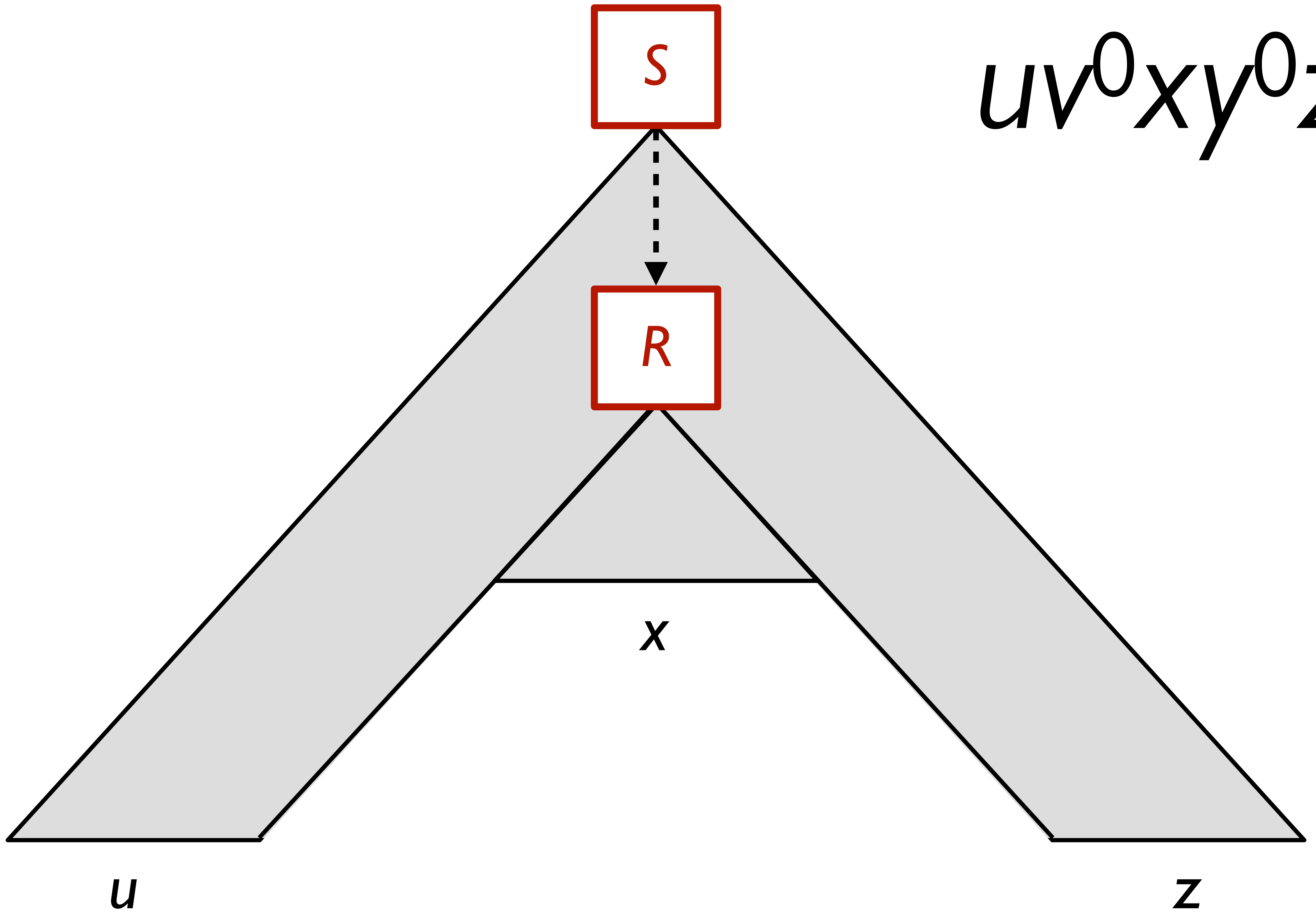
$uvxyz \in L$





$uvxyz \in L$

$uv^0xy^0z \in L$



Anna strikes again

Anna: The language $B = \{a^n b^n c^n \mid n \in \mathbb{N}_0\}$ is not context-free.

Elsa: Oh, yes it is!

Anna: Do you have a context-free grammar that generates it?

Elsa: Of course! Here it is:

$$
\begin{aligned}
A &\rightarrow aA \\
A &\rightarrow aB \\
B &\rightarrow bC \\
C &\rightarrow BC \\
C &\rightarrow c
\end{aligned}
$$

Anna: How many variables does your grammar have?

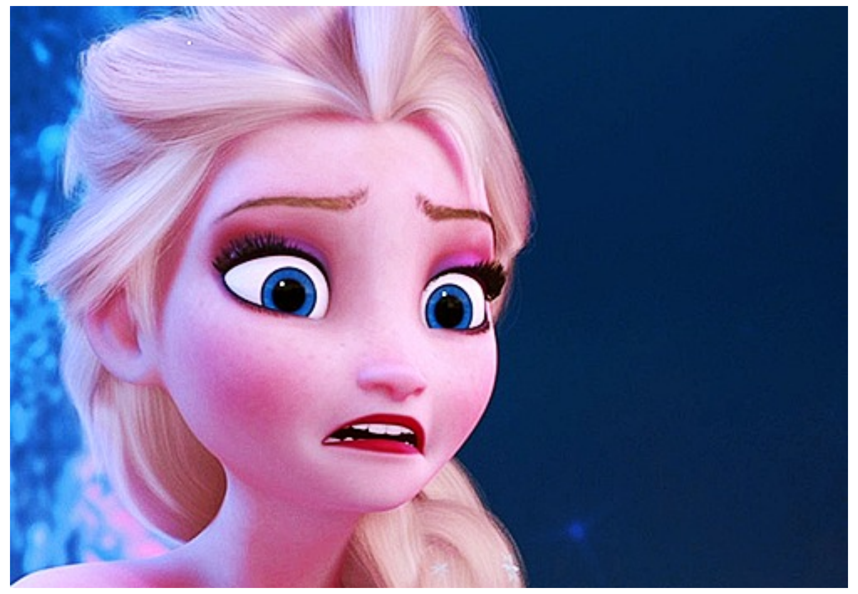
Elsa: Three.

Anna: What's the longest right-hand side for a rule?

Elsa: Two variables/terminals.

Anna: Does your grammar generate this string:

aaaaaabbbbbbcccccc?

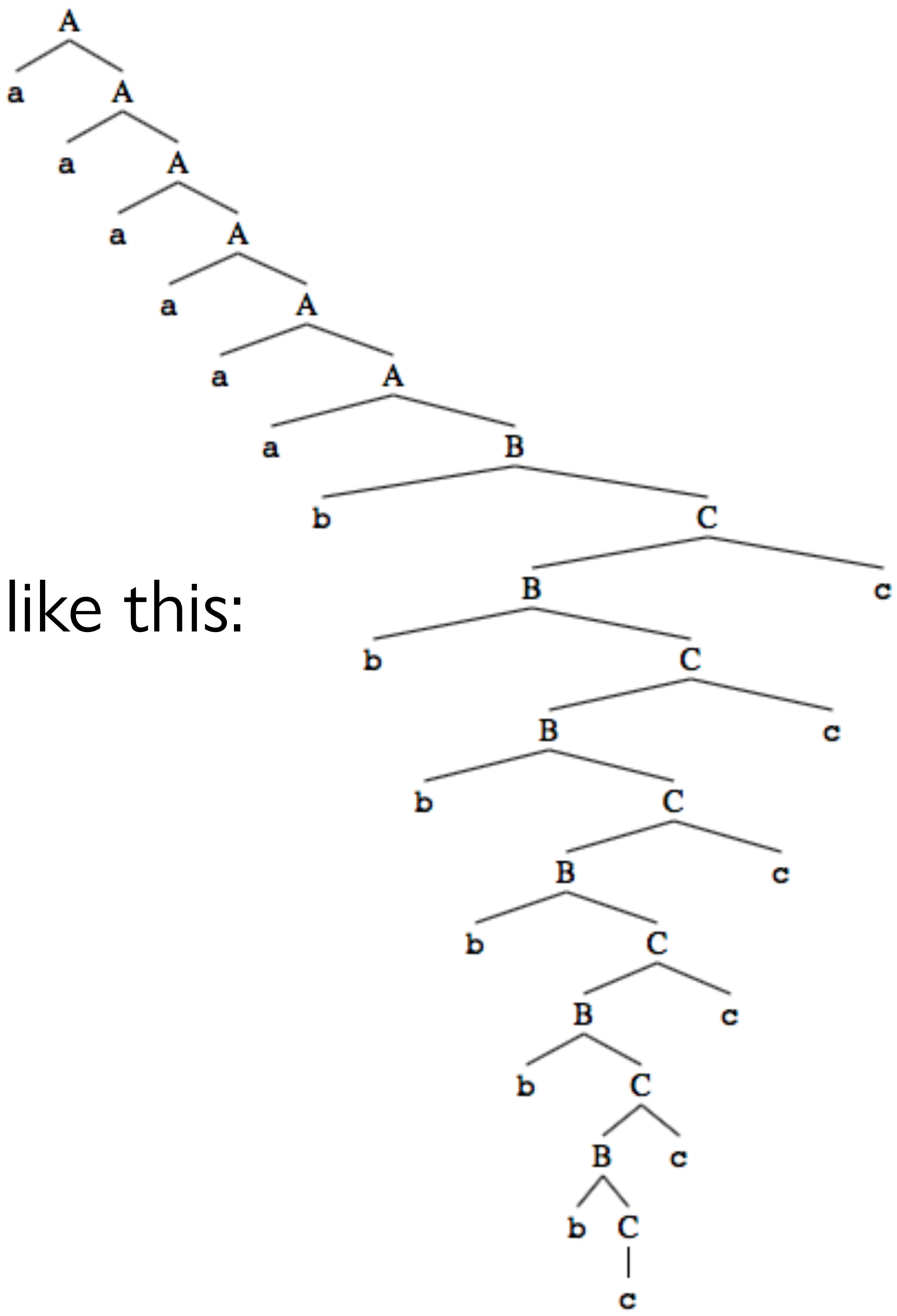
Elsa: 

Yikes. Give me a minute.

Elsa: I asked the computer to check for me.

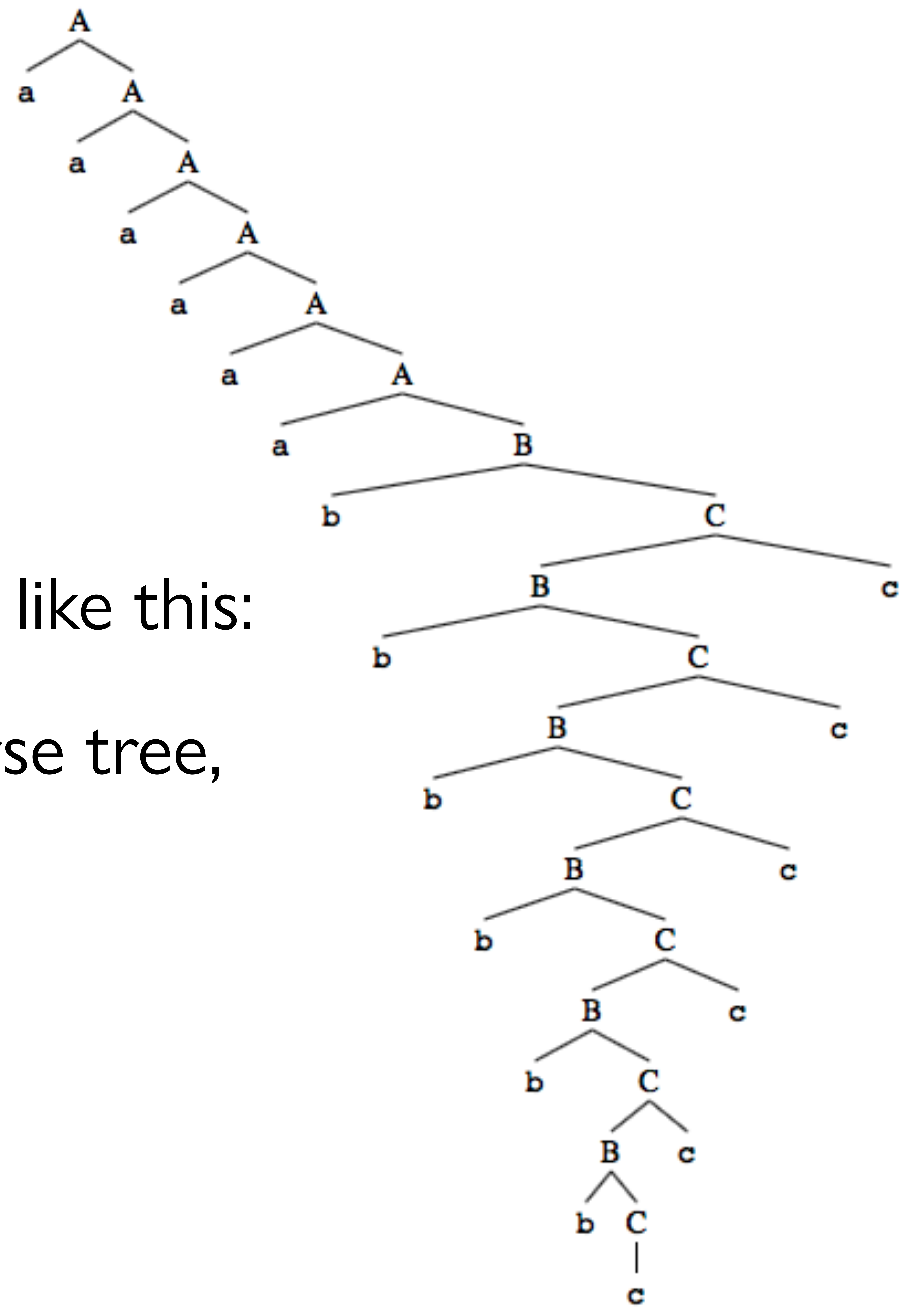
[*We'll see how to do this next class!*]

It says my grammar generates that string like this:



Elsa: I asked the computer to check for me.
[*We'll see how to do this next class!*]

It says my grammar generates that string like this:



Anna: Ok, what's the ***longest path*** in the parse tree, from the start symbol to a terminal?

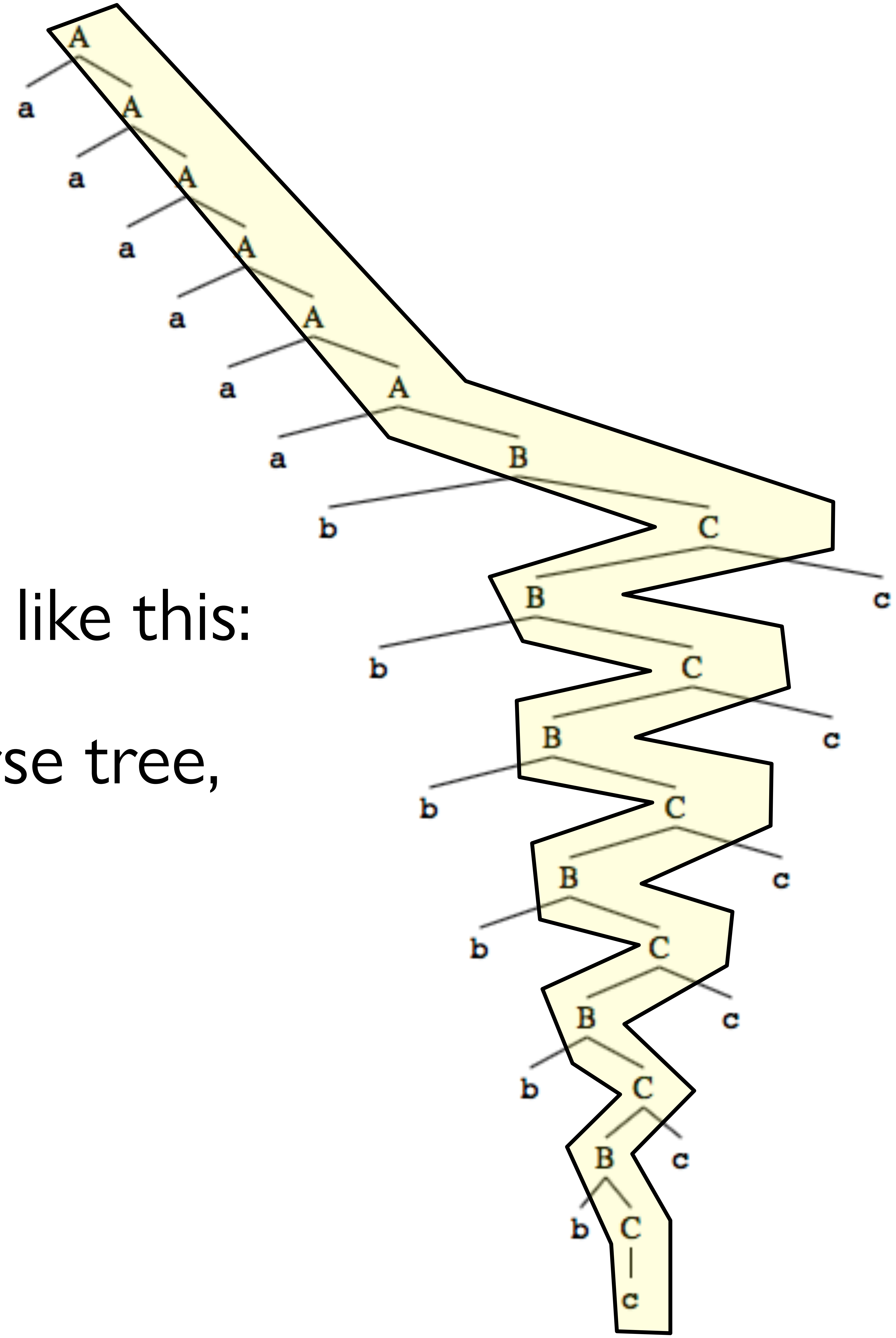
Elsa: Let me see…

Elsa: I asked the computer to check for me.
[*We'll see how to do this next class!*]

It says my grammar generates that string like this:

Anna: Ok, what's the *longest path* in the parse tree,
from the start symbol to a terminal?

Elsa: Here it is:

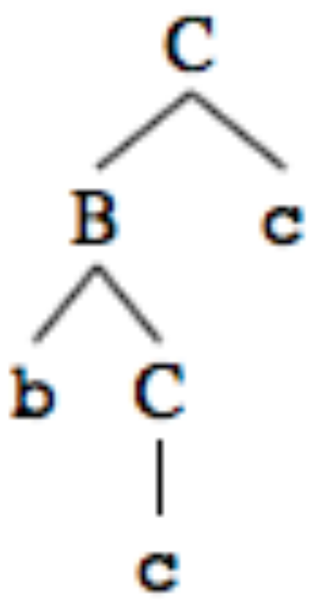


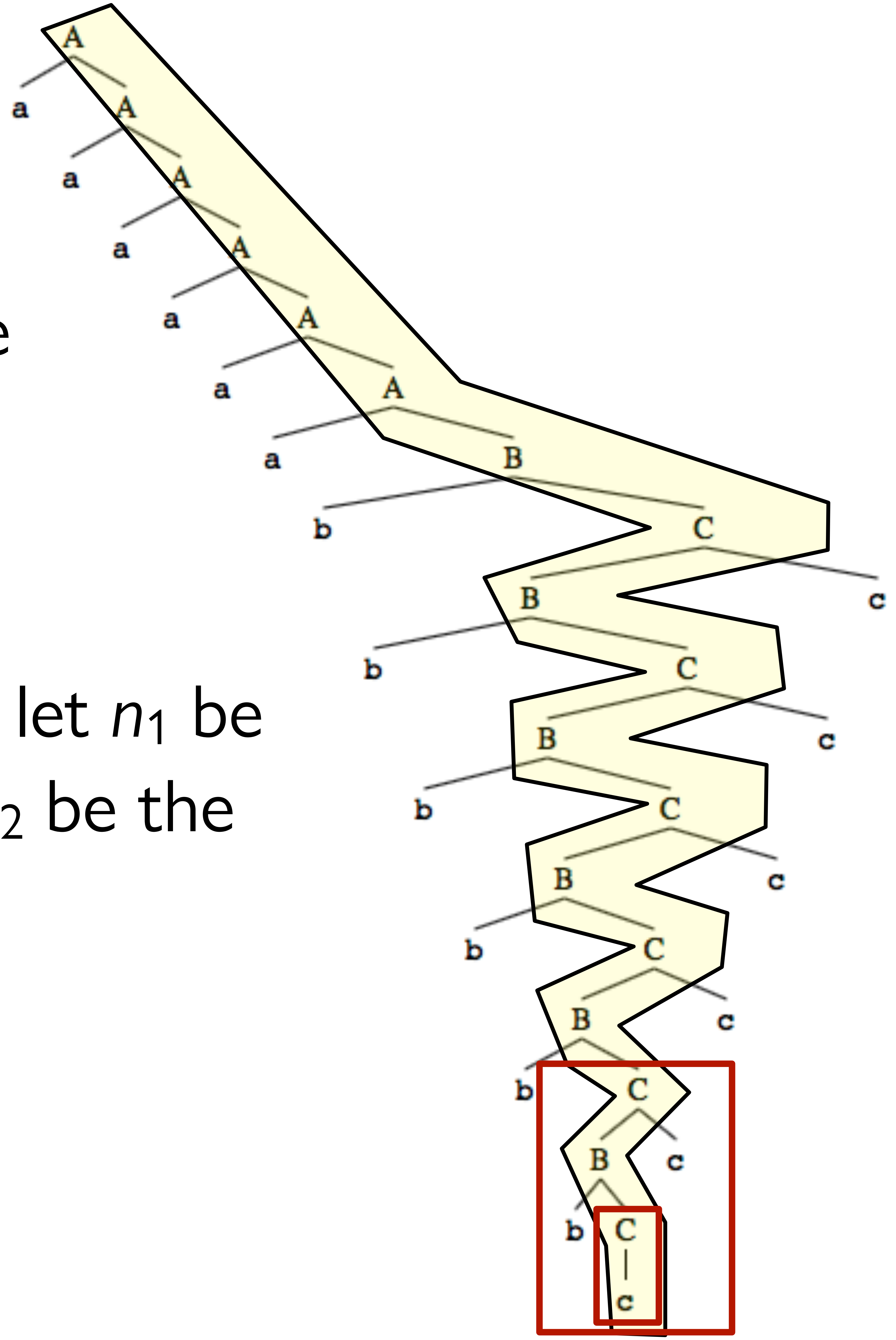
Anna: In this path, you use the same variable twice, right?

Elsa: Yes.

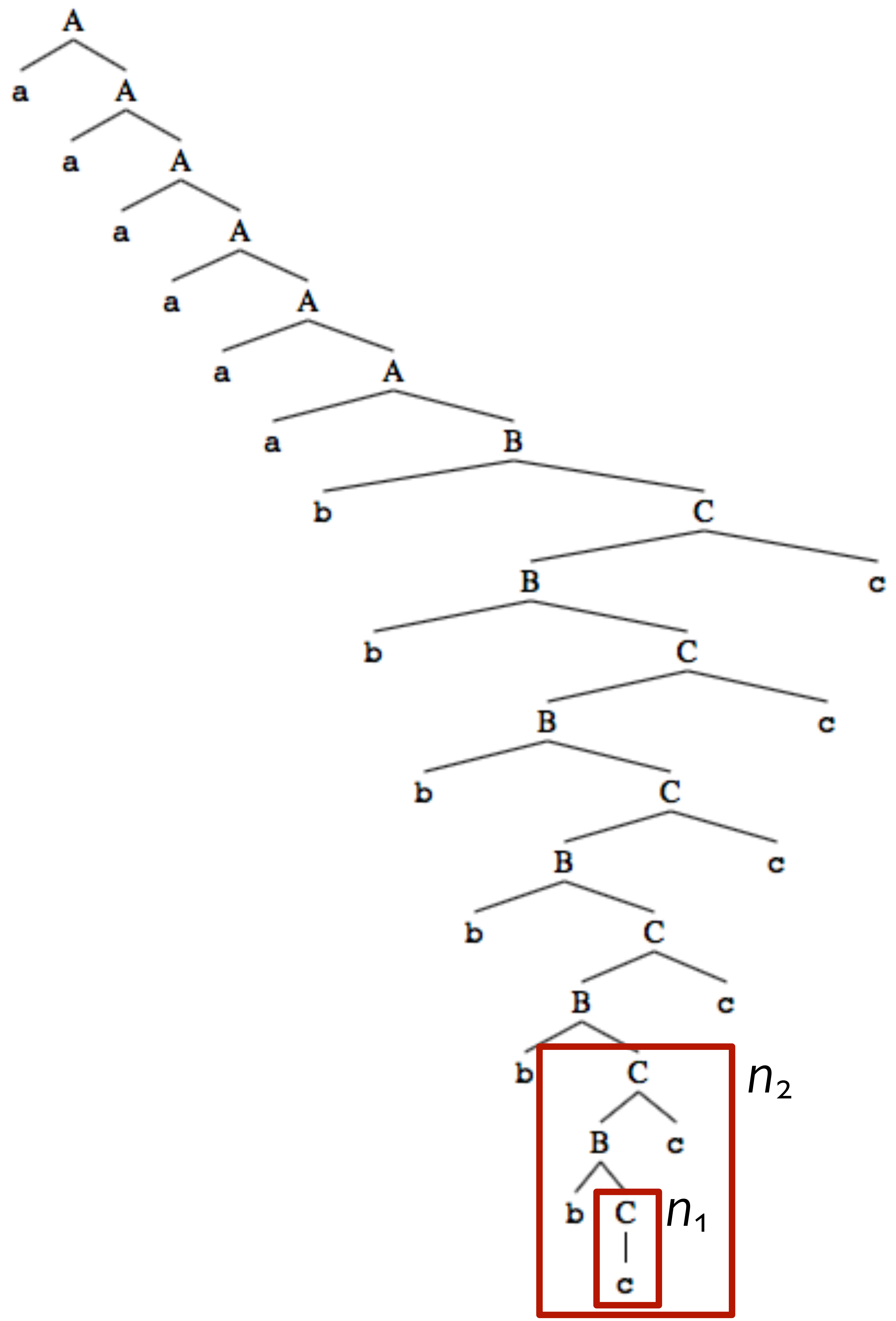
Anna: Starting from the bottom of the tree, let n_1 be the first occurrence of the variable and n_2 be the second occurrence.

Elsa: Okay, n_1 = 

and n_2 = 

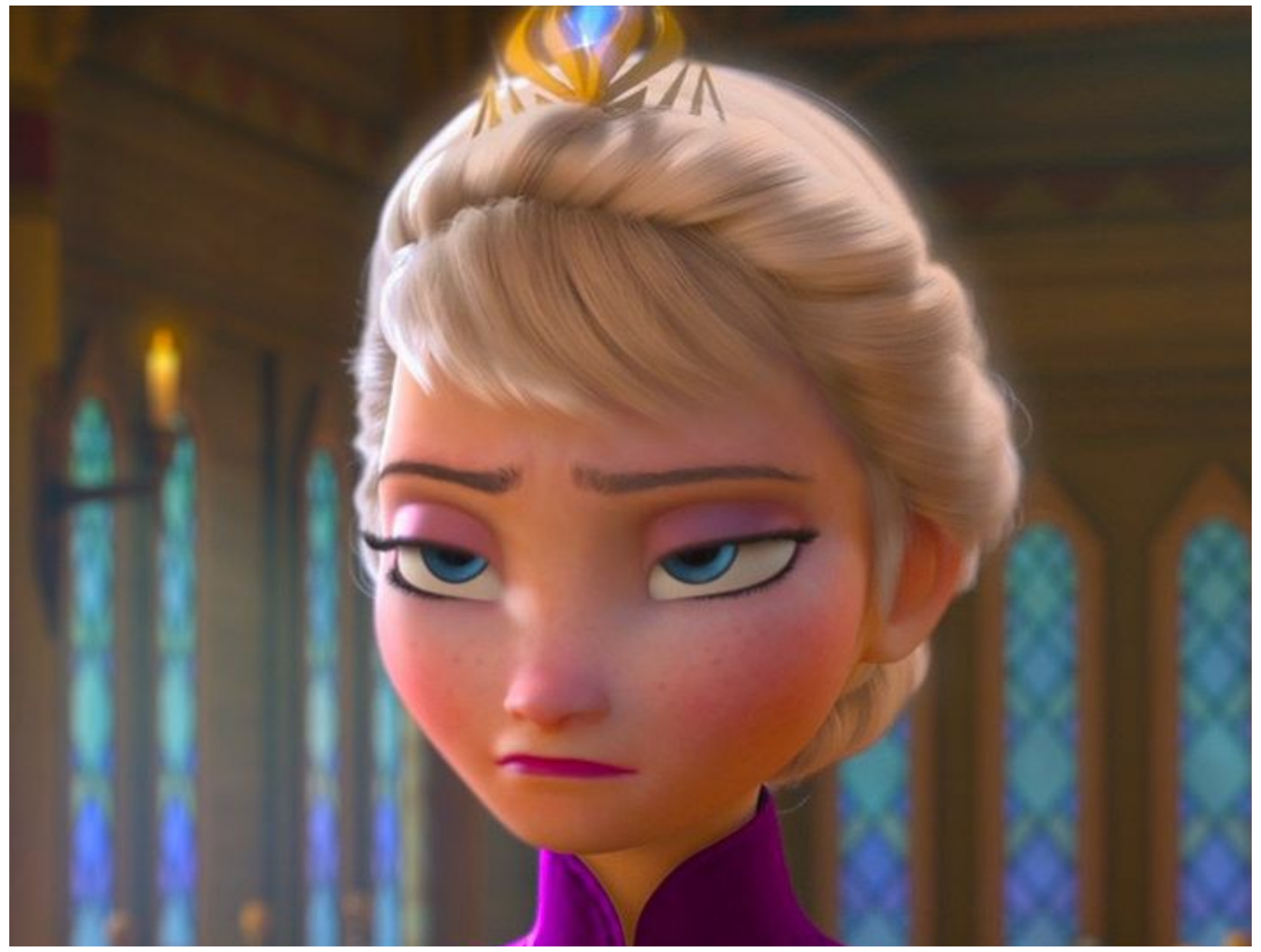


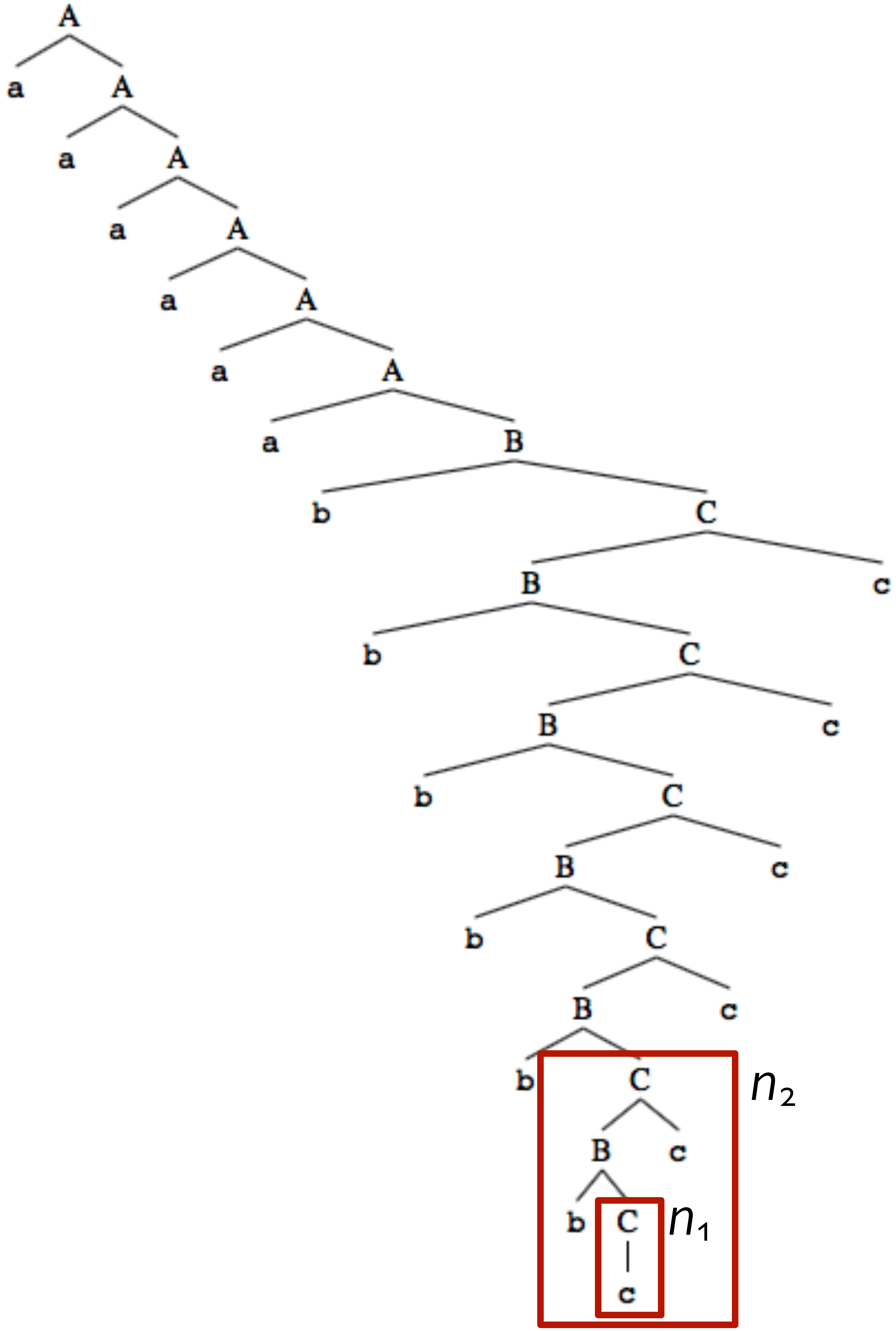
Anna: Let u, v, x, y, and z be such that $s = uvxyz$, where x is the string under n_1 and vxy is the string under n_2.



Anna: Let u, v, x, y, and z be such that $s = uvxyz$, where x is the string under n_1 and vxy is the string under n_2.

Elsa: I want to find someone else to talk to.

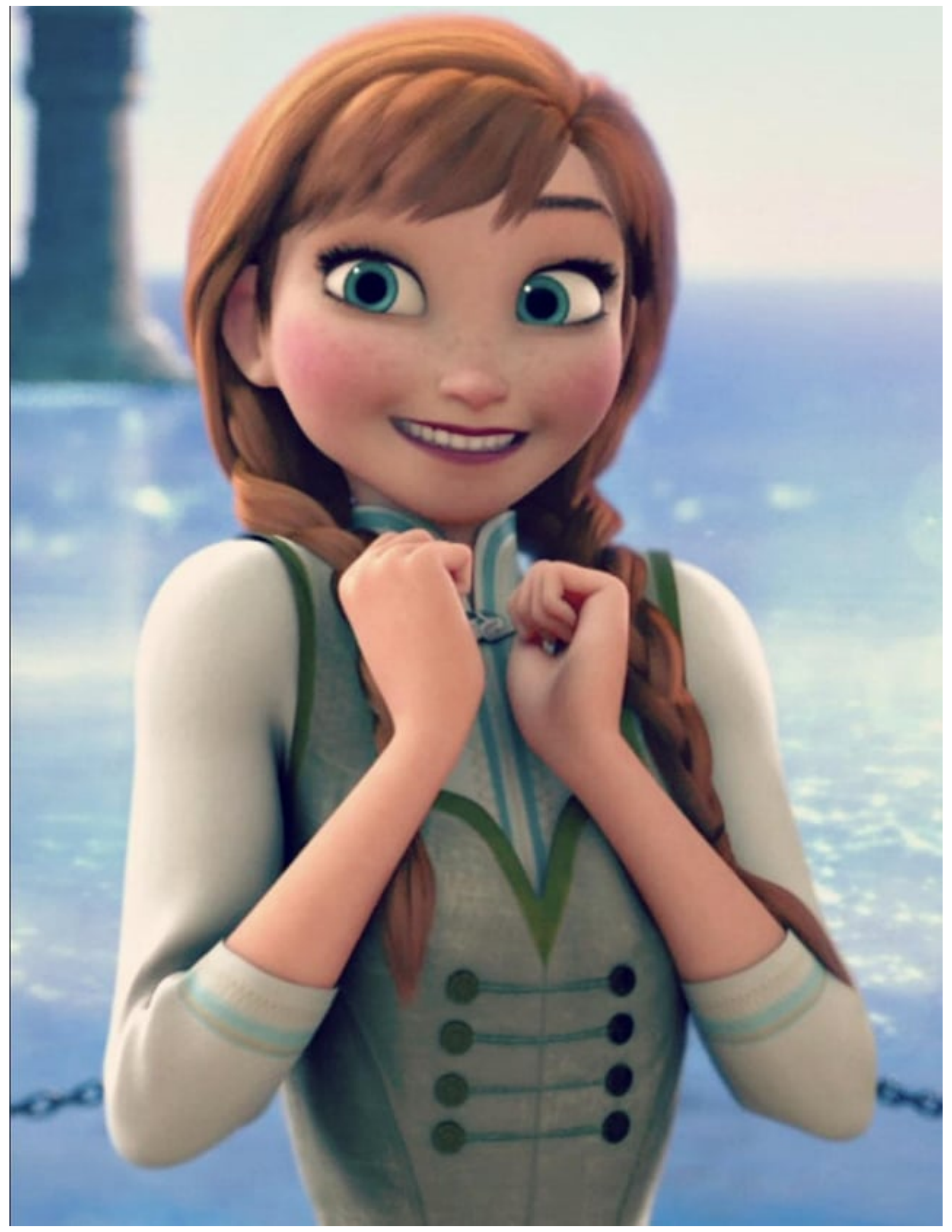


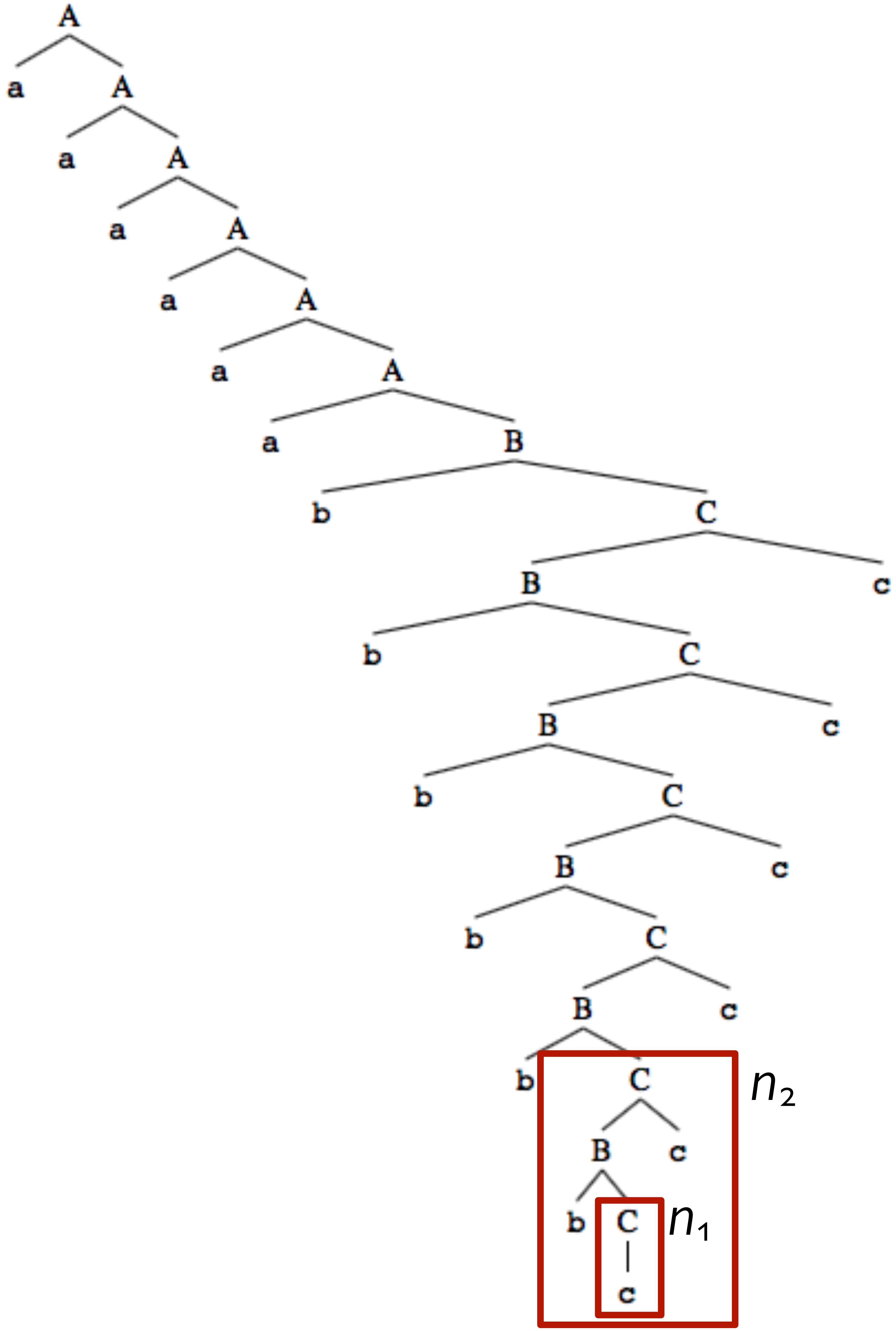


Anna: Let u, v, x, y, and z be such that $s = uvxyz$, where x is the string under n_1 and vxy is the string under n_2.

Elsa: I want to find someone else to talk to.

Anna: Come on, we're almost there.





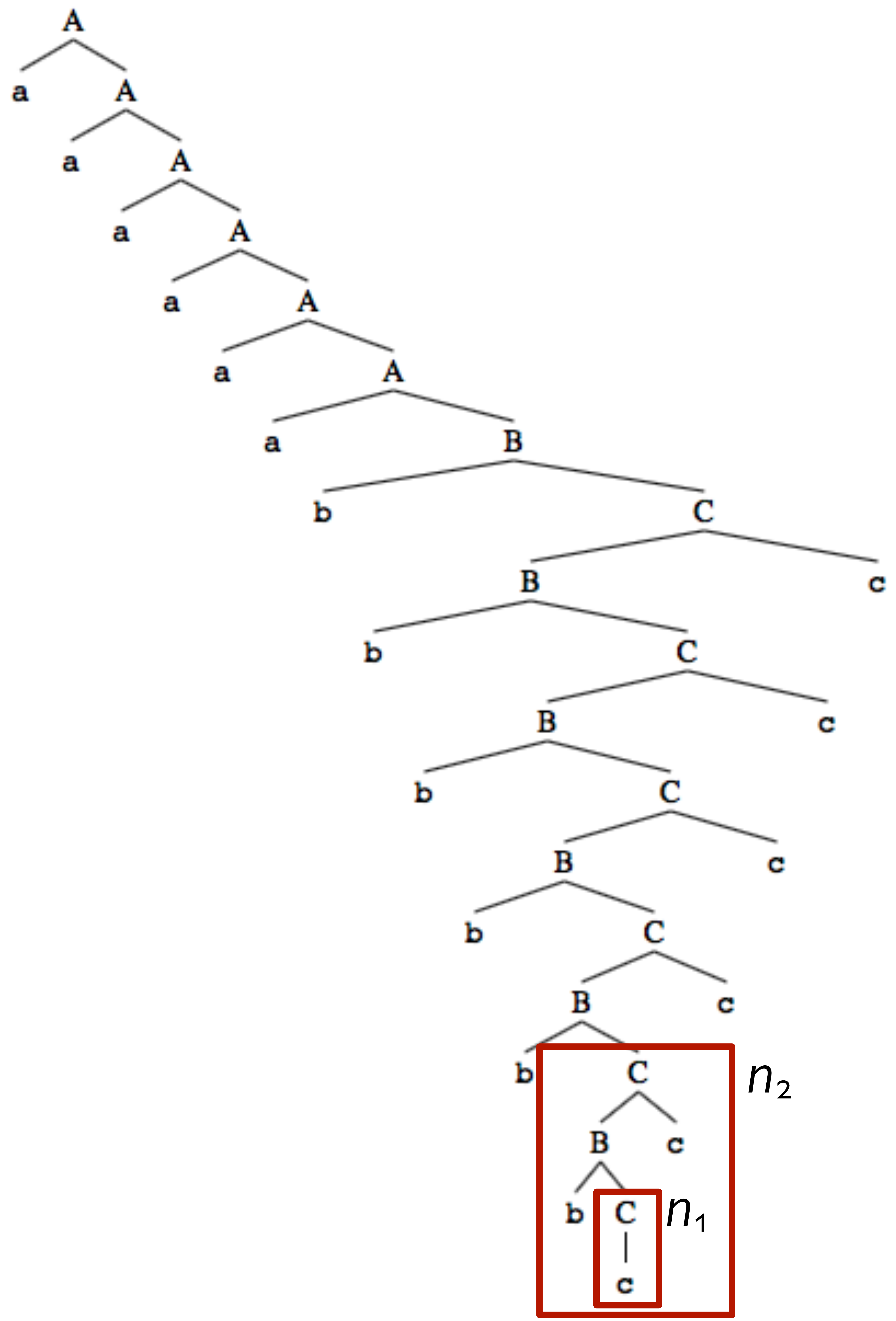
Anna: Let u, v, x, y, and z be such that $s = uvxyz$, where x is the string under n_1 and vxy is the string under n_2.

Elsa: I want to find someone else to talk to.

Anna: Come on, we're almost there.

Elsa: Okay:

u = aaaaaabbbbb
v = b
x = c
y = c
z = cccc



Anna: Does your grammar generate…

uv^2xy^2z = aaaaaabbbbbbccccccc

…this string?

Elsa: I’ll ask the computer.

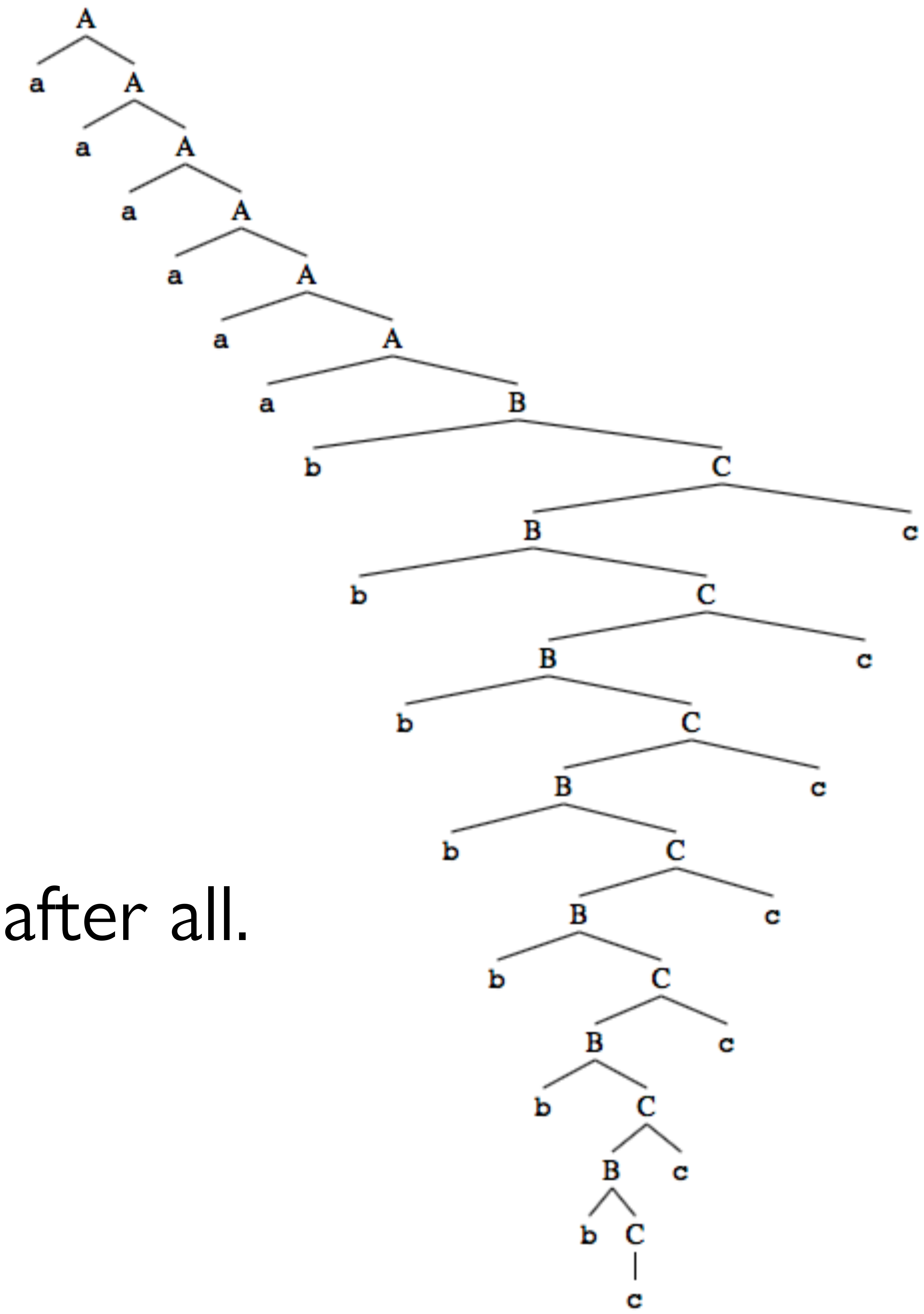
Anna: Does your grammar generate…

uv^2xy^2z = aaaaaabbbbbbcccccc

…this string?

Elsa: I'll ask the computer.

Yes. I was wrong – it's not a context-free after all. Can I go now?



The Pumping Lemma for CFLs

Bar-Hillel, Perles, and Shamir, 1961

For any context-free language L,

there exists a positive integer p such that

for any string $s \in L$ of length p or more,

there exist strings u, v, x, y, z such that

$s = uvxyz$, — *s can be broken into 5 pieces*

$|vxy| \leq p$, — *the middle 3 pieces aren't too long*

$vy \neq \varepsilon$, and — *the 2nd & 4th pieces aren't **both** empty*

$uv^ixy^iz \in L$ for all $i \in \mathbb{N}_0$ — *the 2nd & 4th pieces can be replicated 0 or more times*

The Pumping Lemma for CFLs

For any context-free language L,

there exists a positive integer p such that

for any string $s \in L$ of length p or more,

there exist strings u, v, x, y, z such that

$s = uvxyz$,

$|vxy| \leq p$,

$vy \neq \varepsilon$, and

$uv^ixy^iz \in L$ for all $i \in \mathbb{N}_0$

Note that we need to pump v and y ***together****; you can't pump just one or the other*

The Pumping Lemma for CFLs

For any context-free language L,

there exists a positive integer p such that

for any string $s \in L$ of length p or more,

there exist strings u, v, x, y, z such that

$s = uvxyz$,

$|vxy| \leq p$,

$vy \neq \varepsilon$, and

$uv^ixy^iz \in L$ for all $i \in \mathbb{N}_0$

The two strings to pump, collectively, can't be too long and must be close together.

THEOREM $L = \{a^n b^n c^n \mid n \in \mathbb{N}_0\}$ is not a CFL.

For any context-free language L,
there exists a positive integer p such that
for any string $s \in L$ of length p or more,
there exist strings u, v, x, y, z such that
$s = uvxyz$,
$|vxy| \leq p$,
$vy \neq \varepsilon$, and
$uv^ixy^iz \in L$ for all $i \in \mathbb{N}_0$

THEOREM $L = \{a^n b^n c^n \mid n \in \mathbb{N}_0\}$ is not a CFL.

PROOF By contradiction; assume L is a CFL.

For any context-free language L,
there exists a positive integer p such that
for any string $s \in L$ of length p or more,
there exist strings u, v, x, y, z such that
$s = uvxyz$,
$|vxy| \leq p$,
$vy \neq \varepsilon$, and
$uv^ixy^iz \in L$ for all $i \in \mathbb{N}_0$

THEOREM $L = \{a^n b^n c^n \mid n \in \mathbb{N}_0\}$ is not a CFL.

PROOF By contradiction; assume L is a CFL. Let p be the pumping length guaranteed by the Pumping Lemma.

For any context-free language L,
there exists a positive integer p such that
for any string $s \in L$ of length p or more,
there exist strings u, v, x, y, z such that
$s = uvxyz$,
$|vxy| \le p$,
$vy \ne \varepsilon$, and
$uv^ixy^iz \in L$ for all $i \in \mathbb{N}_0$

THEOREM $L = \{a^n b^n c^n \mid n \in \mathbb{N}_0\}$ is not a CFL.

PROOF By contradiction; assume L is a CFL. Let p be the pumping length guaranteed by the Pumping Lemma. Let $s = a^p b^p c^p$. Since $s \in L$ and $|s| = 3p \geq p$, we can write $s = uvxyz$ such that $|vxy| \leq p$, $vy \neq \varepsilon$, and for any $i \in \mathbb{N}_0$, $uv^ixy^iz \in L$.

For any context-free language L,
there exists a positive integer p such that
for any string $s \in L$ of length p or more,
there exist strings u, v, x, y, z such that
$s = uvxyz$,
$|vxy| \leq p$,
$vy \neq \varepsilon$, and
$uv^ixy^iz \in L$ for all $i \in \mathbb{N}_0$

THEOREM $L = \{a^n b^n c^n \mid n \in \mathbb{N}_0\}$ is not a CFL.

PROOF By contradiction; assume L is a CFL. Let p be the pumping length guaranteed by the Pumping Lemma. Let $s = a^p b^p c^p$. Since $s \in L$ and $|s| = 3p \geq p$, we can write $s = uvxyz$ such that $|vxy| \leq p$, $vy \neq \varepsilon$, and for any $i \in \mathbb{N}_0$, $uv^ixy^iz \in L$. We consider two cases for vxy:

Case 1:

Case 2:

For any context-free language L,
there exists a positive integer p such that
for any string $s \in L$ of length p or more,
there exist strings u, v, x, y, z such that
$s = uvxyz$,
$|vxy| \leq p$,
$vy \neq \varepsilon$, and
$uv^ixy^iz \in L$ for all $i \in \mathbb{N}_0$

THEOREM $L = \{a^n b^n c^n \mid n \in \mathbb{N}_0\}$ is not a CFL.

PROOF By contradiction; assume L is a CFL. Let p be the pumping length guaranteed by the Pumping Lemma. Let $s = a^p b^p c^p$. Since $s \in L$ and $|s| = 3p \geq p$, we can write $s = uvxyz$ such that $|vxy| \leq p$, $vy \neq \varepsilon$, and for any $i \in \mathbb{N}_0$, $uv^i xy^i z \in L$. We consider two cases for vxy:

Case 1:

Case 2:

For any context-free language L,
there exists a positive integer p such that
for any string $s \in L$ of length p or more,
there exist strings u, v, x, y, z such that
$s = uvxyz$,
$|vxy| \leq p$,
$vy \neq \varepsilon$, and
$uv^i xy^i z \in L$ for all $i \in \mathbb{N}_0$

*Proofs using the Pumping Lemma for CFLs tend to be harder than those for regular languages because there's no restriction on **where** in the string the portion that can be pumped can be.*

THEOREM $L = \{a^n b^n c^n \mid n \in \mathbb{N}_0\}$ is not a CFL.

PROOF By contradiction; assume L is a CFL. Let p be the pumping length guaranteed by the Pumping Lemma. Let $s = a^p b^p c^p$. Since $s \in L$ and $|s| = 3p \geq p$, we can write $s = uvxyz$ such that $|vxy| \leq p$, $vy \neq \varepsilon$, and for any $i \in \mathbb{N}_0$, $uv^ixy^iz \in L$. We consider two cases for vxy:

Case 1: vxy is completely contained in a^p, b^p, or c^p.

Case 2:

For any context-free language L,
there exists a positive integer p such that
for any string $s \in L$ of length p or more,
there exist strings u, v, x, y, z such that
$s = uvxyz$,
$|vxy| \leq p$,
$vy \neq \varepsilon$, and
$uv^ixy^iz \in L$ for all $i \in \mathbb{N}_0$

For any context-free language L,
there exists a positive integer p such that
for any string $s \in L$ of length p or more,
there exist strings u, v, x, y, z such that
$s = uvxyz$,
$|vxy| \leq p$,
$vy \neq \varepsilon$, and
$uv^ixy^iz \in L$ for all $i \in \mathbb{N}_0$

THEOREM $L = \{a^n b^n c^n \mid n \in \mathbb{N}_0\}$ is not a CFL.

PROOF By contradiction; assume L is a CFL. Let p be the pumping length guaranteed by the Pumping Lemma. Let $s = a^p b^p c^p$. Since $s \in L$ and $|s| = 3p \geq p$, we can write $s = uvxyz$ such that $|vxy| \leq p$, $vy \neq \varepsilon$, and for any $i \in \mathbb{N}_0$, $uv^ixy^iz \in L$. We consider two cases for vxy:

Case 1: vxy is completely contained in a^p, b^p, or c^p. In that case, the string $uv^2xy^2z \notin L$ because this string has more copies of a or b or c than the other two symbols.

Case 2:

For any context-free language L,
there exists a positive integer p such that
for any string $s \in L$ of length p or more,
there exist strings u, v, x, y, z such that
$s = uvxyz$,
$|vxy| \leq p$,
$vy \neq \varepsilon$, and
$uv^ixy^iz \in L$ for all $i \in \mathbb{N}_0$

THEOREM $L = \{a^nb^nc^n \mid n \in \mathbb{N}_0\}$ is not a CFL.

PROOF By contradiction; assume L is a CFL. Let p be the pumping length guaranteed by the Pumping Lemma. Let $s = a^pb^pc^p$. Since $s \in L$ and $|s| = 3p \geq p$, we can write $s = uvxyz$ such that $|vxy| \leq p$, $vy \neq \varepsilon$, and for any $i \in \mathbb{N}_0$, $uv^ixy^iz \in L$. We consider two cases for vxy:

Case 1: vxy is completely contained in a^p, b^p, or c^p. In that case, the string $uv^2xy^2z \notin L$ because this string has more copies of a or b or c than the other two symbols.

Case 2: vxy either consists of as and bs or of bs and cs. (It cannot consist of all three symbols because $|vxy| \leq p$.)

THEOREM $L = \{a^n b^n c^n \mid n \in \mathbb{N}_0\}$ is not a CFL.

PROOF By contradiction; assume L is a CFL. Let p be the pumping length guaranteed by the Pumping Lemma. Let $s = a^p b^p c^p$. Since $s \in L$ and $|s| = 3p \geq p$, we can write $s = uvxyz$ such that $|vxy| \leq p$, $vy \neq \varepsilon$, and for any $i \in \mathbb{N}_0$, $uv^ixy^iz \in L$. We consider two cases for vxy:

Case 1: vxy is completely contained in a^p, b^p, or c^p. In that case, the string $uv^2xy^2z \notin L$ because this string has more copies of a or b or c than the other two symbols.

Case 2: vxy either consists of as and bs or of bs and cs. (It cannot consist of all three symbols because $|vxy| \leq p$.)

For any context-free language L,
there exists a positive integer p such that
for any string $s \in L$ of length p or more,
there exist strings u, v, x, y, z such that
$s = uvxyz$,
$|vxy| \leq p$,
$vy \neq \varepsilon$, and
$uv^ixy^iz \in L$ for all $i \in \mathbb{N}_0$

Note that we chose s so that vxy can't span all three groups of symbols, making it impossible to pump all three groups at once!

THEOREM $L = \{a^n b^n c^n \mid n \in \mathbb{N}_0\}$ is not a CFL.

PROOF By contradiction; assume L is a CFL. Let p be the pumping length guaranteed by the Pumping Lemma. Let $s = a^p b^p c^p$. Since $s \in L$ and $|s| = 3p \geq p$, we can write $s = uvxyz$ such that $|vxy| \leq p$, $vy \neq \varepsilon$, and for any $i \in \mathbb{N}_0$, $uv^ixy^iz \in L$. We consider two cases for vxy:

Case 1: vxy is completely contained in a^p, b^p, or c^p. In that case, the string $uv^2xy^2z \notin L$ because this string has more copies of a or b or c than the other two symbols.

Case 2: vxy either consists of as and bs or of bs and cs. (It cannot consist of all three symbols because $|vxy| \leq p$.) Then if vxy has no cs in it, $uv^2xy^2z \notin L$ since it contains more as or bs than cs.

For any context-free language L,
there exists a positive integer p such that
for any string $s \in L$ of length p or more,
there exist strings u, v, x, y, z such that
$s = uvxyz$,
$|vxy| \leq p$,
$vy \neq \varepsilon$, and
$uv^ixy^iz \in L$ for all $i \in \mathbb{N}_0$

For any context-free language L,
there exists a positive integer p such that
for any string $s \in L$ of length p or more,
there exist strings u, v, x, y, z such that
$s = uvxyz$,
$|vxy| \leq p$,
$vy \neq \varepsilon$, and
$uv^ixy^iz \in L$ for all $i \in \mathbb{N}_0$

THEOREM $L = \{a^nb^nc^n \mid n \in \mathbb{N}_0\}$ is not a CFL.

PROOF By contradiction; assume L is a CFL. Let p be the pumping length guaranteed by the Pumping Lemma. Let $s = a^pb^pc^p$. Since $s \in L$ and $|s| = 3p \geq p$, we can write $s = uvxyz$ such that $|vxy| \leq p$, $vy \neq \varepsilon$, and for any $i \in \mathbb{N}_0$, $uv^ixy^iz \in L$. We consider two cases for vxy:

Case 1: vxy is completely contained in a^p, b^p, or c^p. In that case, the string $uv^2xy^2z \notin L$ because this string has more copies of a or b or c than the other two symbols.

Case 2: vxy either consists of as and bs or of bs and cs. (It cannot consist of all three symbols because $|vxy| \leq p$.) Then if vxy has no cs in it, $uv^2xy^2z \notin L$ since it contains more as or bs than cs. Similarly, if vxy has no as in it, $uv^2xy^2z \notin L$ because it contains more bs or cs than as.

For any context-free language L,
there exists a positive integer p such that
for any string $s \in L$ of length p or more,
there exist strings u, v, x, y, z such that
$s = uvxyz$,
$|vxy| \le p$,
$vy \ne \varepsilon$, and
$uv^ixy^iz \in L$ for all $i \in \mathbb{N}_0$

THEOREM $L = \{a^nb^nc^n \mid n \in \mathbb{N}_0\}$ is not a CFL.

PROOF By contradiction; assume L is a CFL. Let p be the pumping length guaranteed by the Pumping Lemma. Let $s = a^pb^pc^p$. Since $s \in L$ and $|s| = 3p \ge p$, we can write $s = uvxyz$ such that $|vxy| \le p$, $vy \ne \varepsilon$, and for any $i \in \mathbb{N}_0$, $uv^ixy^iz \in L$. We consider two cases for vxy:

Case 1: vxy is completely contained in a^p, b^p, or c^p. In that case, the string $uv^2xy^2z \notin L$ because this string has more copies of a or b or c than the other two symbols.

Case 2: vxy either consists of as and bs or of bs and cs. (It cannot consist of all three symbols because $|vxy| \le p$.) Then if vxy has no cs in it, $uv^2xy^2z \notin L$ since it contains more as or bs than cs. Similarly, if vxy has no as in it, $uv^2xy^2z \notin L$ because it contains more bs or cs than as.

In either case, we contradict the Pumping Lemma. Thus our assumption must have been wrong, and L is not a CFL. ∎

Regular languages cannot count arbitrarily high:

$\{a^n b^n \mid n \in \mathbb{N}_0\}$ is not regular.

While CFLs cannot count arbitrarily high *twice*:

$\{a^n b^n c^n \mid n \in \mathbb{N}_0\}$ is not context-free.

Keep the following in mind when using the Pumping Lemma for CFLs, where $s = uvxyz$:

Both v and y must be pumped ***at the same time***.

v and y ***don't*** need to be contiguous in the string – a non-empty x can be in between.

One of v or y may be empty.

vxy can be ***anywhere*** in the string – not just at the start or in the middle. (The lengths of u and z can be anything!)

I *strongly* suggest recommend reading through

the presentation in Sipser and

the examples in the handout on the Pumping Lemma for CFLs

to get a better sense for how these proofs work.

Proving a language is non-context-free

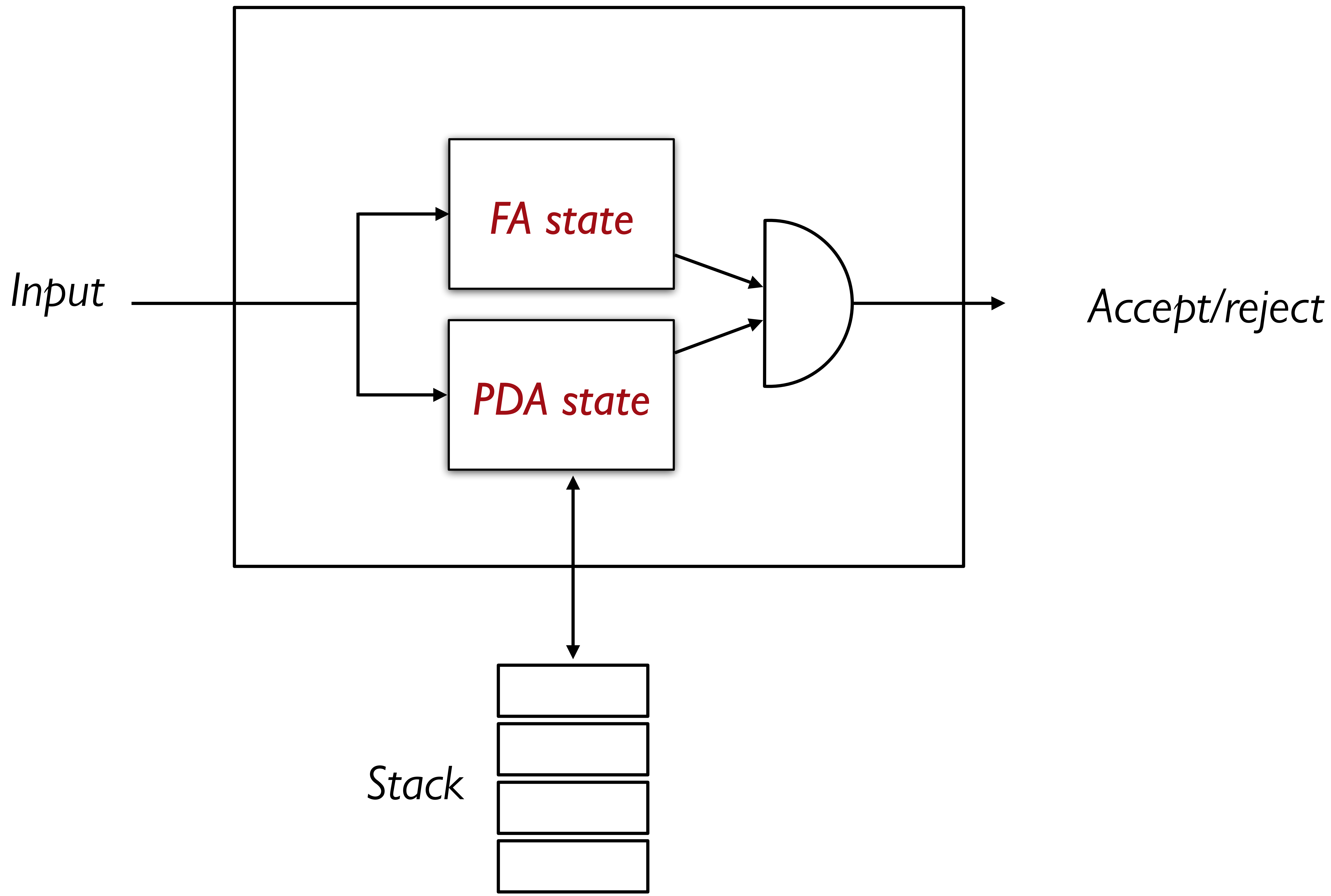
The Pumping Lemma gives a property that all context-free languages satisfy; we show that a language *doesn't* satisfy it.

If the language is complicated, we can use closure properties to find a simpler language that's either known to be non-context-free or is easier to do a Pumping Lemma proof for.

One more helpful closure property

THEOREM Let $L \subseteq \Sigma^*$ be a context-free language and let $R \subseteq \Sigma^*$ be a regular language. Then $L \cap R$ is a context-free language.

PROOF IDEA We can simulate running the corresponding PDA and DFA in parallel, similar to how we simulated running two DFAs in parallel.



Input
FA state
PDA state
Accept/reject
Stack

Closure of CFLs under intersection with a regular language is very helpful when we want to prove a language is non-context-free.

It lets us "filter" a language to contain only strings of a certain form.

Let $L = \{w \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}^* \mid n_\mathtt{a}(w) = n_\mathtt{b}(w) = n_\mathtt{c}(w)\}$.

We can prove L is non-context-free without doing a new Pumping Lemma proof.

$L \cap \mathrm{L}(\mathbf{a^*b^*c^*}) = \{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n \mid n \in \mathbb{N}_0\}$, which we already proved is non-context-free!

Non-closure properties of CFLs

Intersection

THEOREM There exist context-free languages $L_1, L_2 \in \Sigma^*$ such that $L_1 \cap L_2$ is not a context-free language.

Intersection

Let $L_1 = \{a^n b^n c^m \mid n, m \in \mathbb{N}_0\}$.

Let $L_2 = \{a^m b^n c^n \mid n, m \in \mathbb{N}_0\}$.

L_1 and L_2 are context-free.

We can prove this by designing a CFG – or we can just observe that each of these languages is the concatenation of two CFLs.

However, $L_1 \cap L_2 = \{a^n b^n c^n \mid n \in \mathbb{N}_0\}$, which is not a CFL.

This proves (by contradiction) that context-free languages are not closed under intersection.

Intersection: Caveat

This is a common point of confusion!

Let $L_1 = \{a^n b^n c^n \mid n \in \mathbb{N}_0\}$.

Let $L_2 = \{a^n b^n d^n \mid n \in \mathbb{N}_0\}$.

Then $L_1 \cap L_2 \neq \{a^n b^n \mid n \in \mathbb{N}_0\}$!

$L_1 \cap L_2$ consists of strings that are in *both* L_1 and L_2.

Every non-empty string in L_1 contains a c.

Every non-empty string in L_2 contains a d.

Thus, their intersection only includes the empty string:

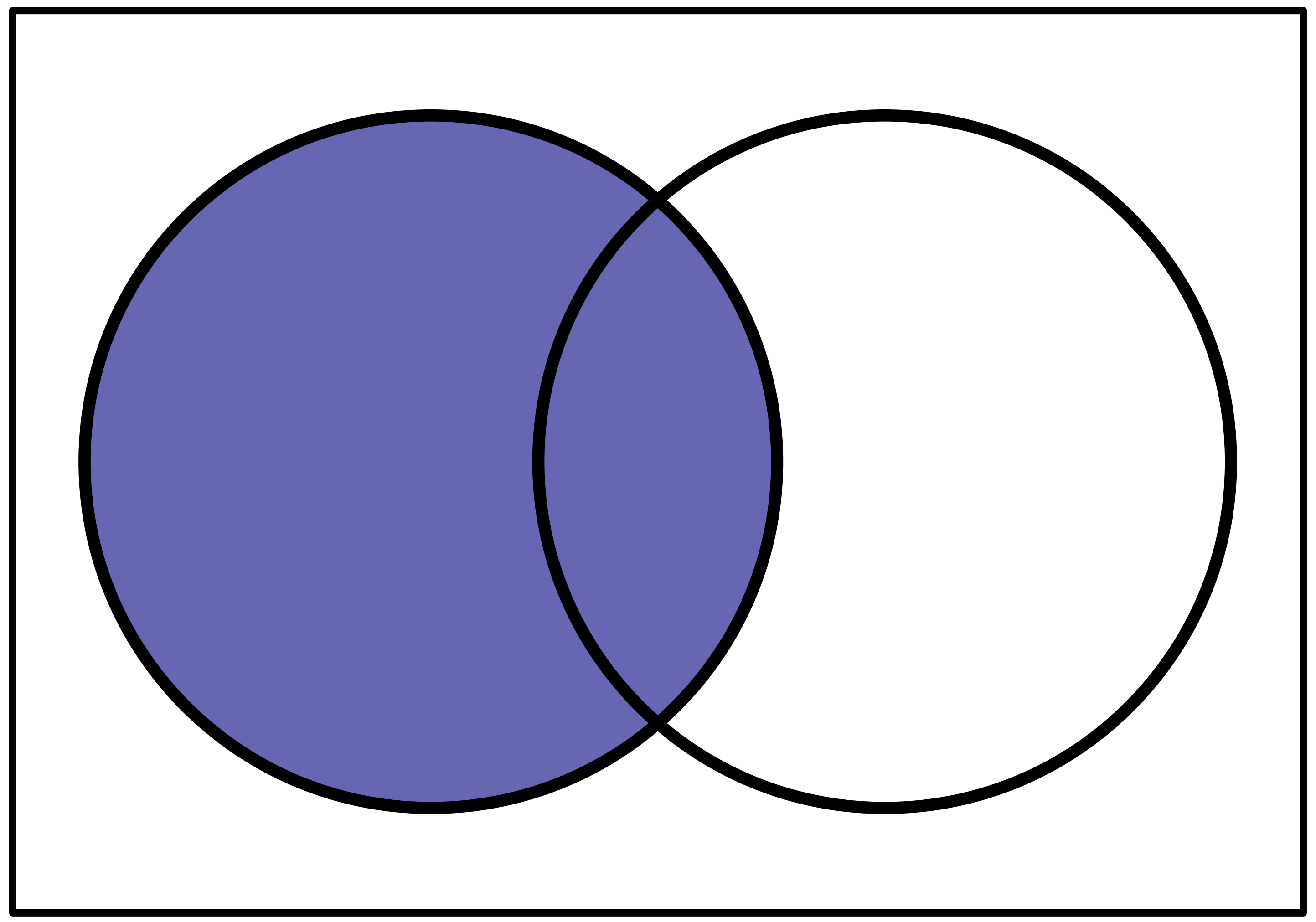
$L_1 \cap L_2 = \{\varepsilon\}$.

Complement

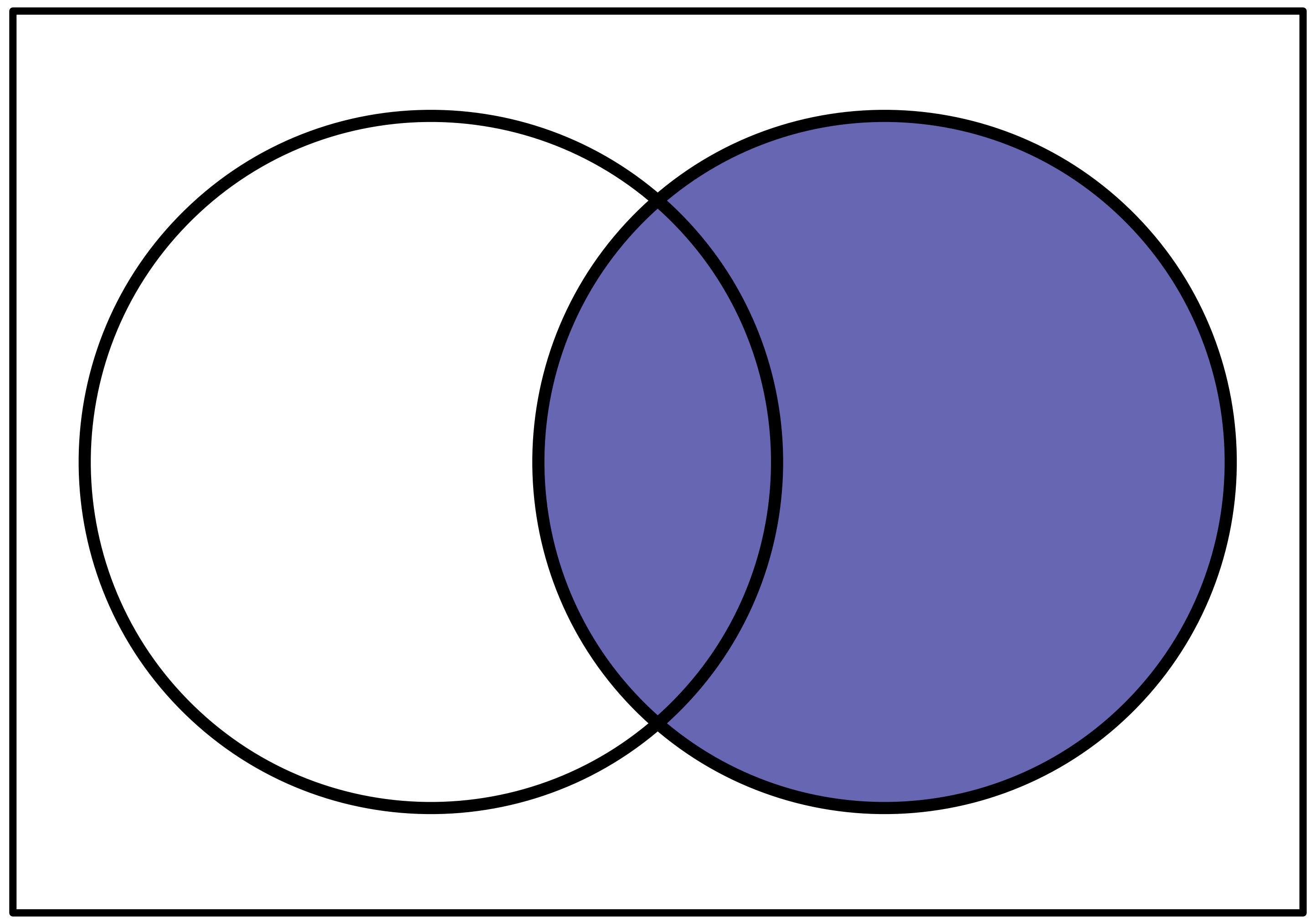
THEOREM There exists a context-free language L such that its complement, $\overline{L}$, is not a context-free language.

Complement

PROOF IDEA Using union and complement, we can construct the intersection.



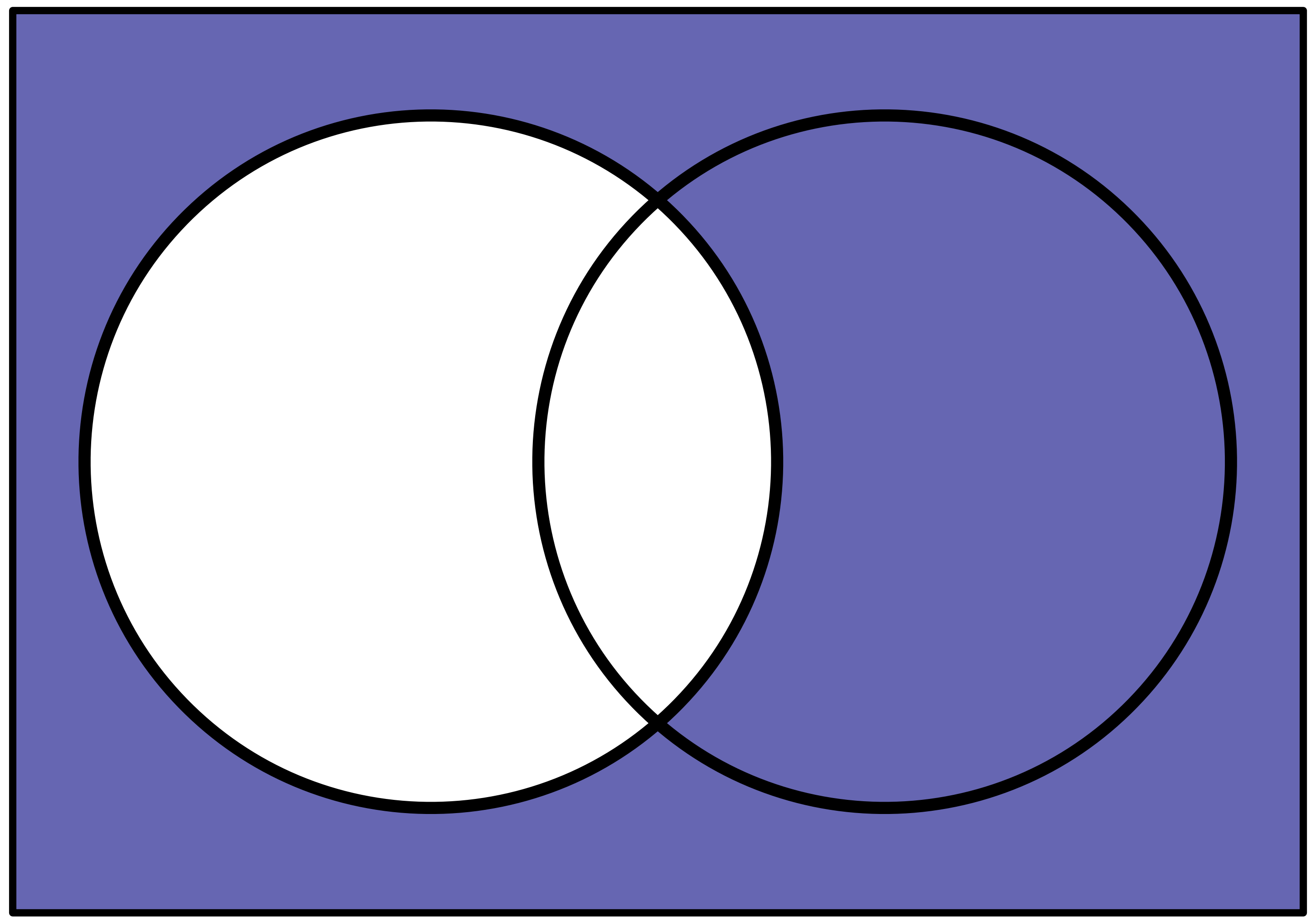
L_1



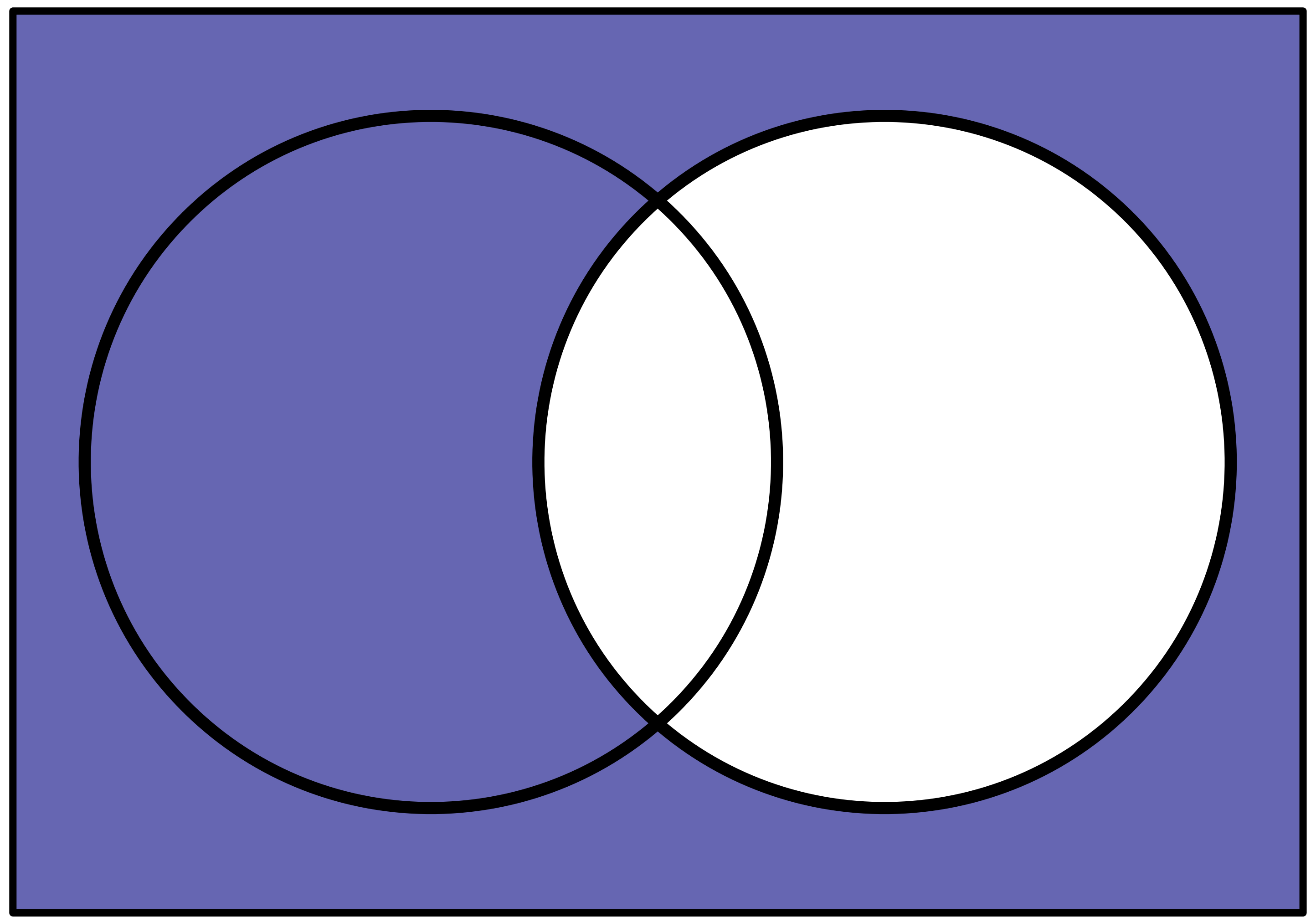
L_2

Complement

PROOF IDEA Using union and complement, we can construct the intersection.



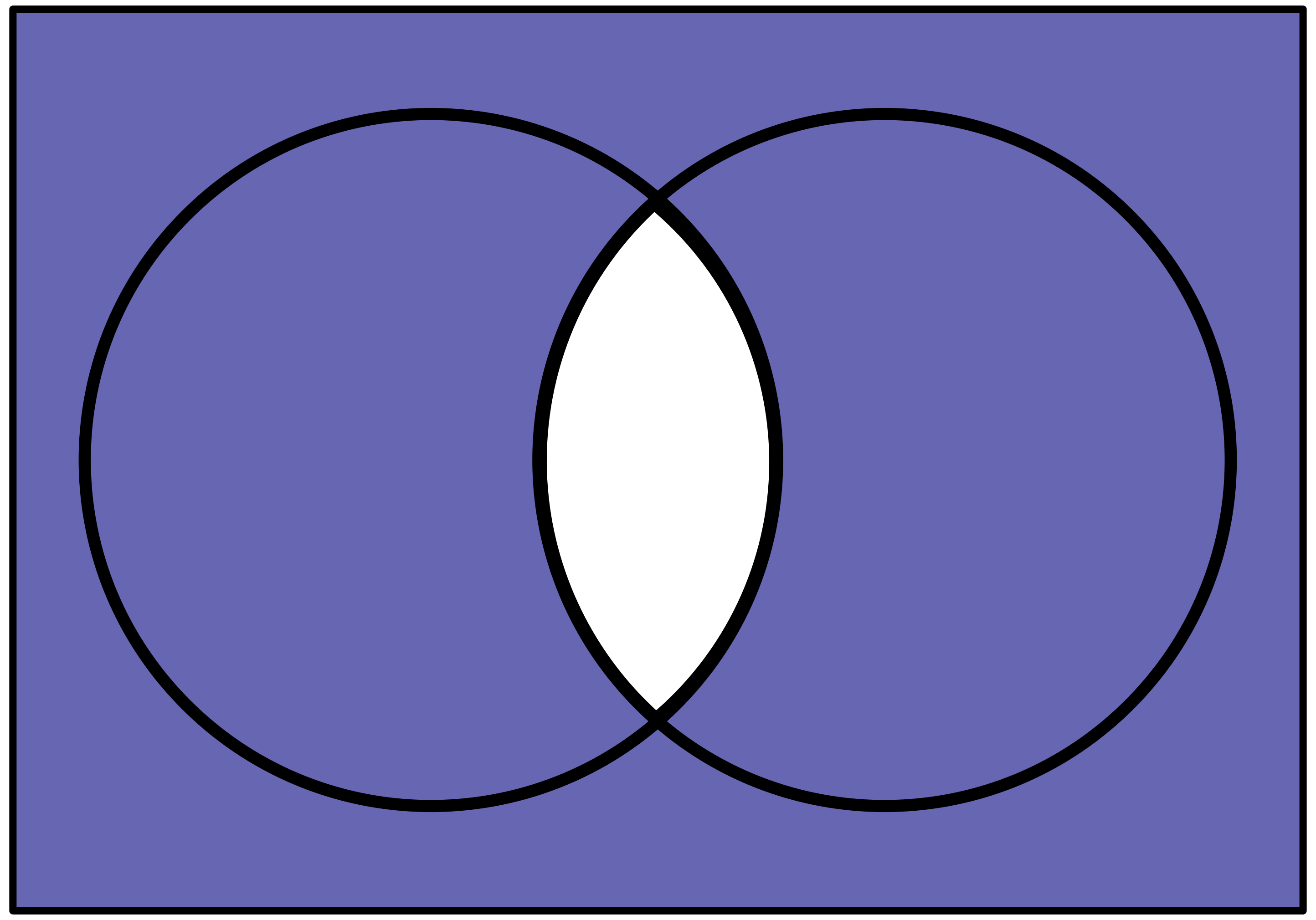
$\overline{L_1}$



$\overline{L_2}$

Complement

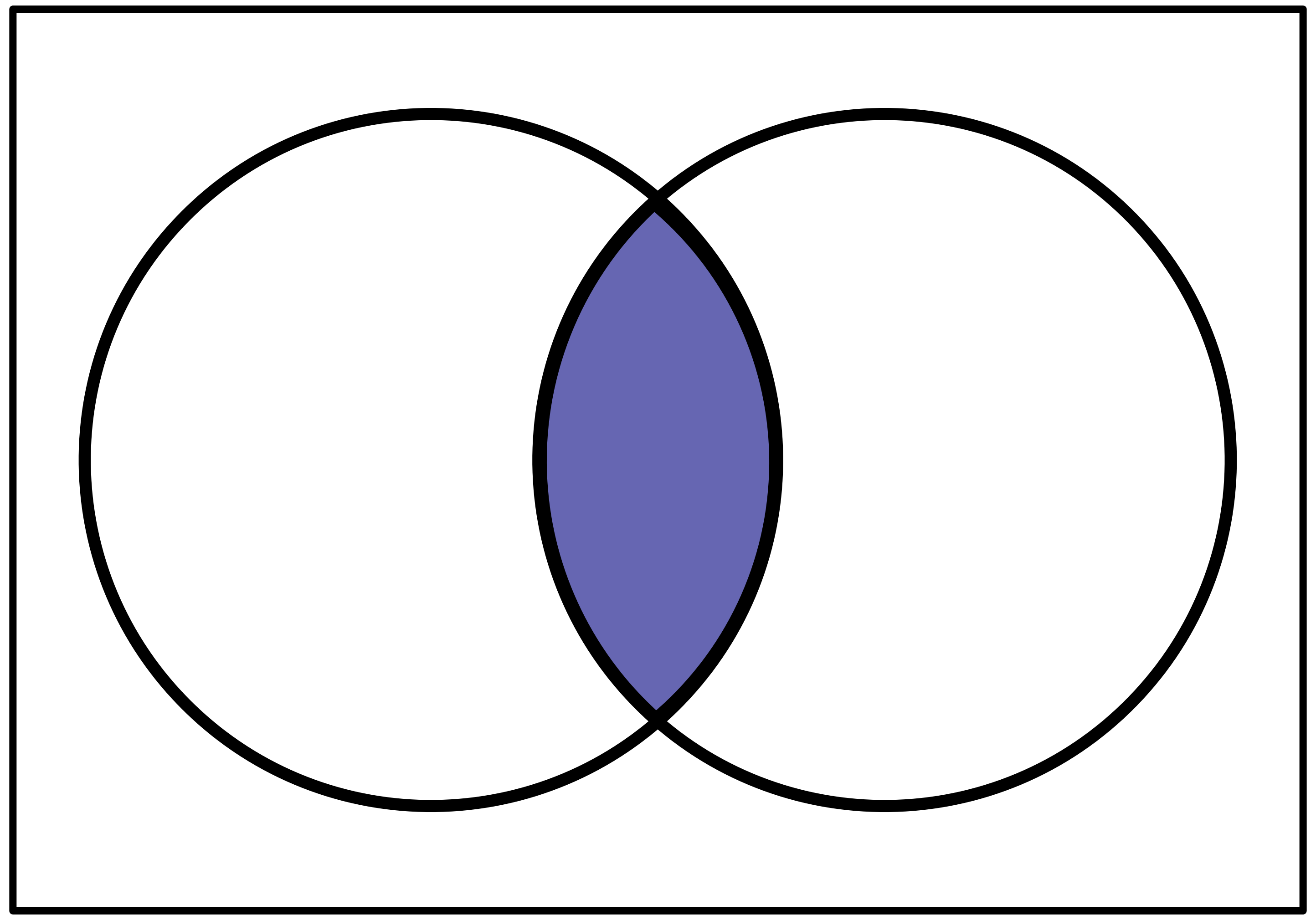
PROOF IDEA Using union and complement, we can construct the intersection.



$\overline{L_1} \cup \overline{L_2}$

Complement

PROOF IDEA Using union and complement, we can construct the intersection.



$\overline{\overline{L_1} \cup \overline{L_2}}$

Complement

PROOF IDEA Using union and complement, we can construct the intersection.

Since the context-free languages are closed under union but ***not*** under intersection, this means they ***can't be closed under complement*** either.

Set difference

THEOREM There exist context-free languages $L_1, L_2 \in \Sigma^*$ such that $L_1 - L_2$ is not a context-free language.

PROOF IDEA We know the complement of a CFL might not be context-free, and we can construct the complement from the difference.

Σ^* is context-free because it is regular.

But $\Sigma^* - L = \overline{L}$, which may not be context-free.

Closure properties of context-free languages

If L_1 and L_2 are context-free languages, then

$L_1 \cup L_2$ is context-free,

$L_1 L_2$ is context-free,

L_1^* is context-free, and

L_1^R is context-free,

but

$L_1 \cap L_2$ is ***not*** guaranteed to be context-free,

$\overline{L_1}$ is ***not*** guaranteed to be context-free, and

$L_1 - L_2$ is ***not*** guaranteed to be context-free!

Context-free languages and beyond

Summary

We've seen that CFLs are strictly more powerful than the regular languages.

They can be described by CFGs (generation) or PDAs (recognition).

- Non-deterministic PDAs are equivalent in power to CFGs
- Deterministic PDAs are less powerful – but are used in practice

Context-free language have a pumping lemma, just as regular languages do.

Computability theory

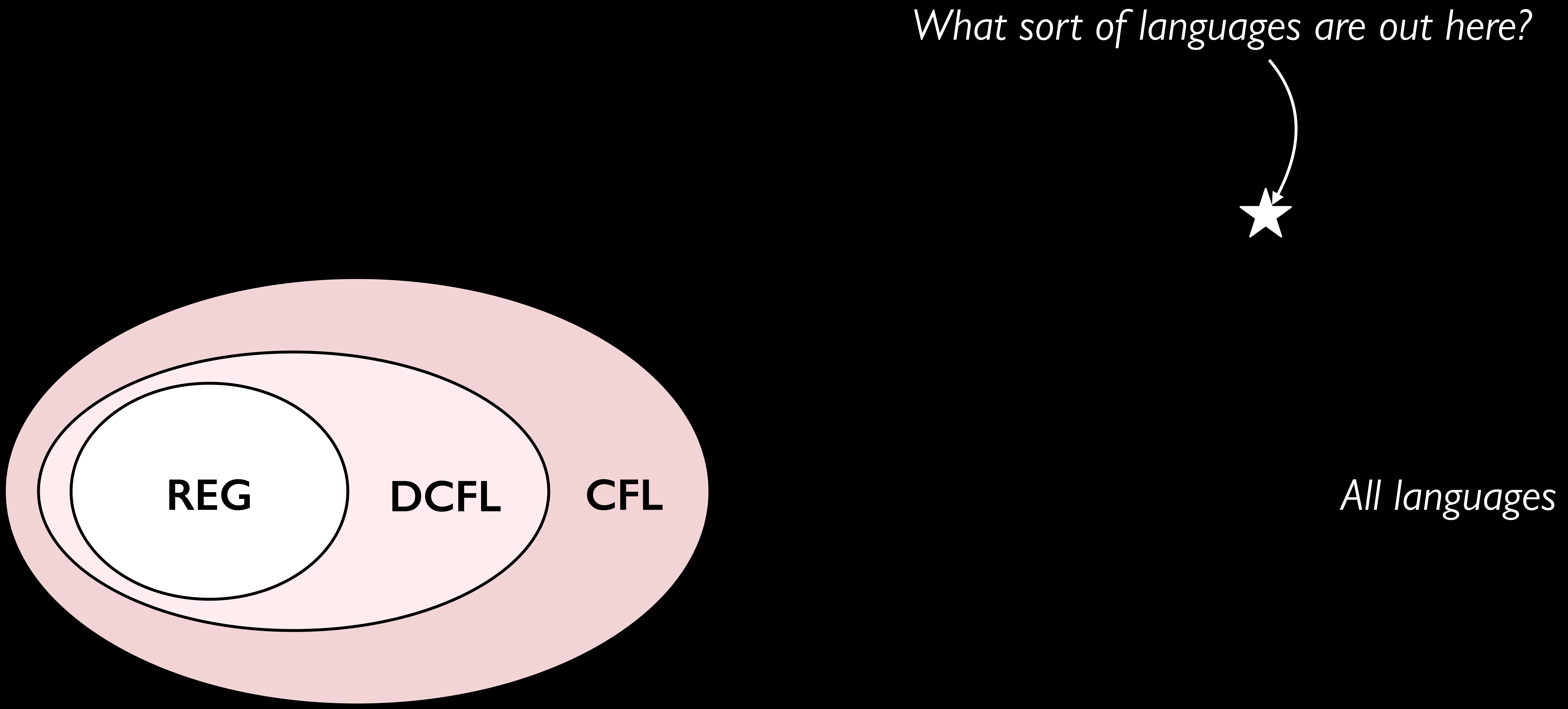
Finite automata represent computers with finite memory.

They recognize precisely the regular languages.

Pushdown automata represent computers with a weak infinite memory.

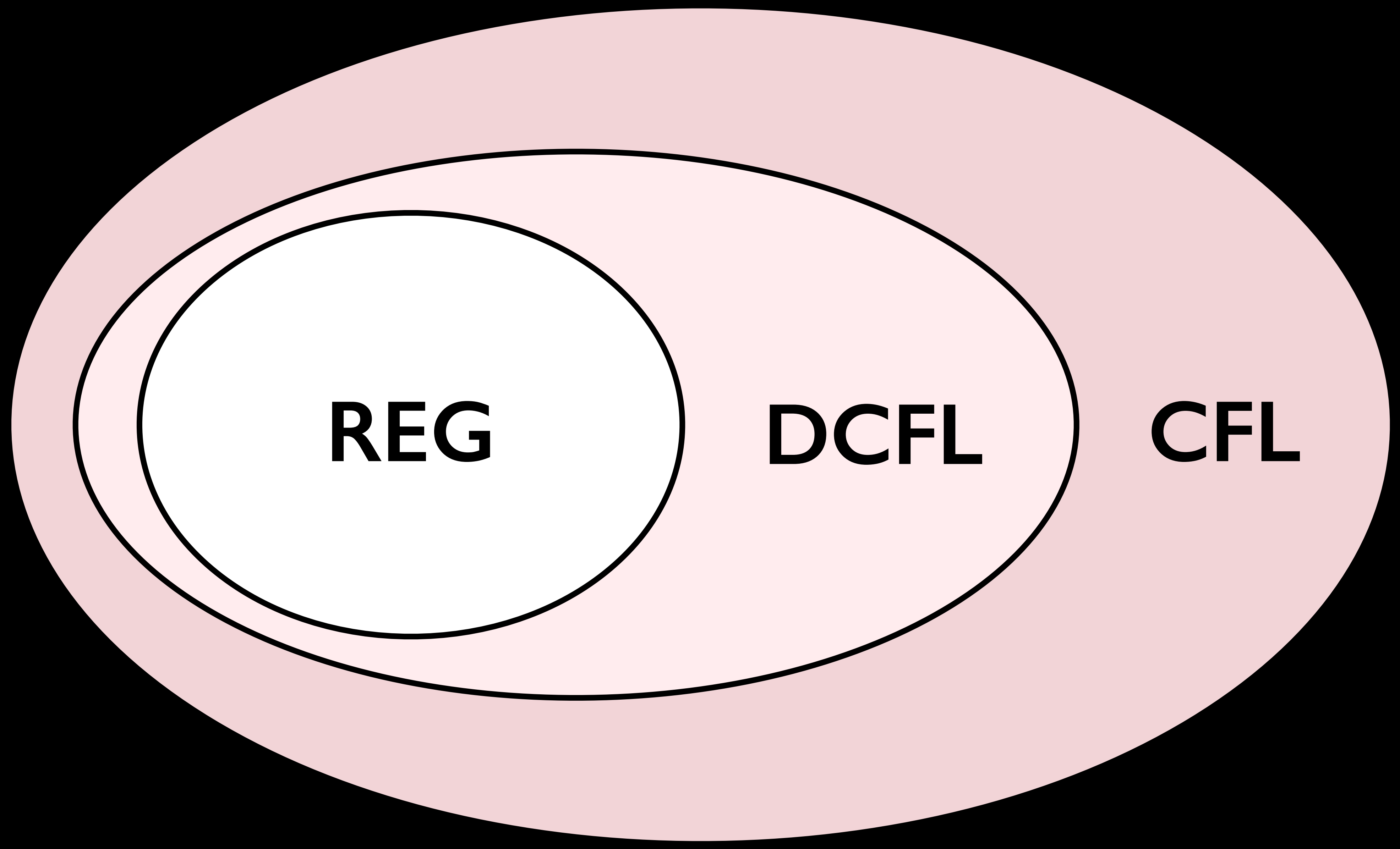
They recognize precisely the context-free languages.

Regular and context-free languages are comparatively weak.


What sort of languages are out here?
REG
DCFL
CFL
All languages

What sort of languages are out here?

A lot! And we'll see more soon!



All languages

Next class we'll have a special topic: We'll look at a practical way to parse a string using a CFG.

Next week we'll review and then have Exam 2.

Acknowledgments

This lecture incorporates material from:

David Chiang, University of Notre Dame

Nancy Ide, Vassar College

Timothy Ng, University of Chicago

Keith Schwarz, Stanford University

Michael Sipser, *Introduction to the Theory of Computation*