Design contex-free grammars that generate:

$L_1 = \{ u\, v : u \in \{a,b\}^* , v \in \{a, c\}^*,$

and $|u| \leq |v| \leq 3\,|u| \}.$

$L_2 = \{ a^p\, b^q\, c^p\, a^r\, b^{2r} : p, q, r \geq 0\}$

Midterm tutorial: Monday June 25 at 7:30pm, ECS 116.

Tutorial this week: Bring any questions you have about assignment #3 or old midterms. Material covered if you have no other questions: Context-free grammars and Pushdown Automata.

Assignment #3 is due at the beginning of class on Tuesday June 26.

Alan Turing's 100th Birthday Celebration

A DAY OF FREE EVENTS in ECS 660

Friday, June 22nd, 2012

No class Friday June 22.
Please attend Turing Celebration Events instead

9:00-9:30	The Life of Alan Turing - **Daniel German**
9:30-11:00	Turing's Seminal 1936 Paper - **John Ellis**
11:00-12:00	The history of the Entscheidungsproblem, from Llull to Turing and beyond - **Bruce Kapron**
12:00 – 1:30	Pizza Lunch, Turing Art, Music & Wumpus World
1:30-2:30	The Turing Test and Searle's "Chinese Room" - **Maarten van Emden**
2:30-3:30	The Enigma machine and Turing's Cryptoanalysis - **Venkatesh Srinivasan**
3:30-4:00	Birthday cake, coffee and tea with special guest **Olive Bailey** - worked with Turing at Bletchley Park
4:00-6:00	**"Breaking the Code"** - Acclaimed film about Turing's life – ECS 104

www.csc.uvic.ca/Events/turing100.htm

Sponsors:

PIMS

University of Victoria
Computer Science

Today:

We will go back over more examples of context-free grammars, pushdown automata and parse trees. I previously went through that material fairly quickly so you would be able to do the assignment last weekend if you wanted to.

L= {w $\in$ {a,b}* : w has abb and a prefix and bba as a suffix}

1. Create a regular context-free grammar that generates L.

2. Give a context-free grammar for L that is not regular.

Recall: A regular context-free grammar can have at most one non-terminal on the RHS of each rule, if there is a non-terminal, it is the last symbol on the RHS of the rule.

$L(M) = \{ w \in \Sigma^* : (s, w, \varepsilon) \vdash^* (f, \varepsilon, \varepsilon)$ for some final state f in $F\}$.

To accept, there must exist a computation which:

1. Starts in the start state with an empty stack.
2. Applies transitions of the PDA which are applicable at each step.
3. Consumes all the input.
4. Terminates in a final state with an empty stack.

PDA's are non-deterministic:

$L = \{ w\, w^R : w \in \{a, b\}^* \}$

Start state: s, Final State: {t}

State	Input	Pop	Next state	Push
s	a	ε	s	A
s	b	ε	s	B
s	ε	ε	t	ε
t	a	A	t	ε
t	b	B	t	ε

Guessing wrong time to switch from s to t gives non-accepting computations.

Some non-accepting computations on aaaa:

1. Transfer to state t too early:

$(s, aaaa, \varepsilon) \vdash (s, aaa, A) \vdash (t, aaa, A)$

$\vdash (t, aa, \varepsilon)$

Cannot finish reading input because stack is empty.

2. Transfer to state t too late:

$(s, aaaa, \varepsilon) \vdash (s, aaa, A) \vdash (s, aa, AA)$

$\vdash (s, a, AAA) \vdash (t, a, AAA) \vdash (t, \varepsilon, AA)$

Cannot empty stack.

Accepting computation on aaaa:

$(s, aaaa, \varepsilon) \vdash (s, aaa, A) \vdash (s, aa, AA)$

$\vdash (t, aa, AA) \vdash (t, a, A) \vdash (t, \varepsilon, \varepsilon)$

The computation started in the start state with the input string and an empty stack.

It terminated in a final state with all the input consumed and an empty stack.

L= {w in {a, b}* : w has the same number of a's and b's}

Start state: s

Final states: {s}

State	Input	Pop	Next state	Push
s	a	ε	s	B
s	a	A	s	ε
s	b	ε	s	A
s	b	B	s	ε

L= {w in {a, b}* : w has the same number of a's and b's}

State state: s, Final states: {f}

State	Input	Pop	Next state	Push
s	ε	ε	t	X
t	a	X	t	BX
t	a	A	t	ε
t	a	B	t	BB
t	b	X	t	AX
t	b	A	t	AA
t	b	B	t	ε
t	ε	X	f	ε

A more deterministic solution:

Stack will never contain both A's and B's.

X- bottom of stack marker.

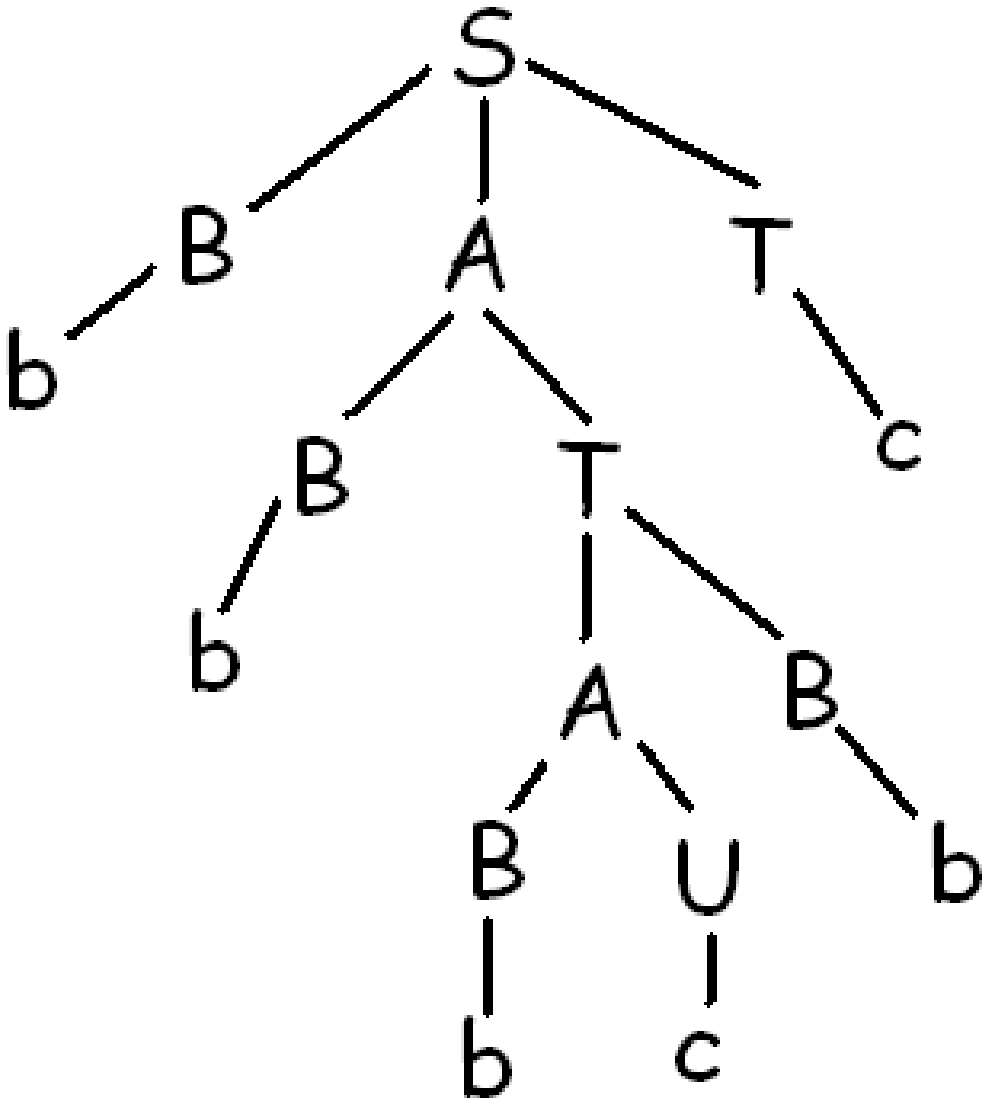
Parse Trees- $G = (V, \Sigma, R\ S)$

Nodes of parse trees are each labelled with one symbol from $V \cup \{\varepsilon\}$.

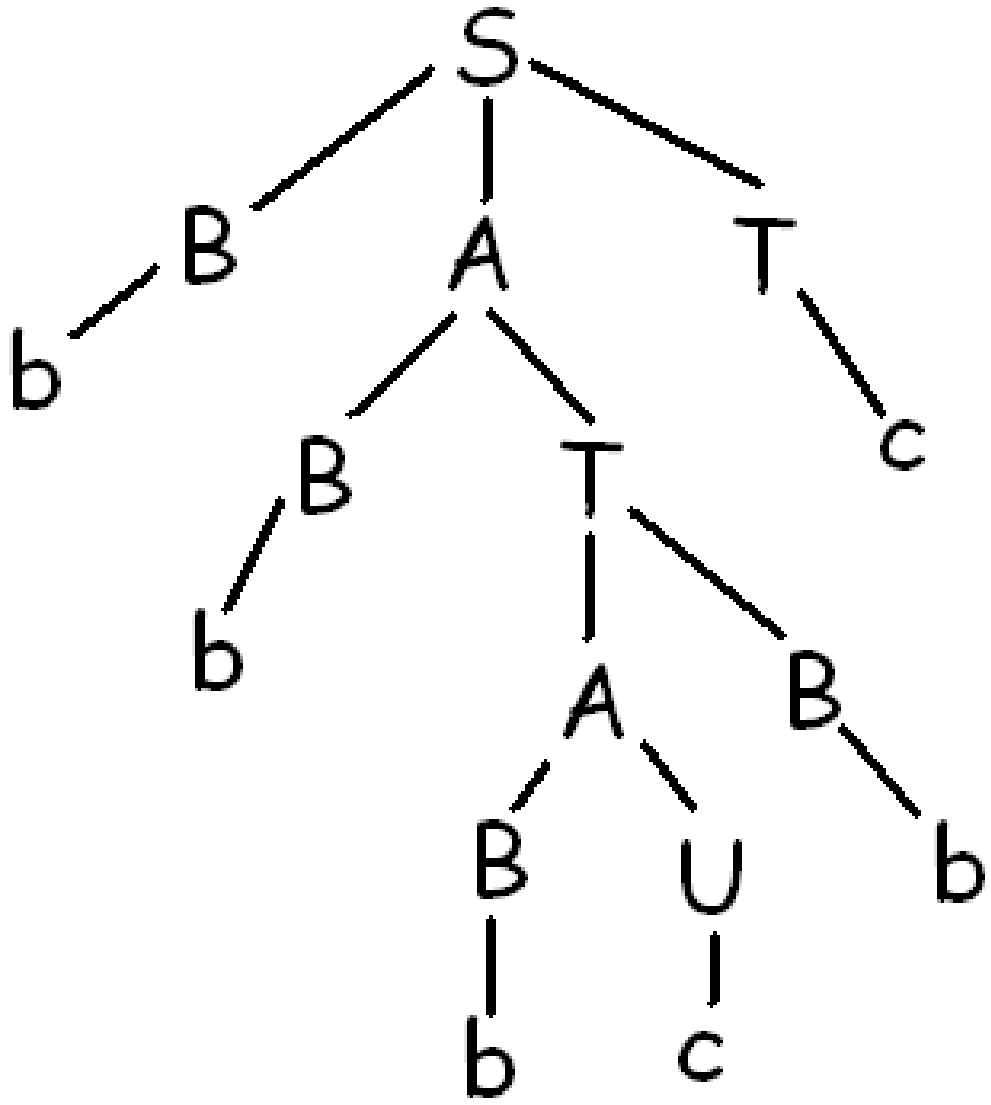
Node at top- root, labelled with start symbol S.

Leaves- nodes at the bottom.

Yield- concatenate together the labels on the leaves going from left to right.

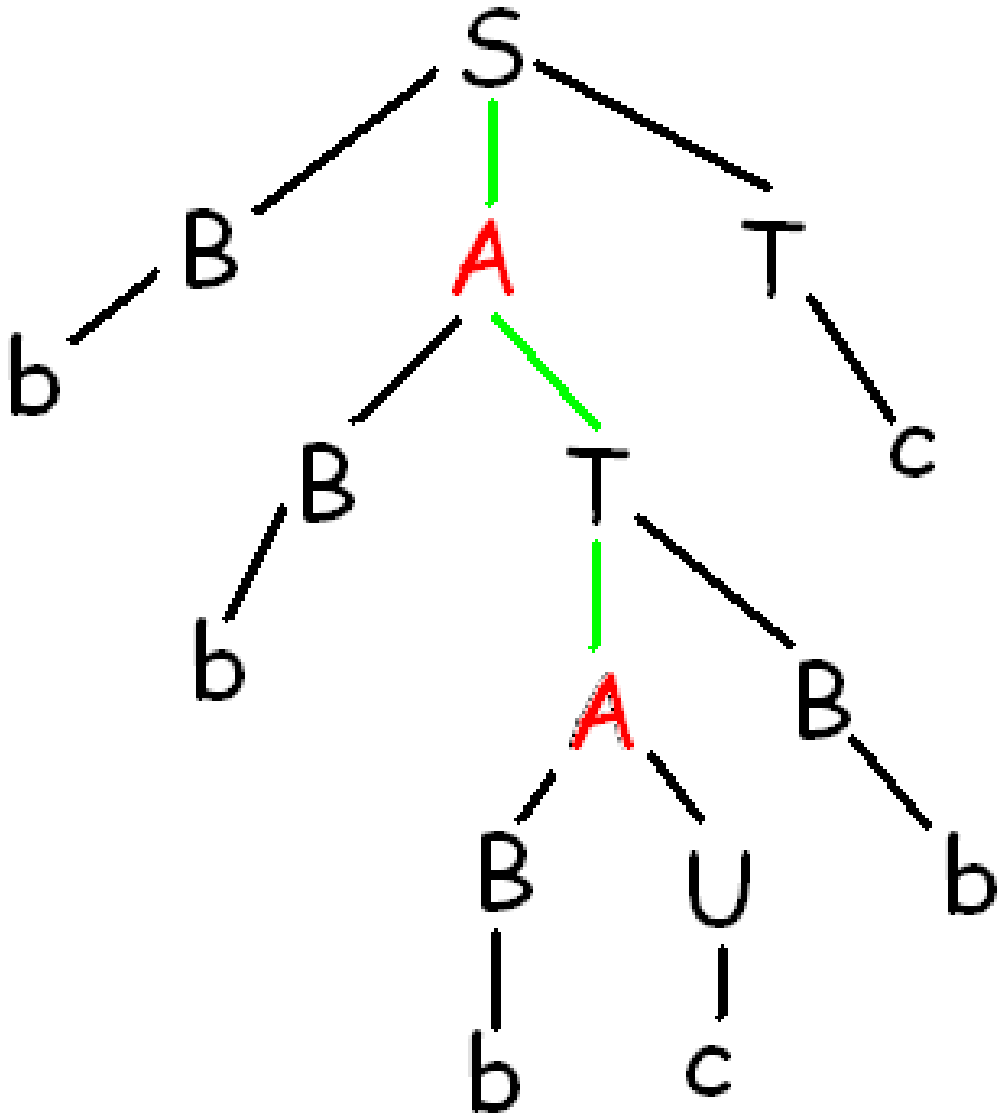


Find the leftmost derivation for this parse tree.



To pump:

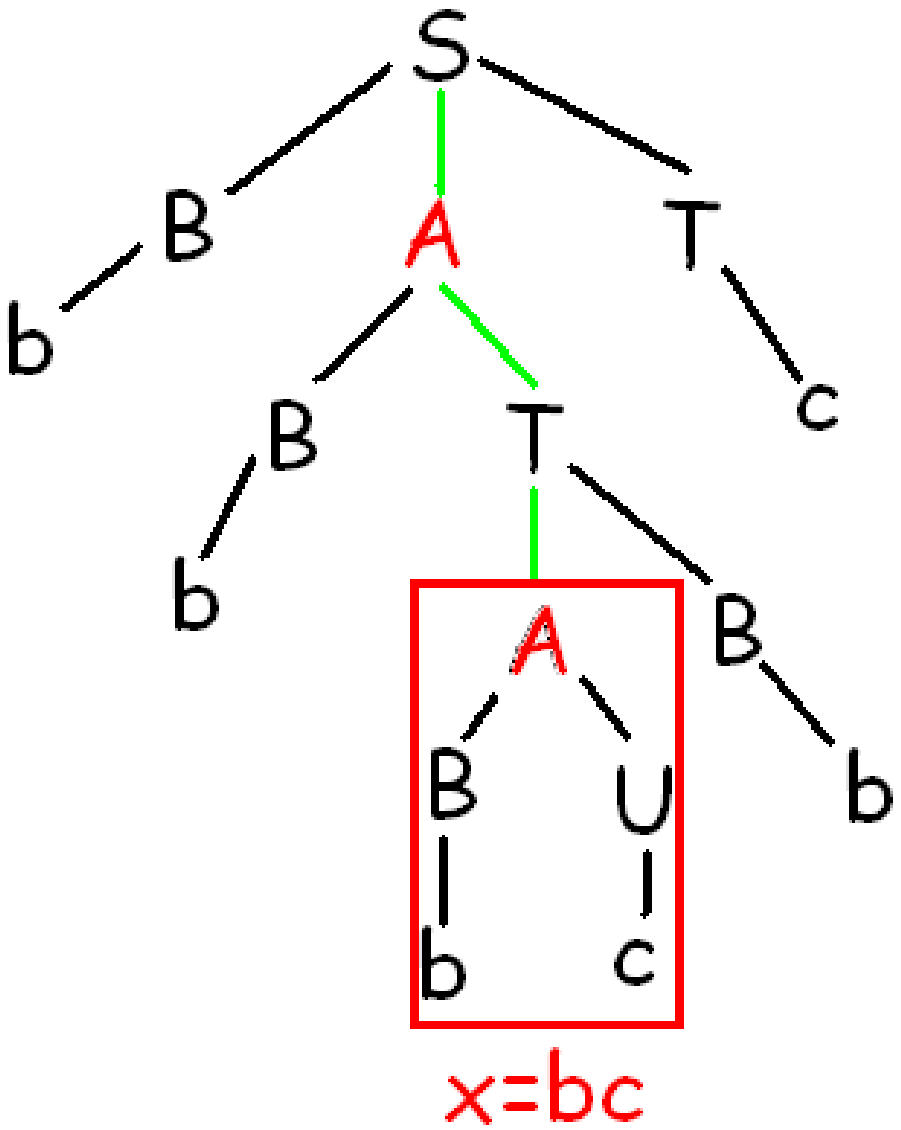
Look for a path from root with a repeated non-terminal.



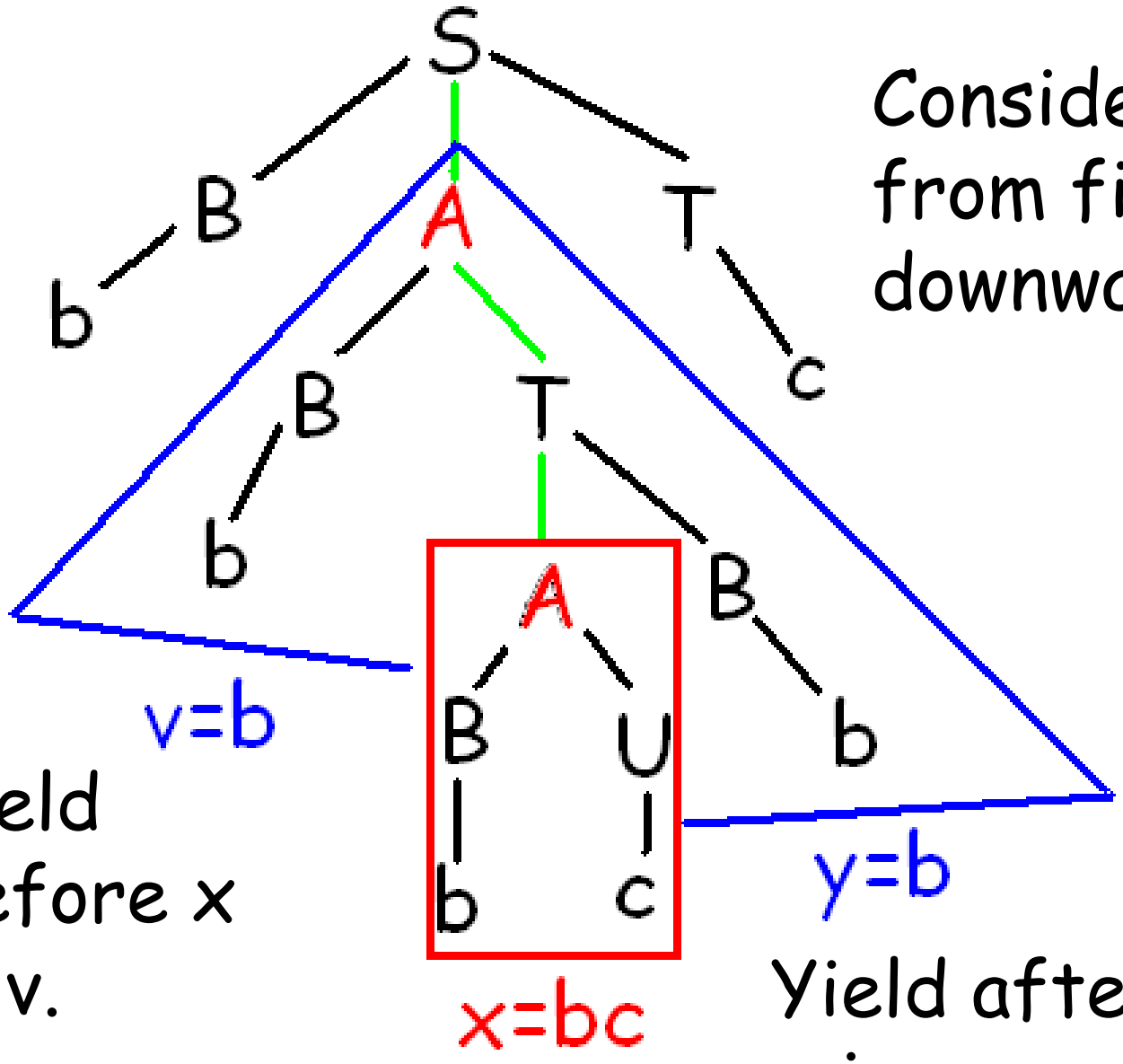
Path:

S - A - T - A

Non-terminal A is repeated.



Yield from second copy downwards gives x.



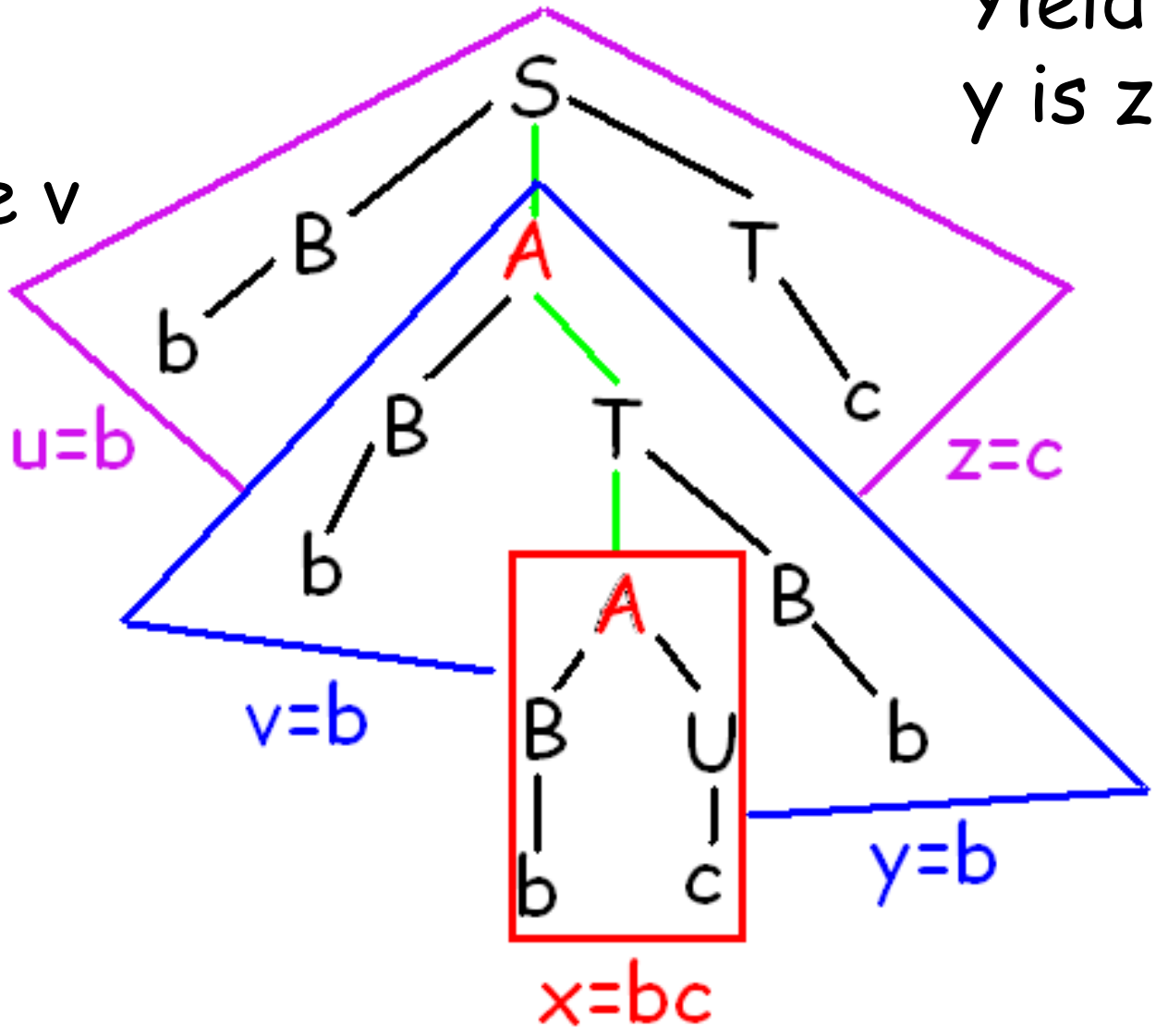
Consider tree from first copy downwards.

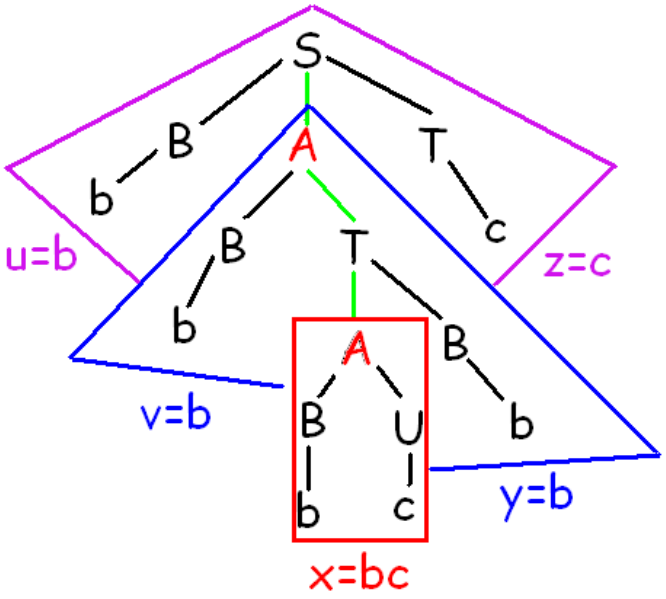
Yield before x is v.

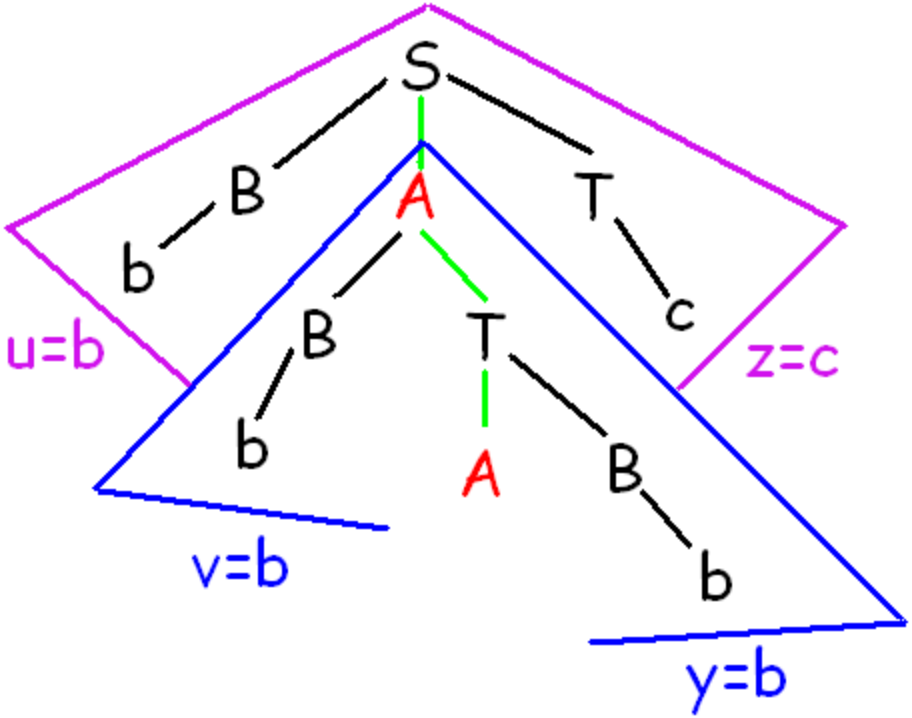
Yield after x is y.

Yield of whole tree before v is u.

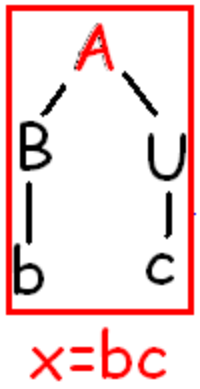
Yield after y is z.

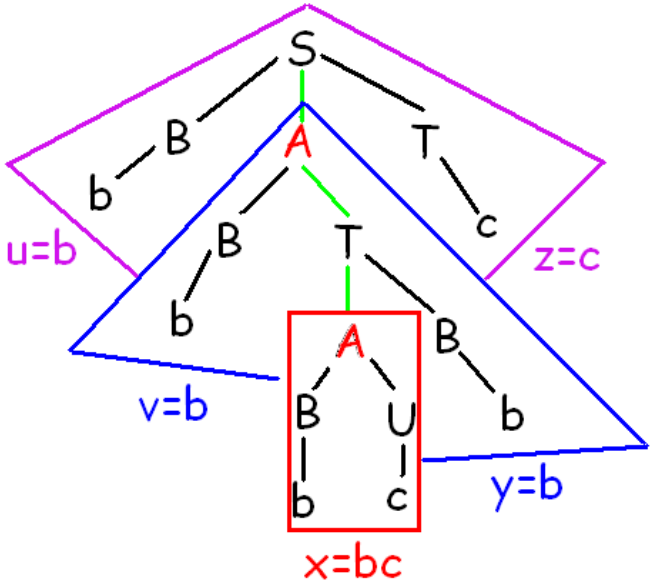




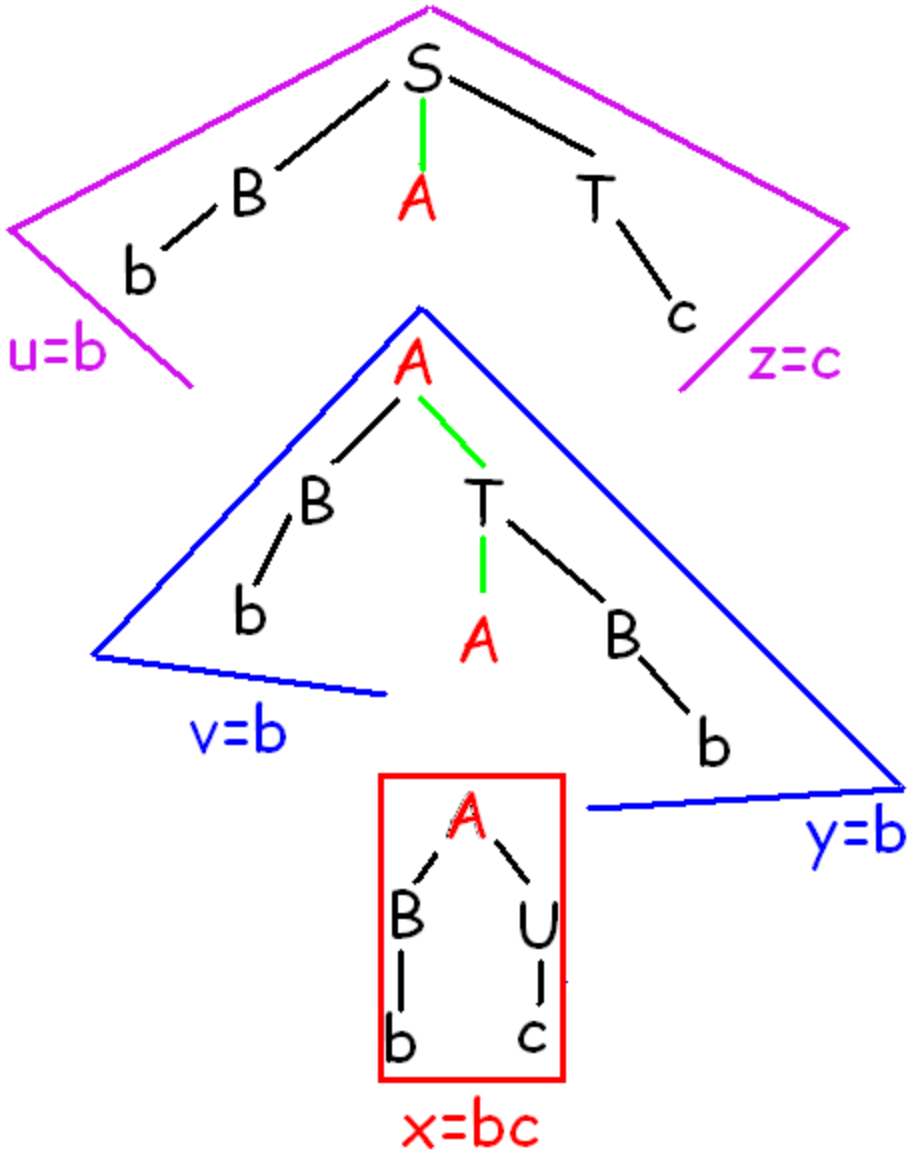


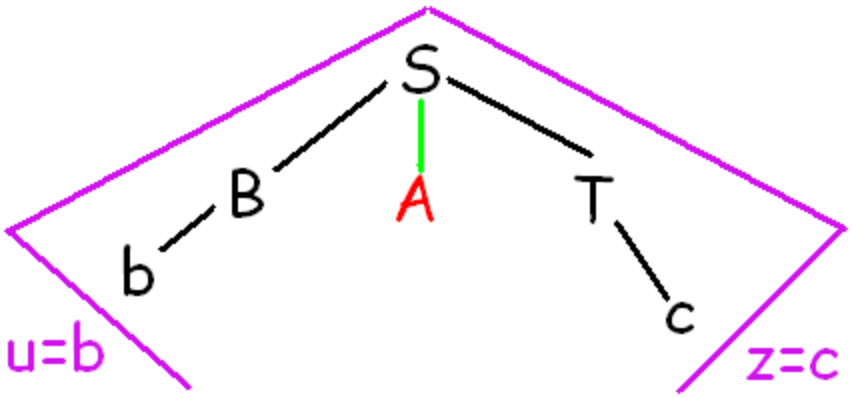
Cut off part of tree from second copy downwards.



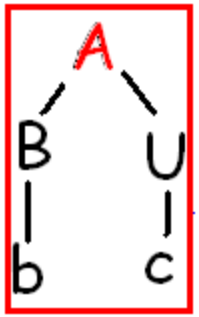


Cut off part from first copy downwards.





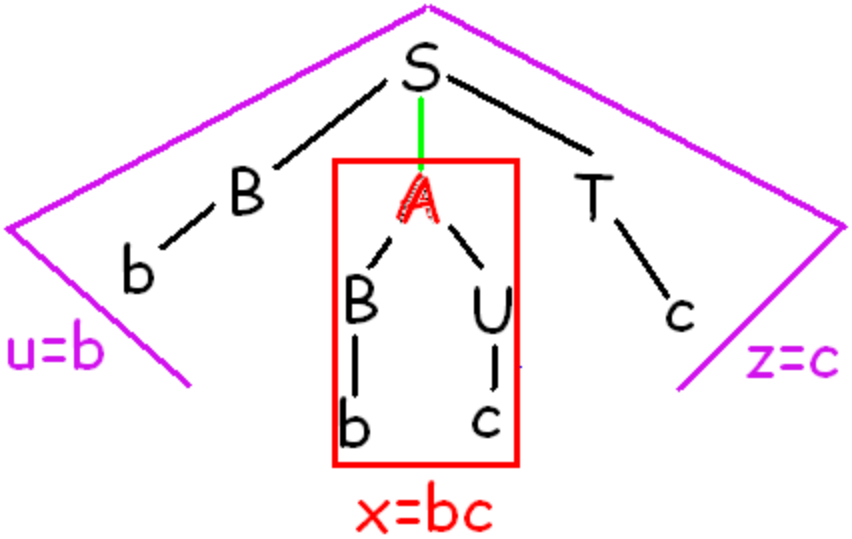
To pump zero times:

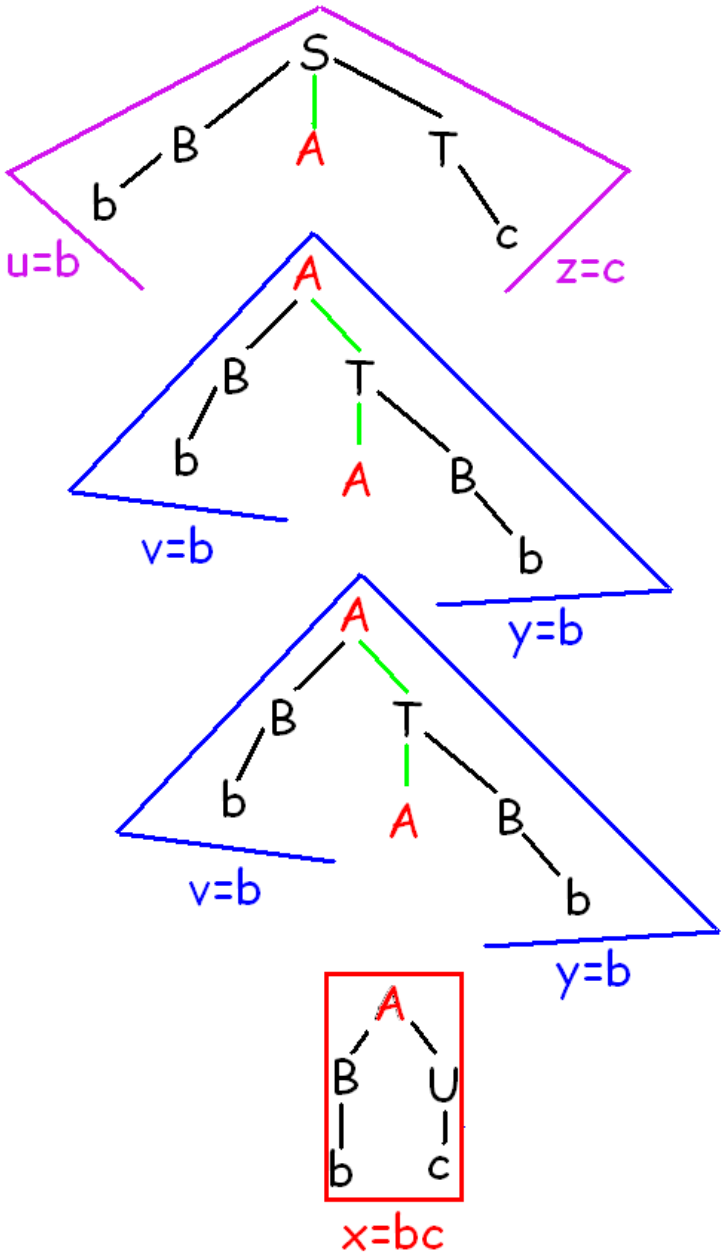


New yield:

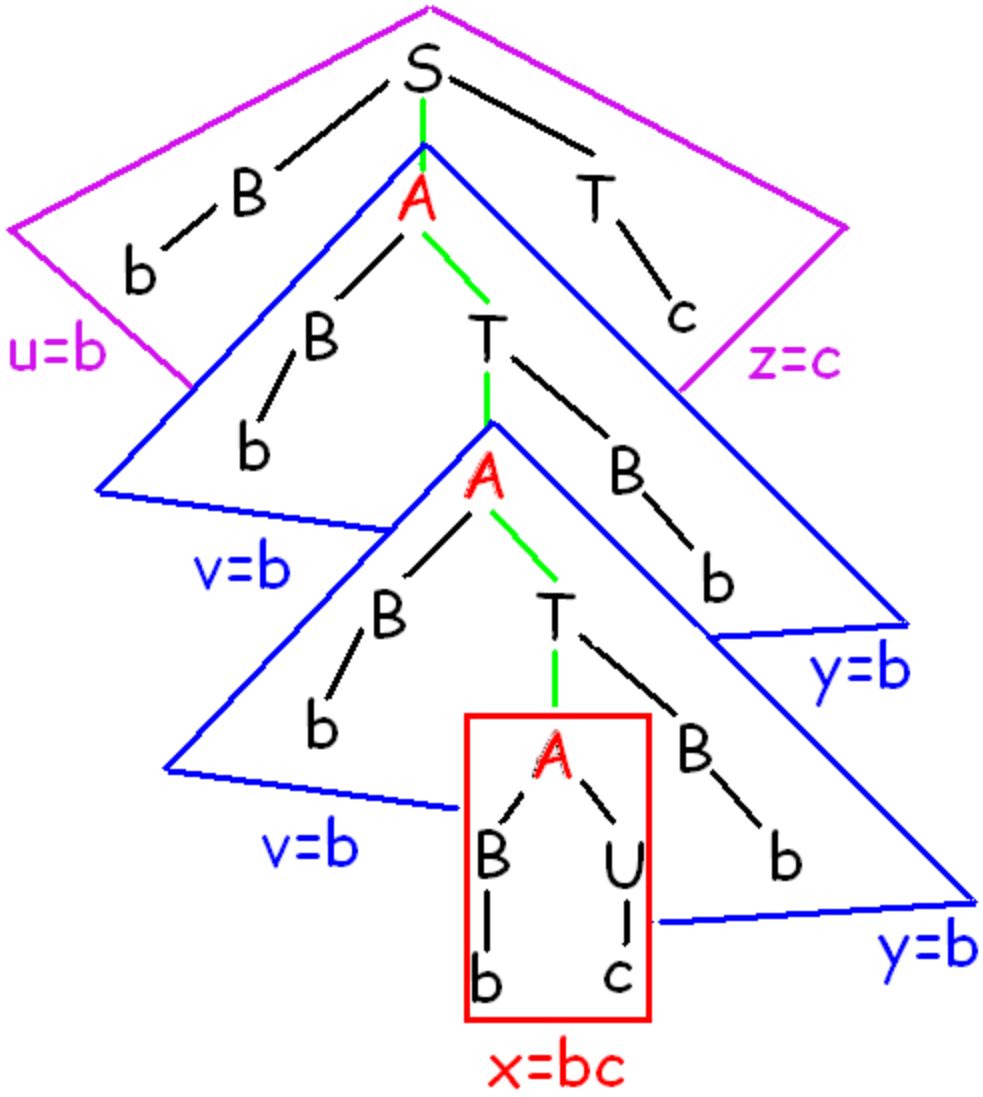
u x z = b bc c

= u v^0 x y^0 z

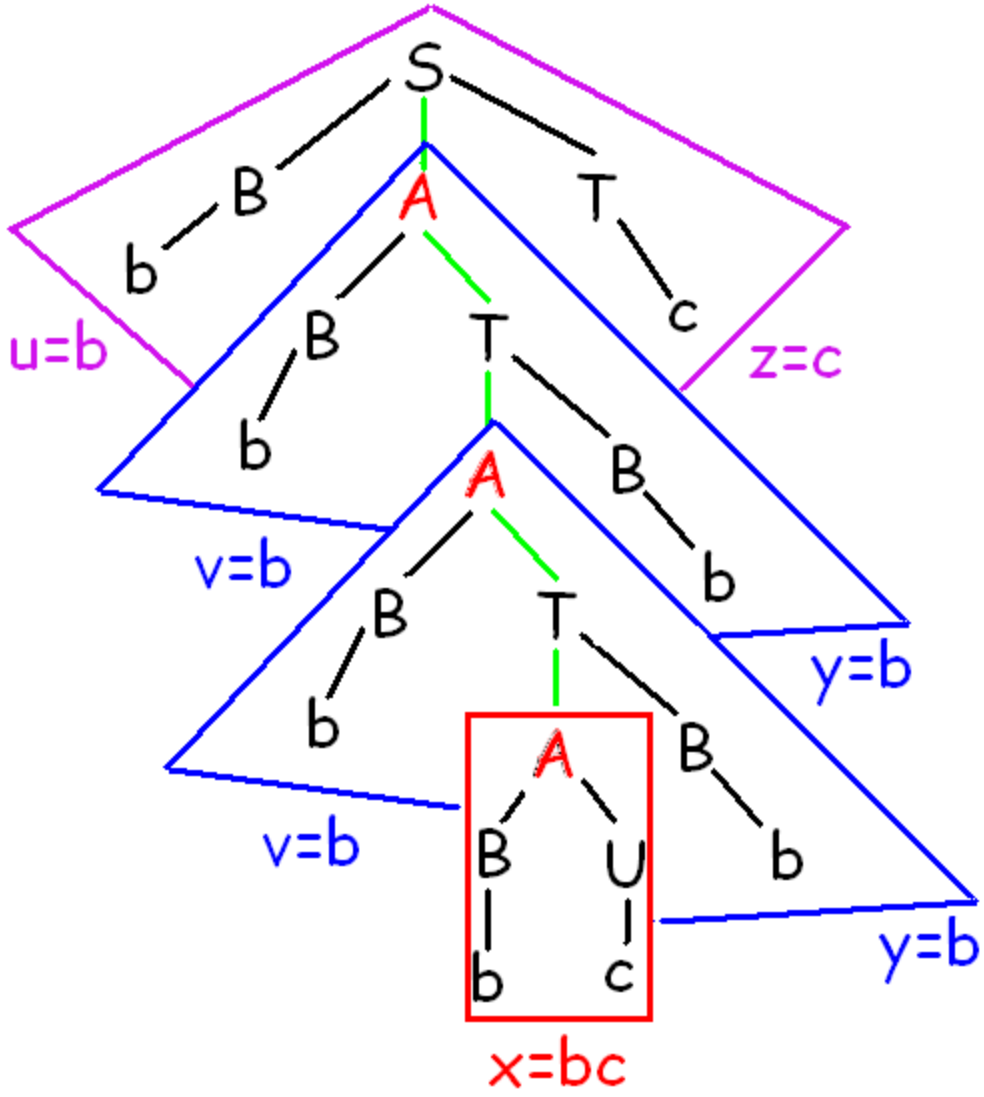




To pump twice:



Pumping twice:



New yield is:

u v v x y y z

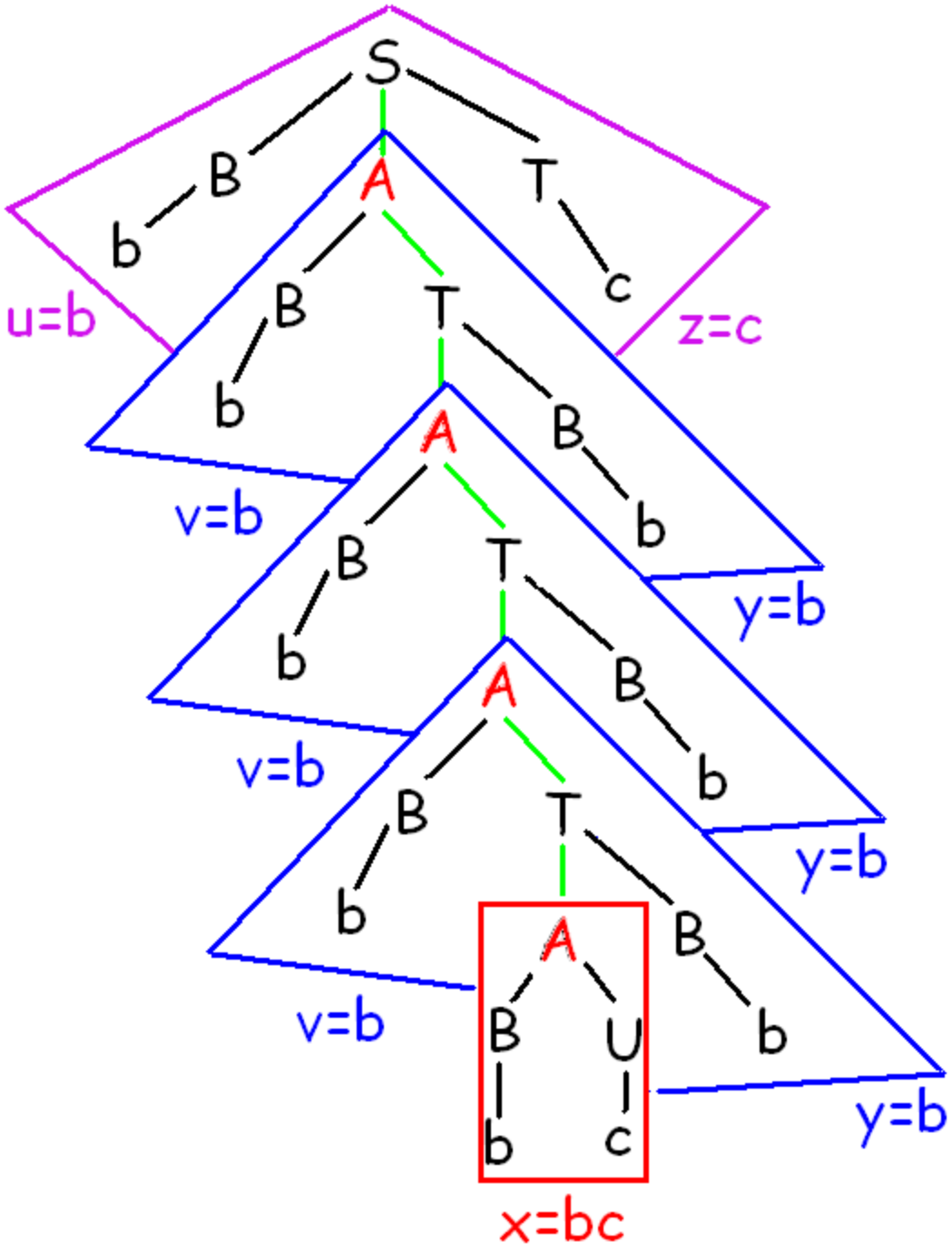
= u v^2 x y^2 z

To pump three times:

New yield is:

u vvv x yyy z

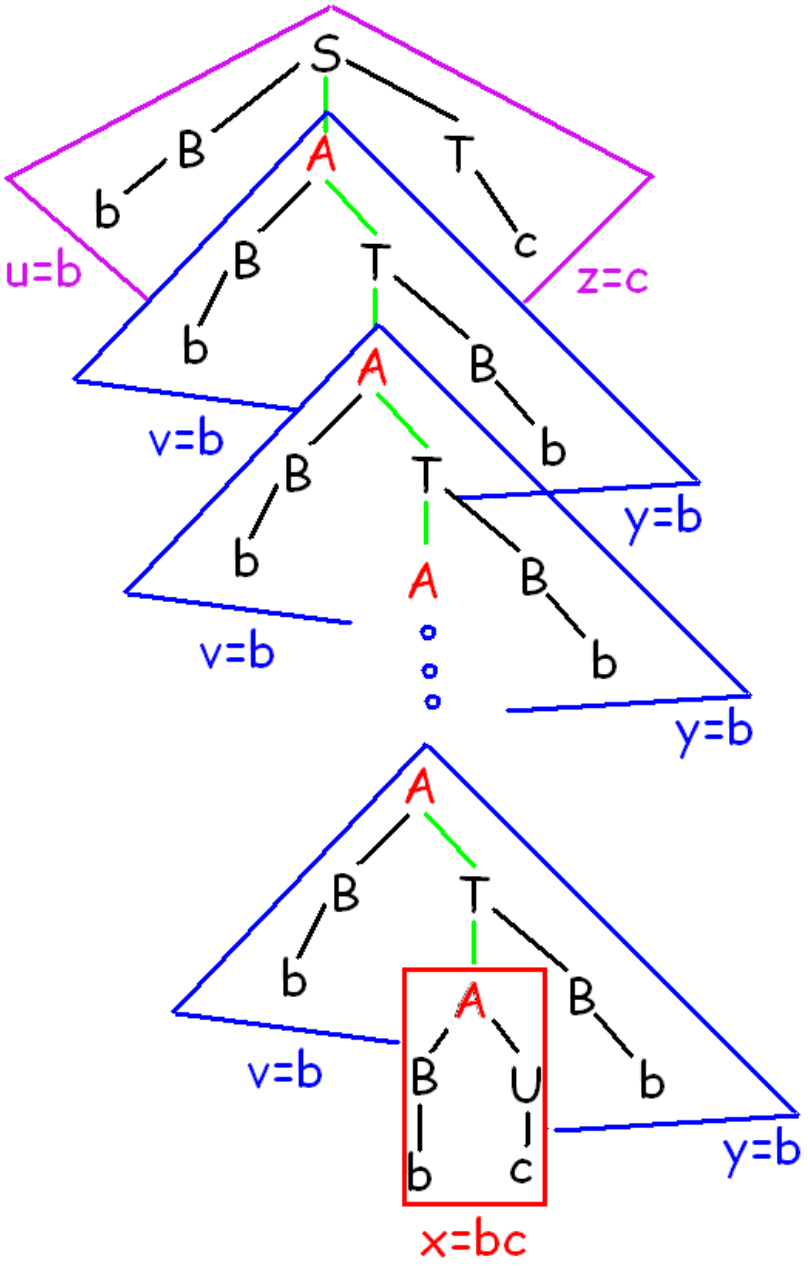
= u v^3 x y^3 z



To pump n times:

New yield is:

$= u\, v^n\, x\, y^n\, z$



The Pumping Theorem for Context-Free Languages:

Let G be a context-free grammar.

Then there exists some constant k which depends on G such that for any string w which is generated by G with $|w| \geq k$,

there exists u, v, x, y, z, such that

1. $w = u\ v\ x\ y\ z$,
2. $|v| + |y| \geq 1$, and
3. $u\ v^n\ x\ y^n\ z$ is in L for all $n \geq 0$.

1. Draw a parse tree for the following derivation:

$S \Rightarrow C\ A\ C \Rightarrow C\ A\ b\ b \Rightarrow b\ b\ A\ b\ b \Rightarrow$

$b\ b\ B\ b\ b \Rightarrow b\ b\ a\ A\ a\ a\ b\ b$

$\Rightarrow b\ b\ a\ b\ a\ a\ b\ b$

2. Show on your parse tree u, v, x, y, z as per the pumping theorem.

3. Prove that the language for this question is an infinite language.