## **Example: Regular Grammar**<sub>JP</sub>

Enter a regular grammar that generates the following language L over the alphabet  $\Sigma = \{a, b\}$ : L = { w | w contains at least two **a** symbols }

## Sample Solution (see RG XaXaX.jff)

Consider that the only constraint is two appearances of the symbol **a** and that any number of symbols may appear before and between them.

Recall that a regular grammar may be specified in right-linear format. Thus all productions must have at most one variable in the right-hand side and that variable must be consistently to the right of any terminals.

Note that one possible DFA for this language is the following (see DFA\_XaXaX.jff):



This suggests that the start symbol, in correspondence to state q0, should be able to produce any number of **b** symbols followed by a string that contains the required number of **a** symbols.

 $S \rightarrow bS$ 

Further, once an  $\mathbf{a}$  is produced, the remaining string must simply contain at least one  $\mathbf{a}$  symbol, corresponding to state q1.

 $S \to a A$ 

Similarly, the variable A should be able to produce any number of b symbols followed by a string that contains at least one a symbol.

$$\begin{array}{c} A \rightarrow bA \\ A \rightarrow aB \end{array}$$

The remaining rules combine to produce all strings over  $\{a, b\}^*$ .

 $B \rightarrow \epsilon \mid aB \mid bB$ 

Here is the DFA will states annotated with the variables of this grammar.



1. Enter this grammar into JFLAP.

			Editor	
Table 1	Fext Siz	ze		
		0		-
LHS		RHS		
S	$\rightarrow$	bS		
S	$\rightarrow$	ъA		
А		ьbА		
А		aВ		
В		λ		
В		ъ		
В		bB		

2. Check the type of grammar using *Test* > *Test for Grammar Type*.

HS	RHS													
5	$\rightarrow$ bS													
5	$\rightarrow$ aA													
4	ightarrow bA													
4	$\rightarrow$ aB													
3	$\rightarrow \lambda$													
3	$\rightarrow$ aB													
3	$\rightarrow$ bB													
•				Gran	nma	ır Typ	e							
	(lat)	This is a rig	ht-linear	Gramm	ıar (I	Regu	lar Gr	ramm	ar anc	l Cont	text-f	ree G	ramma OK	ır)

3. Verify known strings using the Brute Force Parse.

ila In	nut T	est Conv	ort	JFLAP : (RG_Xa	XaX.jff)	
ile ili	iput i	est conv	en	Editor	Parcor	
				Editor	Parser	
Table 1	Fext Size			-		
				0		
<b></b>				ion Table		•
Star	t Pause	Step	erivat	tion Table		Ŧ
, r						
Input	abba					
String	accepte	d! 6 nodes	genera	ated.		
				~		
LHS		RHS		Production	Derivation	
					c	
S	$\rightarrow$	bS	-		3	
S	$\rightarrow$	bS		S→aA	aA	
S S	$\rightarrow$ $\rightarrow$	bS aA		S→aA A→bA	aA abA	
S S	$\rightarrow$ $\rightarrow$ $\rightarrow$	bS aA		S→aA A→bA A→bA A→aB	aA abA abbA abbA	
S S A	$\begin{array}{c} \rightarrow \\ \rightarrow \\ \rightarrow \end{array}$	bS aA bA		$S \rightarrow aA$ $A \rightarrow bA$ $A \rightarrow bA$ $A \rightarrow aB$ $B \rightarrow \lambda$	aA abA abbA abbaB abba	
S S A A	$\begin{array}{c} \rightarrow \\ \rightarrow \\ \rightarrow \\ \rightarrow \\ \rightarrow \end{array}$	bS aA bA aB		S→aA A→bA A→bA A→aB B→λ	aA abA abbA abbaB abba	
S S A A B	$ \begin{array}{c} \rightarrow \\ \rightarrow $	bS aA bA aB λ		S→aA A→bA A→bA A→aB B→λ	aA abA abbA abbaB abba	
S A A B	$\begin{array}{c} \rightarrow \\ \rightarrow \\ \rightarrow \\ \rightarrow \\ \rightarrow \\ \rightarrow \\ \rightarrow \end{array}$	bS aA bA aB λ		$S \rightarrow aA$ $A \rightarrow bA$ $A \rightarrow aB$ $B \rightarrow \lambda$	aA abA abbA abbaB abbaB	
S A A B B	$\begin{array}{c} \rightarrow \\ \rightarrow \end{array}$	bS aA bA aB λ aB		$S \rightarrow aA$ $A \rightarrow bA$ $A \rightarrow bA$ $A \rightarrow aB$ $B \rightarrow \lambda$	aA abA abbA abbaB abba	
S A A B B B	$\begin{array}{c} \rightarrow \\ \rightarrow $	bS aA bA aB λ aB bB		S→aA A→bA A→bA A→aB B→ $\lambda$	aA abA abbA abbaB abba	
S A A B B B B	$\begin{array}{c} \rightarrow \\ \rightarrow $	bS aA bA aB λ aB bB		S→aA A→bA A→bA A→aB B→λ	aA abA abbA abbaB abba	
S A A B B B	$\begin{array}{c} \rightarrow \\ \rightarrow $	bS aA bA aB λ aB bB	A	S→aA A→bA A→bA A→aB B→λ	aA abA abbA abbaB abba	
S A A B B B	$\begin{array}{c} \rightarrow \\ \end{array}$	bS aA bA aB λ aB bB		S→aA A→bA A→bA A→aB B→λ	aA abA abbA abbaB abba	