Regular Pumping Lemma

We use one of the provided examples in JFLAP to explain the regular pumping lemma. Remember that to show that a language is not regular using the contrapositive argument of the pumping lemma, you have to show the following.

Regardless of the value of m chosen, there exists some string w in the provided language of length greater than m such that there is no way that the string w can be decomposed into three parts $w = xyz$ and satisfy the following 3 conditions

1. $|y| > 0$
2. $|xy| \leq m$
3. xy^iz is in the language for all $i \geq 0$.

JFLAP treats the showing of a language to not be regular in a manner similar to adversarial arguments. That is, the user is given the chance to pick a pumping length and the computer will show why that will not work by first producing a string and then showing the user how regardless of which way they go about making their partition into x, y and z.

The chosen example is $L = \{a^n b^k c^{n+k} : n \geq 0, k \geq 0\}$

Solution

As with any proof involving showing a language to not be regular using the pumping lemma, assume the language is regular and has a pumping length m.

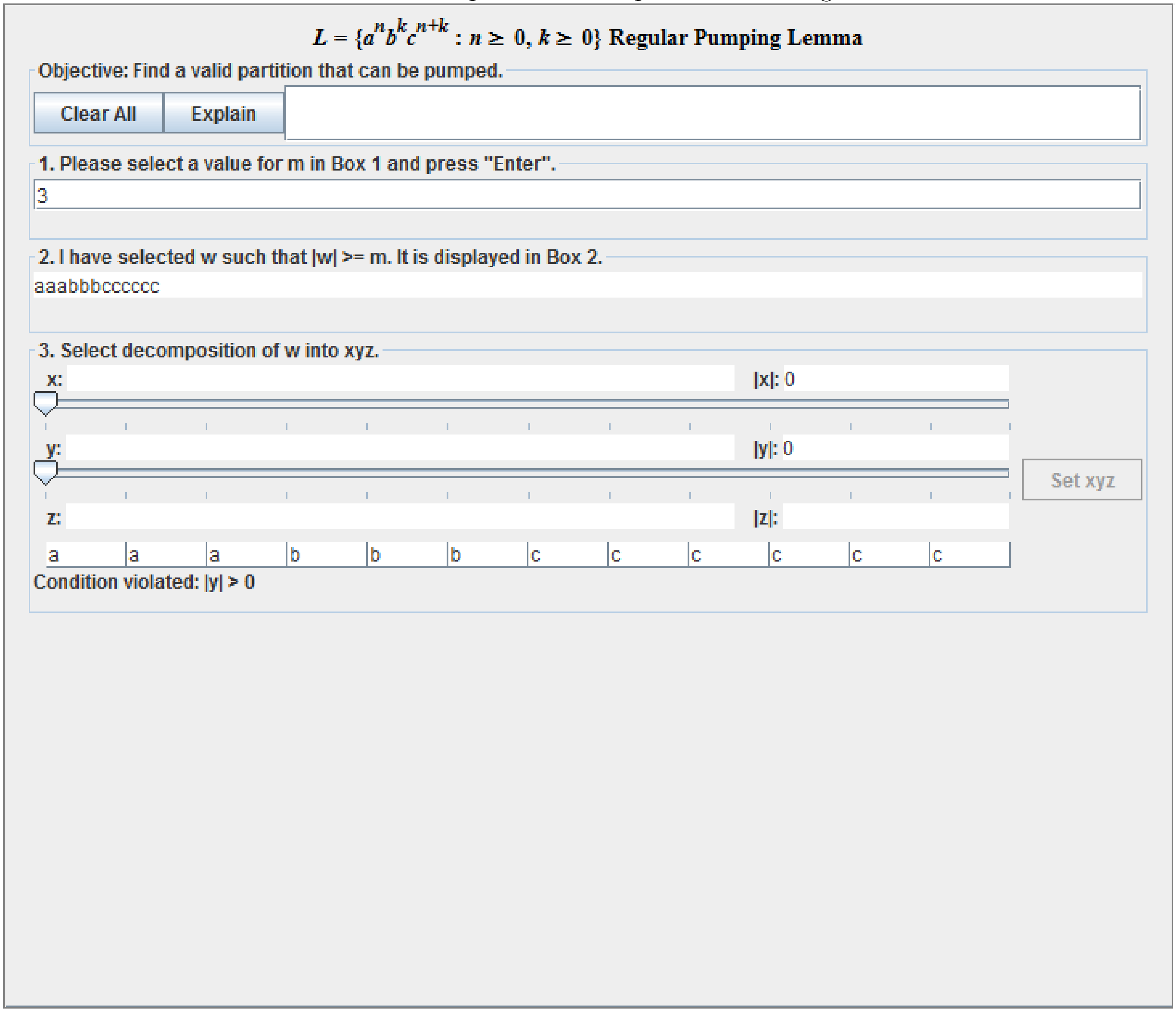
The next step is to come up with a string that cannot be decomposed in accordance with the requirements of the pumping lemma.

If you click 'Explain' in JFLAP you get an explanation of the solution, which we present here, with some slight edits.

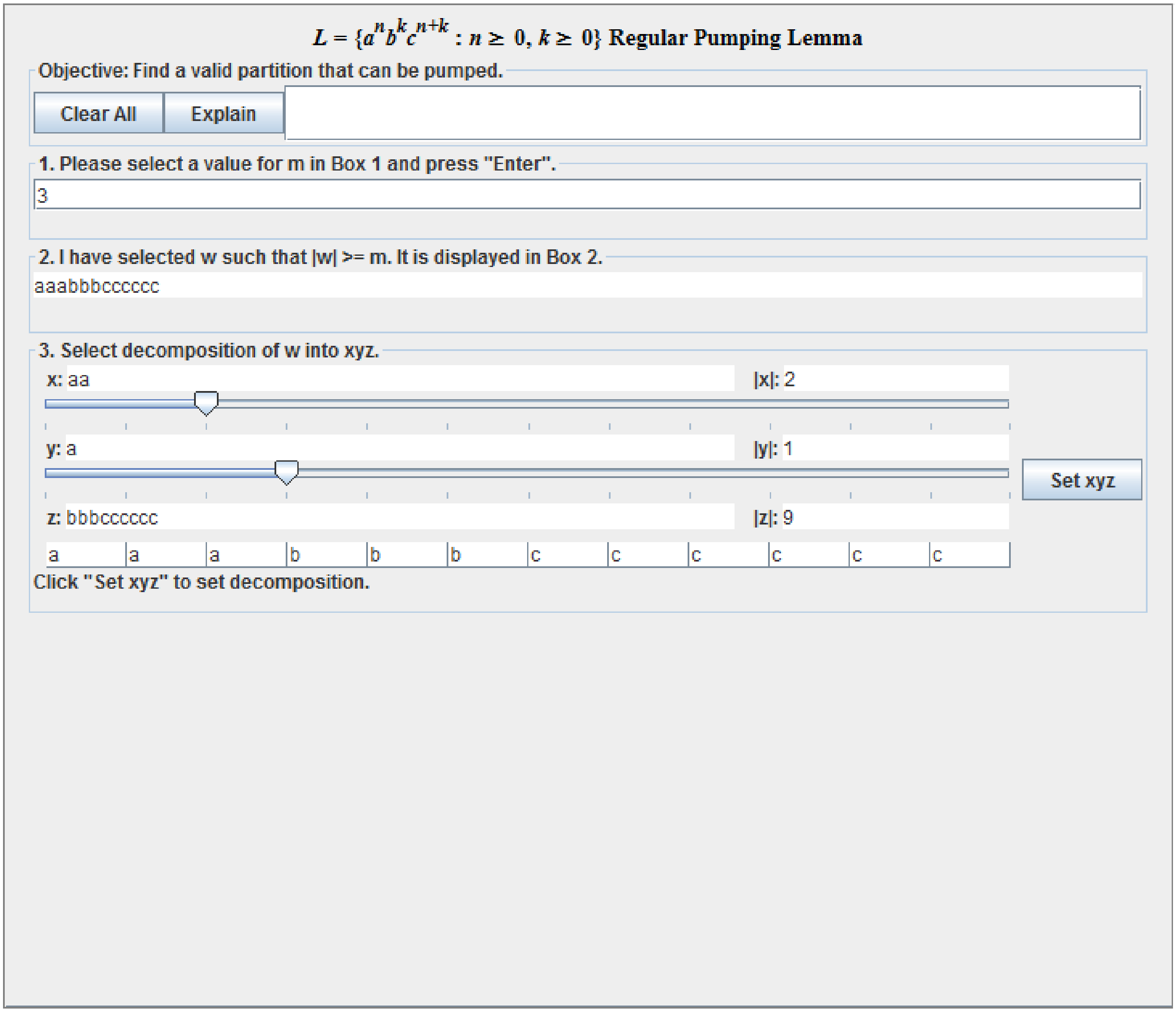
Unfortunately no valid partition of w exists. For any m value, a possible value for w is $a^m b^m c^{2m}$. The y value thus would be a multiple of a. That is, it is some string of as. If $i = 0$ (also called pumping down), the string becomes at most $a^{m-1} b^m c^{2m}$, which is not in the language. Thus, the language is not regular.

To help with the understanding of this proccess, JFLAP allows you to pick whether you or the computer makes the first move.

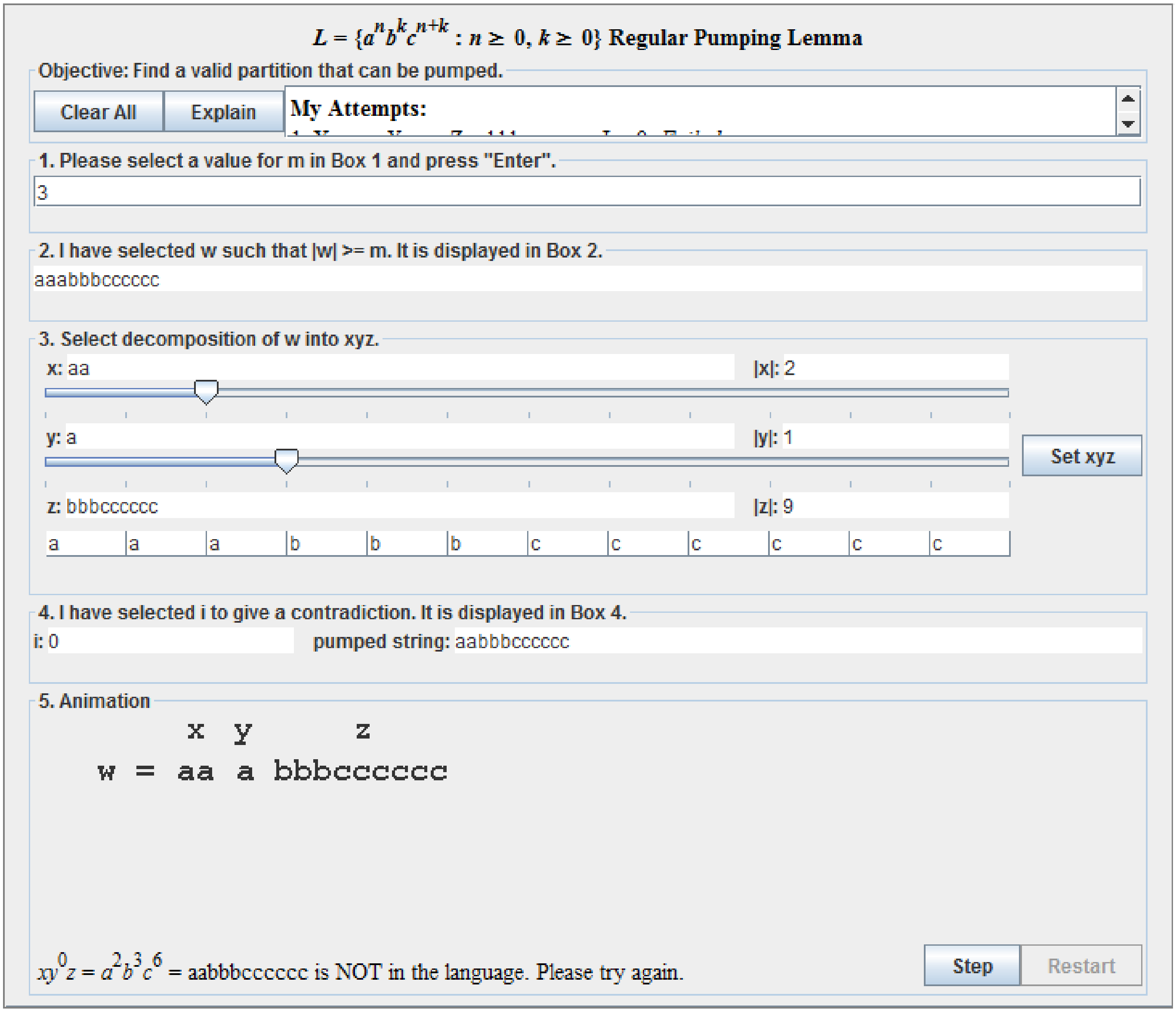
If you pick 'you go first' in this case, you are allowed to enter a value of m. In this case, say we enter a value of 3 and then the computer comes up with the string *aaabbbcccccc*.



Upon being presented with the decomposition screen, we've used the sliders to make $x = aa$ and $y = a$. Note that we cannot make y straddle the boundary between the as and the bs because that would violate the length constraint on xy.



Clicking the 'set xyz' button makes the computer then do the work of producing a contradiction as seen below

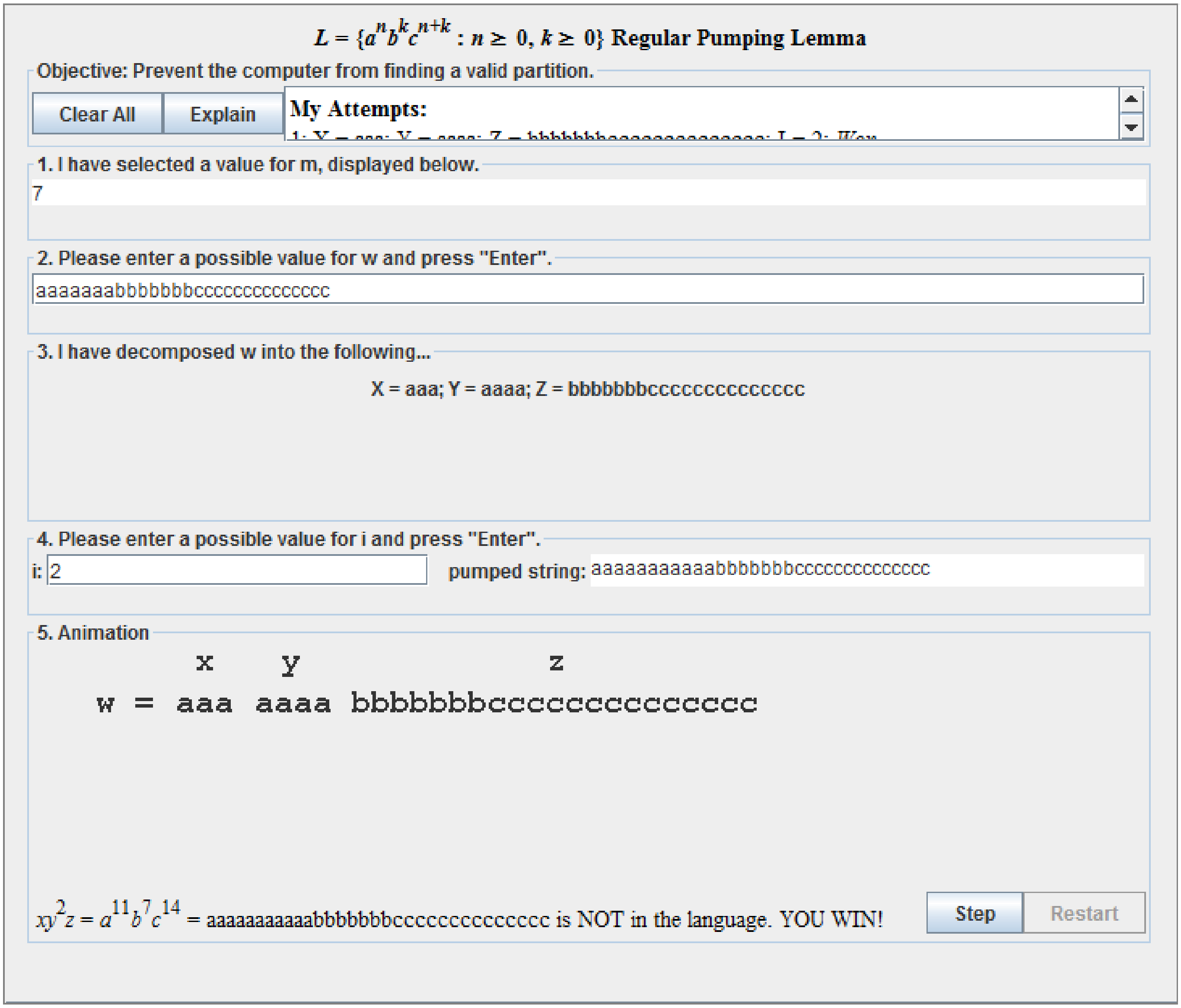


More attempts can be made if needed. Here are the results of 3 attempts

```
My Attempts:
3: X = a; Y = aaa; Z = bbbbcccccccc; I = 2; Failed
2: X = ; Y = a; Z = abbcccc; I = 0; Failed
1: X = aa; Y = a; Z = bbbcccccc; I = 0; Failed
```

If instead we choose to let the computer go first, we are given the job of doing the pumping.

Here is a screenshot showing an instance of that. Note that just for variety, we've chosen to set $i = 2$ and pump the string up.


L = {a^n b^k c^n+k : n ≥ 0, k ≥ 0} Regular Pumping Lemma
Objective: Prevent the computer from finding a valid partition.
Clear All
Explain
My Attempts:
1. I have selected a value for m, displayed below.
7
2. Please enter a possible value for w and press "Enter".
aaaaaaabbbbbbbcccccccccccccc
3. I have decomposed w into the following...
X = aaa; Y = aaaa; Z = bbbbbbbcccccccccccccc
4. Please enter a possible value for i and press "Enter".
i: 2
pumped string: aaaaaaaaaaabbbbbbbcccccccccccccc
5. Animation
x
y
z
w = aaa aaaa bbbbbbbcccccccccccccc
xy^2z = a^11 b^7 c^14 = aaaaaaaaaaabbbbbbbcccccccccccccc is NOT in the language. YOU WIN!
Step
Restart