Recursive Definitions for Languages

Not only can we use regular expressions to define a language, but we can also define languages recursively. Here is an example of such a definition:

1) □ □ L
2) if w □ L, then awb □L
3) A string *w* □L only if it can be obtained from the basis(1) by finite number of applications of the recursive step.

From this definition, it's fairly easy to see that L = {a^nb^n | n□ N}. However, just because we can "see it" doesn't constitute a proof. We will use induction to prove this result for L, and then look at a few other similar examples.

First, we will show that all strings of the form a^nb^n □L. We will use induction on n.

Base case n=0 : a^0b^0 = □□L.

Inductive Hypothesis : Assume for an arbitrary n=k that a^kb^k□L.

Inductive Step : Under this assumption, we must prove for n=k+1 that $a^{k+1}b^{k+1}$□L.

$a^{k+1}b^{k+1}$ = a(a^kb^k)b. (Let w = a^kb^k.)
= awb.

By our inductive hypothesis, w□L. Furthermore, step 2 tells us that if w□L, then awb w□L as well. Thus, we have deduced that $a^{k+1}b^{k+1}$ □n L, as desired.

It remains to be shown that all strings in L fit this form. Assume to the contrary that there exists a string in L NOT of the form a^nb^n. We know that all strings in L (except for the empty string) start with a and end with b. (Why?) Let w be the smallest string in L that is NOT of the form a^nb^n. But, we know that w = axb, for some string x because w must start with a and end with b. By definition, for w to be in L, x must also be in L. BUT, if x is in L, we KNOW that x is NOT of the form a^nb^n either. This contradicts the fact that w was the smallest such string in L. Thus, the string w does not exist, and all strings in L fit the form a^nb^n.

Here's another problem:

Let A = {a,b}
Let L = {w | w∈A*, w either contains no occurrences of symbol b, or if w contains symbol b, then each occurrence of symbol b is followed immediately by at least one occerrences of the symbol a.}

Prove that L is a subset of the language whose regular expression is a*((ba)a*)*.

Let $N_b(w)$ be the number of bs in the string w. We will prove our result using induction on $N_b(w)$.

Base case: $N_b(w) = 0$. This means that the string MUST be of the form a*. Notice that all strings of this form are part of the language described by the regular expression above. (Why?)

Inductive hypothesis. For an arbitrary value of $N_b(w) = k$, prove that w is an element of the language described by a*((ba)a*)*.

Inductive Step: Prove for $N_b(w) = k+1$, that w is an element of the language described by a*((ba)a*)*.

Let w □ L such that $N_b(w) = k+1$. That means that we can partition w = xy where $N_b(x) = k$ and $N_b(y) = 1$. By our inductive hypothesis, we know that x□ L. Furthermore, we can assume that y= baa* since each b must be followed by at least one a, and this is the last b. Thus, we have that w is a subset of the language denoted by the regular expression a*((ba)a*)*baa*. But, using our rules of simplification, we find that w is a subset of the langauage a*((ba)a*)*, thus w IS in L.

Here's another question:

Let A = {a,b} denote an alphabet of two distinct symbols. Define a function g: A* □ A* by the following recursive rules:

1) g(□) = □
2) g(a) = b
3) g(bu) = bg(u) for an u □A*.
4) g(aau) = g(au) for an u □A*.
5) g(abu) = bg(bu) for an u □A*.

Prove by induction that $g(a^nb) = b^2$, for all integers n > 0.

Use induction on n.
Base Case: n=1. g(ab) = g(ab□) = bg(b□), using rule #5
= bbg(□) = bb□ = bb.

Inductive Hypothesis: Assume for an arbitrary n=k, that $g(a^kb) = b^2$.

Inductive Step: Prove for n=k+1 that $g(a^{k+1}b) = b^2$.

$g(a^{k+1}b) = g(aa^kb)$