The Theory of Computation
Bernard M.E. Moret
Addison-Wesley-Longman, 1998

Detailed Solution
Chapter 3, Problem 3.25, Part 2

False. The problems is that, given L and L_2, there typically exist many choices of language L' such that $L = L'L_2$; if L is regular and L_2 is finite, at least one of these choices *must* be regular, but not all of them need be. Since the problem tells us that all three languages are given, an affirmative answer would require us to prove that, for any choice of the three languages such that the equality holds and such that L is regular and L_2 is finite, then L' must be regular—and we can find counterexamples.

To get a regular L', we can run the machine for L until the input runs out, then start guessing a suffix and run it both through M—continuing the computation—and through a machine for L_2—starting from the start state. If both machines accept together at any stage, we accept the input string. Since we cannot really run a machine until the input runs out, we use ε transitions instead. Given a DFA M for L, we can build an NFA M' for L_1 as follows: M' will have a copy of M (with all states made non-accepting) and an ε transition from each state q of M to the start of a new submachine, call it M'_q; this new submachine makes *only* ε transitions and combines M (started in state q) and the machine for L_2, started in its start state, running them effectively in parallel. Each M'_q can only make a finite number of ε transitions because the language L_2 is finite; any state of M'_q in which both the copy of M and the copy of the machine for L_2 accept is an accepting state of M'. Thus, in order to accept an input string, M' must make use of one of its ε transitions (otherwise it stays within its copy of M, which has no accepting state), but it can only make such a transition when it has run out of input, because it will enter a submachine that will not process any input on the way to any accepting state—any further input will cause rejection. Moreover, from this point, both the "continuing copy" of M and the machine for L_2 must run together and accept at the same time, i.e., M' must guess a suffix for its input that belongs to L_2 and that, when concatenated to its input, belongs to L.

As a counterexample (on a one-symbol alphabet, no less), let $L_1 = \{0^k \mid k > 2 \text{ and not a prime}\}$ and let $L_2 = \{\varepsilon, 0\}$. Thus L_1 is not regular and L_2 is finite. Note that $L = L_1 L_2$ now consists of all strings of 0s whose length is either a non-prime or a non-prime plus 1. But that is *all* strings of 0s (of at least 4 characters), since any prime beyond 3 is just some non-prime plus 1. Thus $L = \{0^k \mid k > 3\}$ is regular and we have a counterexample. But note that we could have chosen $L' = L$ and still have had $L = L' L_2$.