

Solution guidelines for Minor 2

1a The infinite union is not regular and it can be easily proved using a counterexample like $L_i = a^i b^i$. Using arguments based on Myhill Nerode theorem is not easy to carry out without specific example since you will not be able to argue about infinite equivalence classes.

1b This is regular and the simplest argument is to construct a product machine with an extra component $(p_i, q_j, \{1, 2\})$ to keep track of which machine is being simulated and toggle between them.

2. There is a four state machine

3a This language is a union of strings where there is at least one pair that is not equal hence it is a CFL.

3b The reverse of a language is CFL and can be argued as follows. Assume that the given grammar is given in CNF. Then by reversing the RHS of all productions $A \rightarrow BC$ to $A \rightarrow CB$, we obtain the required grammar. Rename the variables as $A', B'C'$ etc. The proof is by induction on the number of derivation steps (for all variables) by noting that if $B \xrightarrow{*} w_1$ and $C \xrightarrow{*} w_2$ then $B' \xrightarrow{*} w_1^R$ $C' \xrightarrow{*} w_2^R$ then $A' \rightarrow C'B' \xrightarrow{*} w_2^R w_1^R = (w_1 w_2)^R$.

4. The problem is decidable. A *clean* proof of this must unambiguously establish a bound on the running time in terms of some concrete parameter. It doesn't suffice to say that *It will terminate sometime* since we are not arguing about r.e. languages but recursive languages.

The key ingredients are the CFL recognition algorithm, i.e., the dynamic programming based CYK algorithm and the pumping lemma. We know that there is some n (based on the grammar) such that the language L is infinite if there exists some string $w \in L$ such that $2n > |w| > n$. This can be easily tested by exhaustively trying all strings upto length $2n$ if they belong to L . If L is finite and L contains exactly 100 strings then we accept else not. Moreover n can be calculated as the number of variables in the CNF.

If you don't invoke CYK and PL, then you have to prove it from scratch. For example, if you are trying to generate all possible strings from the given grammar and arguing that the language is infinite if a variable repeats along some path, you have to discuss why it implies infinite number of strings - we used it in the proof of PL for CNF, it may not be true for an arbitrary grammar with useless symbols. Moreover, the method for generating all possible parse trees must be described formally as well as it must be argued till what level, it must be done to guarantee termination. A terminating condition like *till 100 strings are generated* is fallacious since it may go on for ever when the grammar has less than 100 strings.