5.4.2.a. NO.

Short answer: Suppose such a Turing machine existed. Then suppose we have a machine M and input w. Make a machine that modifies the input M so that all halt states in M transition to a new state q. Then use the machine suggested here to determine if this modified M reaches state q. This would solve the halting problem.

Long answer:

> **Proof.** *We will prove that this problem is undecidable by reducing the halting problem to it.*
> *Suppose there exists a machine M_1 that decides the language of Turing machine, state, string triples (M, q, w) such that M reaches state q when given input w.*

Long answer/proof for **5.4.2.a**, *continued*

Let M_2 be the Turing machine that operates as follows: When given the description of a machine M and input w, M_2 constructs the description of a machine M' such that M' is like M except that it has one more state q, and all the transitions in M that would move to a halting state are changed so that they now transition to q. Then M_2 acts like M_1 on the description of M', q, and w.

Note that by how we defined M_2, it must be that M_2 accepts M, w if and only if M_1 accepts M', q, w.

Further, M_2 decides the halting problem: Suppose a machine M halts on input w. Then the machine M' that M_2 constructs will reach state q on input w, and so M_1 and therefore M_2 will accept it. Next suppose M does not halt on input w. Then the machine M' will never reach state q, and so M_1 and therefore M_2 will reject it.

Since it is impossible for a machine to decide the halting problem, M_2 cannot exist, and therefore M_1 cannot exist. Thus this problem is undecidable. $\square$

5.4.2.b. NO.

Short answer: If we had such a machine we could use it to decide the problem in part a by setting p to the start state.

Long answer:

Proof. *We will prove that this problem is undecidable by reducing the problem in part a to it.*
Suppose there exists a machine M_1 that decides the language of Turing machine, state, state (M, p, q) triples such that there is a configuration with with state p that yields a configuration with state q.

Long answer/proof for **5.4.2.b**, *continued*

Let M_2 be the Turing machine that operates as follows: When given the description of a machine M, a state q, and a string w, M_2 constructs the description of a machine M' such that M' is like M except that it has a new start state s. (Let s_0 be the start state of M.) When M' is in state s, it erases whatever is on its tape and writes w in its place. Then it moves its head to the beginning and transitions to state s_0; from then on, M' operates like M. After constructing M', M_2 also adds the description of s and q on the tape and then acts like M_1 does on its input; in other words, it gives (M', s, q) as input to M_1.

Note that by how we defined M_2, it must be that M_2 accepts (M, q) if and only if M_1 accepts (M', s, q).

Further, M_2 solves the problem described in part a: Suppose a machine M reaches state q starting with string w. Then the machine M' that M_2 constructs will reach q from state s. Next suppose a machine M never reaches state q starting with string w. Then the machine M' that M_2 constructs will never reach q from state s.

Since it is impossible for a machine to decide the problem in part a, M_2 cannot exist, and therefore M_1 cannot exist. Thus this problem is undecidable. $\square$