

APPENDIX A  
PROOFS

*Proof of Lemma 1.* For each clause  $c_i \in R$ , we introduce an auxiliary variable  $y_i$ . For each literal  $l \in c'_i$ , we add the assertion  $\neg l \wedge y_i$  to the solver. Let's assume  $c_i = l_1 \vee l_2 \vee \dots \vee l_k$ . We add assertions  $\neg l'_1 \vee y_i, \neg l'_2 \vee y_i, \dots, \neg l'_k \vee y_i$  to the solver. Therefore, the overall assertion for clause  $c_i$  added is  $(\neg l'_1 \vee y_i) \wedge (\neg l'_2 \vee y_i) \wedge \dots \wedge (\neg l'_k \vee y_i)$ . Now

$$\begin{aligned} & (\neg l'_1 \vee y_i) \wedge (\neg l'_2 \vee y_i) \wedge \dots \wedge (\neg l'_k \vee y_i) \\ \Leftrightarrow & (\neg l'_1 \wedge \neg l'_2 \wedge \dots \wedge \neg l'_k) \vee y_i \\ \Leftrightarrow & \neg(l'_1 \vee l'_2 \vee \dots \vee l'_k) \vee y_i \\ \Leftrightarrow & \neg c'_i \vee y_i \\ \Leftrightarrow & c'_i \Rightarrow y_i \end{aligned}$$

Therefore, the operation performed in lines 1–4 of FINDCLAUSES is equivalent to adding the assertion  $c'_i \Rightarrow y_i$  for each clause  $c_i \in R$ .

Initially, the set of violating clauses  $\mathcal{G}$  is empty. For the sake of argument, let's assume  $R$  contains only one clause  $c_1$ . We claim the following:

**Claim.** Upon termination,  $\mathcal{G} = \emptyset$  or  $\mathcal{G} = \{c_1\}$ .

*Proof.* If  $c_1 = l_1 \vee l_2 \vee \dots \vee l_k$ , then the assertions added to the solver are  $\neg l'_1 \vee y_1, \neg l'_2 \vee y_1, \dots, \neg l'_k \vee y_1$ . Moreover, the SAT query of line 6 adds the assertions  $S \wedge \delta$ , and  $\neg y_1$ . Combined, these assertions are equivalent to  $(S \wedge \delta \wedge \neg y_1 \wedge \neg c'_1)$  or  $(S \wedge \delta \wedge \neg y_1 \wedge \neg R')$ . There are two cases to consider. If the assertion is

- 1) UNSAT: The post-image of all states in  $S$  is in  $R'$ , and  $c_1$  is not a violating clause. Therefore,  $\mathcal{G} = \emptyset$ .
- 2) SAT: We know that the SAT model for  $S \wedge \delta \wedge \neg y_1 \wedge \neg R'$  is a pair of states  $(a, b')$  such that  $a \in S$ ,  $(a, b') \in \delta$ , but  $b' \notin R'$ , and an assignment to  $y_1$ . Since  $R$  contains only one clause,  $b' \notin R'$  if and only if  $b' \notin c'_1$ . In other words, none of the literals in  $c'_1$  match the literal assignments in state  $b'$ . Therefore,  $\neg l'_1, \neg l'_2, \dots, \neg l'_k$  are true, which makes  $\neg c'_1$  true. The only possible assignment to  $y_1$  is false. Therefore, since  $c_1$  is a violating clause, the corresponding auxiliary variable is assigned false. Clause  $c_1$  is added to  $\mathcal{G}$  in lines 9–10, and the query in line 6 is updated.

Therefore, upon termination  $\mathcal{G} = \emptyset$  or  $\mathcal{G} = \{c_1\}$  if  $S \wedge \delta \wedge \neg R'$  is UNSAT and SAT, respectively.

The above argument for  $R$  containing only one violating clause can be extended to multiple clauses. If a state  $b'$  in the SAT model is missing from multiple clauses in  $R$ , their corresponding auxiliary variables get assigned to false, and all such clauses are added to  $\mathcal{G}$  and the query updated. On every iteration of the loop in lines 6–10, a new state pair is found until all violating clauses have been removed from  $R$  and added to  $\mathcal{G}$ . Therefore, since input to FINDCLAUSES is  $S = S_i, \delta$ , and  $R = R_{i+1}$ , upon termination,  $\mathcal{G}$  contains all violating clauses in  $R_{i+1}$ . □

*Proof of Lemma 2.* In line 3,  $B$  contains all primed variables not in clause  $c'$ . Initially,  $\hat{c}' = c'$ . The SAT model of the query  $S \wedge \delta \wedge \neg \hat{c}'$  is pair of states  $(a, b')$  such that  $a \in S$ ,  $(a, b') \in \delta$ , but  $b' \notin \hat{c}'$ . We know that  $b' \notin \hat{c}'$  if none of the literals in  $\hat{c}'$  match the literal assignments in state  $b'$ . If we pick a literal assignment in  $b$  and add it to  $\hat{c}'$ , then  $b \in \hat{c}$ . The added variable corresponding to the added literal is removed from  $B$  and the loop repeated. On every iteration of the algorithm, multiple states are added to  $\hat{c}'$ . The loop terminates when  $S \wedge \delta \wedge \neg \hat{c}'$  is UNSAT. However, if enough literals have been added such that  $B$  becomes empty and  $S \wedge \delta \wedge \neg \hat{c}'$  is still SAT, then  $\hat{c} = true$  (all states are reachable from  $S$ ). □

*Proof of Lemma 3.* In line 1,  $v$  contains the difference of literals between  $c$  and  $\hat{c}$ , i.e., all literals that are added to  $c$  such that  $S \wedge \delta \wedge \neg \hat{c}'$  is UNSAT. On every iteration of the loop in lines 2–5, we pick a literal  $l$  to drop from  $\hat{c}$ . If  $S \wedge \delta \wedge \neg c' \wedge \neg g'$  is SAT, where  $g = v \setminus l$ , then  $l$  is a required literal and we try dropping another literal. If  $S \wedge \delta \wedge \neg c' \wedge \neg g'$  is UNSAT, we extract the unsat core of the assumption literals. The unsat core is not necessarily minimal.  $v$  is made equal to the unsat assumption literals and the loop repeats. Upon termination,  $v$  contains the minimum literals that when added to  $c'$  to give  $\hat{c}'$  are enough to ensure that  $S \wedge \delta \wedge \hat{c}'$  is valid. □

*Proof of Theorem 1.* The proof for Theorem 1 follows directly from Lemmas 1, 2 and 3. All violating clauses found by FINDCLAUSES are repaired using SHRINKCLAUSE and EXPANDCLAUSE. Therefore, upon termination, FRAMEREPAIR returns repaired frame  $\hat{R}_{i+1}$  such that  $S_i \wedge \delta \Rightarrow \hat{R}'_{i+1}$ . □