A Proof of theorem

Theorem 18. For *PG* P having a correctable error path $\pi \in Paths(TS(P))$, $\pi \notin Paths(TS(\text{PTF}_\pi(P)))$.

Proof. For PG $P = (Loc, Act, \text{Effect}, CR, l_0, g_0)$ defined over variables Var, let $\pi = \ldots \langle l_{i-1}, \eta_{i-1}\rangle \xrightarrow{a_{i-1}} \langle l_i, \eta_i\rangle \xrightarrow{a_i} \langle l_{i+1}, \eta_{i+1}\rangle \ldots$ be an error path having erroneous transition $\mathbf{e} - \text{trans}(\pi) = (\langle\ l_i, \eta_i\rangle, a_i, \langle\ l_{i+1}, \eta_{i+1}\rangle)$ in $TS(P)$. Let $\text{PTF}(P) = P' = (Loc, Act, \text{Effect}, CR', l_0, g_0)$ be defined over variables Var with $CR' = CR \setminus \{(l_i, g, a_i, l_{i+1}) \mid \eta_i \models g\} \bigcup \{(l_i, g \wedge \neg\hat{g}, a_i, l_{i+1}) \mid \eta_i \models g\}$ where $\hat{g} := \bigwedge_{x \in Var} x = \eta_i(x)$.

Let $T' = TS(P')$.

We prove by contradiction that $\pi \notin Paths(T')$. Assume $\pi \in Paths(T')$. This implies the erroneous transition of π, $\mathbf{e} - \text{trans}(\pi) = \langle l_i, \eta_i\rangle \xrightarrow{a_i} \langle l_{i+1}, \eta_{i+1}\rangle$ which can be generated by some of the conditional transitions in PG P'. Let $(l_i, g, a_i, l_{i+1}) \in CR'$ be a conditional transition that generates the erroneous transition $(\mathbf{e} - \text{trans}(\pi) = \langle l_i, \eta_i\rangle \xrightarrow{a_i} \langle l_{i+1}, \eta_{i+1}\rangle)$. This implies that $\eta_i \models g$. However, by Def. 7, the guard g of the conditional transition $(l_i, g, a_i, l_{i+1}) \in CR'$ is a conjunction of two sub-conditionals, i.e. $g = h \wedge \neg\hat{g}$, where $\hat{g} = \bigwedge_{x \in Var} x = \eta_i(x)$. Therefore, $\eta_i \not\models g$. Hence, contradiction.

B An example of mutual exclusion problem

The mutual exclusion problem is a classic example used in concurrent programming theory. The aim is to protect a critical resource from simultaneous access by two processes. The example in Fig. 8a is an incomplete implementation of Peterson's solution [22]. For the solution to work, the guard $(flag[1 - pno] == 0 || turn == pno)$ should be placed before entrance to the critical section. The mutual exclusion property expressed at the end specifies that the number of processes in the critical section must be inclusively between 0 and 1. Protictor succeeds in correcting the model in two iterations, shown in Fig. 8b and 8c. In the first iteration 8b, it prevents user 0 from entering the critical section if user 1 is already present. In the next iteration, it does the same for user 1. At this point, the model adheres completely to the mutual exclusion policy.

C Summary of models corrected by the tool

Table 3 provides some information about the models corrected by Protictor. They can all be found in the repository

Table 3. Models corrected by the developed tool.

Model	Number of Iterations	Number of Variables in Guard	Scenarios Encountered	Code Size Increase (Bytes)	Code Size Original (Bytes)
assertion-mutex	33	4	1, 2	1612	530
atomic	2	4	1, 2	167	669
bakery	10	4	1, 2	857	1653
breaks	1	3	2	39	198
deadlock	2	2	2	72	563
deadlock-np	2	2	2	72	511
dining-philosophers	3	5	2	233	1474
division	1	8	1	124	670
drinks	3	4	1, 2	314	1076
elevator-planning	5	26	2	2404	3159
elses	3	2	1, 2	121	128
ends	1	4	1	60	312
ex-3a-p	3	4	2	171	843
gcd	1	2	2	30	183
gotos	2	3	1, 2	139	257
inc	3	2	1, 2	144	280
manna_pnueli	10	3	1, 2	537	438
msmie	48	6	2	4317	5985
mutual-exclusion	2	4	1, 2	158	272
p320	4	5	1, 2	267	395
paper-example-1	2	1	1	88	538
paper-example-2	2	1	1	88	489
pathfinder	2	3	1, 2	161	1440
petersonN	1	11	2	183	881
philosophers	11	3	1, 2	631	515
producer-consumer	1	1	1	49	283
relay	7	10	1	2226	840
protocol	4	2	1, 2	213	1293
skips	1	4	2	47	228
toy-maker	1	1	1	51	399
toy-maker-complex	4	2	1, 2	200	487
toy-mchan	1	5	2	95	753
traffic-lights	1	1	2	25	328
welfare	4	21	2	1080	1048

```
1  bool turn, flag[2];
2  byte ncrit;
3
4  init
5  {
6    run user(0)
7    run user(1)
8  }
9
10
11 proctype user(byte pno)
12 {
13   do
14   :: flag[pno] = 1;
15      turn = 1 - pno;
16      ncrit++;
17      /* critical section */
18      ncrit--;
19
20      flag[pno] = 0;
21   od
22 }
23
24 ltl p0  {[]((ncrit >= 0) &&
25     (ncrit <= 1))}
```

(a) Original code

```
13 do
14 :: flag[pno] = 1;
15    turn = 1 - pno;
16    atomic
17    {
18      !(flag[0] == 1 &&
19      flag[1] == 1 &&
20      ncrit == 1 &&
21      turn == 0)
22      ncrit++;
23    }
24    /* critical section */
25    ncrit--;
26
27    flag[pno] = 0;
28 od
```

(b) After one iteration

```
16    atomic
17    {
18      !(flag[0] == 1 &&
19      flag[1] == 1 &&
20      ncrit == 1 &&
21      turn == 0) &&
22      !(flag[0] == 1 &&
23      flag[1] == 1 &&
24      ncrit == 1 &&
25      turn == 1)
26      ncrit++;
27    }
```

(c) After two iterations (corrected)

Fig. 8. Mutual Exclusion.