

Санкт-Петербургский государственный политехнический университет

Отчет по курсовой работе

по курсу «Верификация программ»

«Разработка контроллера светофоров на перекрёстке и его верификация»

Студент:	Руцкий В. В.
Группа:	5057/2
Вариант:	9, 13
Преподаватель:	Шошмина И. В.

Санкт-Петербург 2010

Содержание

1	Постановка задачи	2
2	Решение	2
2.1	Моделирование внешней среды	2
2.2	Моделирование светофоров	3
2.3	Моделирование пересечений полос движения	5
3	Верификация алгоритма	6
4	Результат работы	7
A	Исходный код	8

1 Постановка задачи

Задан автомобильный перекрёсток, его конфигурация показана на рис. 1 (вариант 9, 13). На

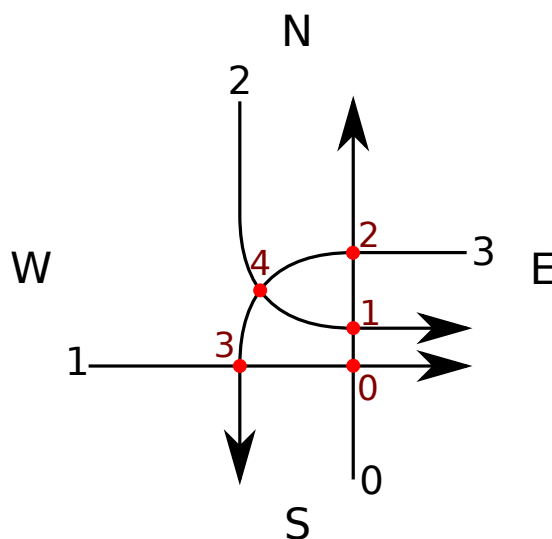


Рис. 1: Схема перекрёстка

перекрёстке возможны следующие направления движения транспорта:

- $S \rightarrow N$,
- $W \rightarrow E$,
- $N \rightarrow E$,
- $E \rightarrow S$.

Каждое направление движения регулируется собственным светофором. На каждом направлении движения установлены датчики, фиксирующие наличие автомобилей.

При отсутствии автомобилей на полосе, соответствующий светофор должен гореть красным светом. При появлении машин на полосе, соответствующий светофор должен убедиться, что по всем полосам, пересекающим текущую, движение транспорта запрещено (их светофоры горят красным), и загореться зелёным светом, пропуская машины. При наличии машин на различных полосах, светофоры должны периодически переключаться, обеспечивая равномерный пропуск машин по всем направлениям.

Требуется разработать модель управления данным светофором на языке *Promela*, удовлетворяющую указанным выше требованиям, и верифицировать её корректность с помощью пакета *Spin*¹.

2 Решение

2.1 Моделирование внешней среды

Внешняя среда в данной задаче моделируется процессами *LineTrafficGenerator*, см. код 1 (полный текст исходных кодов приведён в приложении А). В качестве датчиков, фиксирующих присутствие автомобилей, выступает массив каналов *carsWaiting* — по одному каналу на каждый светофор.

Автомобили генерируются недетерминированно. Для каждого направления создаётся новый процесс *LineTrafficGenerator*, отвечающий за генерацию машин, движущихся через соответствующий светофор.

¹<http://www.spinroot.com/>

Listing 1: Моделирование внешней среды

```

53  /* Car object */
54  mtype = { CAR };
55
56  /* Cars waiting sign for each traffic light */
57  chan carsWaiting[N_TRAFFIC_LIGHTS] = [1] of { mtype };
58
59  proctype LineTrafficGenerator( byte initTlId )
60  {
61      byte tlId;
62
63      tlId = initTlId;
64
65      do
66          :: carsWaiting[tlId] ! CAR;
67      od
68  }

```

2.2 Моделирование светофоров

Каждый светофор моделируется процессом *TrafficLight*, см. код 2.

При своей работе светофор хранит своё состояние (цвет) в глобальном массиве *tlColor* (строка 115).

Светофор зависит от общих ресурсов — пересечений полос движения: чтобы безопасным образом включить зелёный свет на светофоре, необходимо, чтобы на всех светофорах, пути которых пересекают путь данного светофора, горел красный свет, т.е. необходимо единолично захватить ресурсы — пересечения полос, лежащие на пути данного светофора.

Алгоритм работы процесса светофора следующий. При обнаружении машин на полосе светофора, светофор захватывает лежащие на его пути пересечения полос в порядке увеличения их номеров (захват в такой последовательности обеспечивает отсутствие дедлоков, возникающих из-за циклического захвата ресурсов).

После захвата перекрёстков, светофор переключается в состояние зелёного света и пропускает ограниченный поток машин (представленный сообщением *CAR*).

Далее светофор переходит в состояние красного света и возвращает захваченные ранее ресурсы.

Listing 2: Моделирование светофоров

```

111 /* Traffic lights states */
112 mtype = { RED, GREEN };
113
114 /* Traffic light state */
115 mtype tlColor[N_TRAFFIC_LIGHTS];
116
117 /* Main traffic light process */
118 proctype TrafficLight( byte initTlId )
119 {
120     byte tlId;
121
122     tlId = initTlId;
123
124     assert(tlColor[tlId] == RED);
125
126 endTL:
127     do
128         :: carsWaiting[tlId] ? [CAR] ->
129         /* Cars in queue */

```

```

130
131 /* Lock dependent intersections */
132 if
133   :: tId == SN ->
134     lockIntersection(0, tId);
135     lockIntersection(1, tId);
136     lockIntersection(2, tId);
137   :: tId == WE ->
138     lockIntersection(0, tId);
139     lockIntersection(3, tId);
140   :: tId == ES ->
141     lockIntersection(2, tId);
142     lockIntersection(3, tId);
143     lockIntersection(4, tId);
144   :: tId == NE ->
145     lockIntersection(1, tId);
146     lockIntersection(4, tId);
147 fi;
148
149 /* Allow passing */
150 progressPassCar:
151   atomic
152   {
153     printf("MSC:_Traffic_light_#%d:_GREEN\n", tId);
154     tlColor[tId] = GREEN;
155
156     /* Pass car */
157     /* Note: atomic for easier claim construction */
158     carsWaiting[tId] ? CAR;
159     printf("MSC:_Traffix_light_#%d:_pass_cars\n", tId);
160   };
161
162 /* Forbid passing */
163 atomic
164 {
165   printf("MSC:_Traffic_light_#%d:_RED\n", tId);
166   tlColor[tId] = RED;
167 };
168
169 /* Release dependent intersections */
170 if
171   :: tId == SN ->
172     unlockIntersection(2);
173     unlockIntersection(1);
174     unlockIntersection(0);
175   :: tId == WE ->
176     unlockIntersection(3);
177     unlockIntersection(0);
178   :: tId == ES ->
179     unlockIntersection(4);
180     unlockIntersection(3);
181     unlockIntersection(2);
182   :: tId == NE ->
183     unlockIntersection(4);
184     unlockIntersection(1);
185 fi;
186 od;

```

2.3 Моделирование пересечений полос движения

Пересечение полос движений моделируется процессом *Intersection*, см. код 3. Каждому пересечению соответствует один процесс.

Захват и высвобождение пересечений реализовано в функциях-макросах *lockIntersection* и *unlockIntersection*, и работают в соответствии со следующим алгоритмом.

Для захвата пересечения в канал *intersectionLockRequests* посылается запрос *LOCK*. Запросы из этого канала последовательно обрабатываются процессом *Intersection*: на каждый запрос отправляется подтверждение *INT* в канал *intersectionLockGranted*, которое означает, что ресурс единолично предоставлен запросившему его светофору. После окончания работы с ресурсом, светофор высвобождает ресурс посылая сообщение *RELEASE* процессу *Intersection*, выдавшему ресурс. При получении сообщения *RELEASE* процесс *Intersection* переходит в исходное состояние.

Последовательная обработка поступающих сообщений решает проблему голодания процессов, при условии слабой справедливости (*Weak Fairness*) в работе модели.

Listing 3: Моделирование пересечений полос движения

```

70 /* Manager messages */
71 mtype = { LOCK, INT, RELEASE };
72
73 /* Intersections lock/release requests queue.
74 * Message contains requestee traffic light identifier.
75 */
76 chan intersectionLockRequests[N_INTERSECTIONS] =
77   [N_TRAFFIC_LIGHTS] of { mtype, byte };
78 chan intersectionLockGranted[N_TRAFFIC_LIGHTS] =
79   [0] of { mtype };
80 chan intersectionReleaseRequests[N_INTERSECTIONS] =
81   [0] of { mtype };
82
83 /* Macro for obtaining intersection resource */
84 #define lockIntersection( intId, tId ) \
85   intersectionLockRequests[intId] ! LOCK(tId); \
86   intersectionLockGranted[tId] ? INT
87
88 /* Macro for releasing intersection resource */
89 #define unlockIntersection( intId ) \
90   intersectionReleaseRequests[intId] ! RELEASE
91
92 /* Intersection resource manager */
93 proctype Intersection( byte initIntId )
94 {
95   byte intId, tId;
96
97   intId = initIntId;
98
99   endInt:
100   do
101     :: intersectionLockRequests[intId] ? LOCK(tId) ->
102       /* Handle request */
103       intersectionLockGranted[tId] ! INT;
104
105       /* Wait for release */
106       progressGiveIntersection:
107         intersectionReleaseRequests[intId] ? RELEASE;
108   od;

```

3 Верификация алгоритма

Была проведена верификация алгоритма по следующим критериям:

1. Безопасность: на светофорах с пересекающимися путями никогда не должен одновременно гореть зелёный свет. Данное условие проверялось следующей LTL формулой, представленной в строках 264–268 кода 4.
2. Живость: всегда, при появлении машин на светофоре, светофор когда-нибудь обязательно загорится зелёным светом. Данное условие проверялось формулами в строках 274–277 кода 4.
3. Справедливость: никакой поток машин не должен бесконечно стоять на светофоре. Данное условие было проверено средствами *Spin*, при условии слабой справедливости в работе модели.

Listing 4: Предикаты для верификации алгоритма

```

239 /*
240  * Correctness requirements.
241  */
242
243 /* Car crash accident definition */
244 #define accident_01 (tlColor[0] == GREEN && tlColor[1] == GREEN)
245 #define accident_02 (tlColor[0] == GREEN && tlColor[2] == GREEN)
246 #define accident_03 (tlColor[0] == GREEN && tlColor[3] == GREEN)
247 #define accident_13 (tlColor[1] == GREEN && tlColor[3] == GREEN)
248 #define accident_23 (tlColor[2] == GREEN && tlColor[3] == GREEN)
249
250 /* Car waiting at traffic light definition */
251 #define car_waiting_0 (len(carsWaiting[0]) > 0)
252 #define car_waiting_1 (len(carsWaiting[1]) > 0)
253 #define car_waiting_2 (len(carsWaiting[2]) > 0)
254 #define car_waiting_3 (len(carsWaiting[3]) > 0)
255
256 /* Traffic light is green definition */
257 #define tl_green_0 (tlColor[0] == GREEN)
258 #define tl_green_1 (tlColor[1] == GREEN)
259 #define tl_green_2 (tlColor[2] == GREEN)
260 #define tl_green_3 (tlColor[3] == GREEN)
261
262 /* Safety: Intersecting roads traffic light both never has GREEN state */
263 /*
264  * [] (!accident_01)
265  * [] (!accident_02)
266  * [] (!accident_03)
267  * [] (!accident_13)
268  * [] (!accident_23)
269  */
270
271 /* Liveness: If cars wait on traffic light, then in future traffic light
272  * became GREEN */
273 /*
274  * [] (car_waiting_0 -> <> tl_green_0)
275  * [] (car_waiting_1 -> <> tl_green_1)
276  * [] (car_waiting_2 -> <> tl_green_2)
277  * [] (car_waiting_3 -> <> tl_green_3)
278  */

```

4 Результат работы

В результате данной работы были изучены материалы, представленные в [1] и [2], построена и верифицирована модель контроллера светофоров перекрёстка, получен практический опыт работы с системой автоматической верификации *Spin*.

A Исходный код

Listing 5: Исходный код

```

1  /*
2  *   This file is part of traffic lights model development and verification.
3  *
4  *   Copyright (C) 2010   Vladimir Rutsky <altsysrq@gmail.com>
5  *
6  *   This program is free software: you can redistribute it and/or modify
7  *   it under the terms of the GNU General Public License as published by
8  *   the Free Software Foundation, either version 3 of the License, or
9  *   (at your option) any later version.
10 *
11 *   This program is distributed in the hope that it will be useful,
12 *   but WITHOUT ANY WARRANTY; without even the implied warranty of
13 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
14 *   GNU General Public License for more details.
15 *
16 *   You should have received a copy of the GNU General Public License
17 *   along with this program. If not, see <http://www.gnu.org/licenses/>.
18 */
19
20 /* Variant 9–13:
21 * SN, WE, NE, ES.
22 */
23 /*
24 *
25 *           N
26 *           2
27 *           /      ^
28 *          /      /
29 *         /      / 2-----3
30 *        /      /  |
31 *       /      / 4  |
32 *      /      /    |
33 *     /      /    |
34 *    /      /    |
35 *   /      /    |
36 *  /      /    |
37 * /      /    |
38 *      /      |
39 *     /      |
40 *    /      |
41 *   /      |
42 *  /      |
43 * /      |
44 *      |
45 *     |
46 *    |
47 *   |
48 *  |
49 * /
50 *
51 *           S
52 *
53 *           0
54 *           |
55 *          |
56 *         |
57 *        |
58 *       |
59 *      |
60 *     |
61 *    |
62 *   |
63 *  |
64 * /
65 *
66 *           1
67 *           |
68 *          |
69 *         |
70 *        |
71 *       |
72 *      |
73 *     |
74 *    |
75 *   |
76 *  |
77 * /
78 *
79 *           3
80 *           |
81 *          |
82 *         |
83 *        |
84 *       |
85 *      |
86 *     |
87 *    |
88 *   |
89 *  |
90 * /
91 *
92 *           4
93 *           |
94 *          |
95 *         |
96 *        |
97 *       |
98 *      |
99 *     |
100 *    |
101 *   |
102 *  |
103 * /
104 *
105 *           1
106 *           |
107 *          |
108 *         |
109 *        |
110 *       |
111 *      |
112 *     |
113 *    |
114 *   |
115 *  |
116 * /
117 *
118 *           3
119 *           |
120 *          |
121 *         |
122 *        |
123 *       |
124 *      |
125 *     |
126 *    |
127 *   |
128 *  |
129 * /
130 *
131 *           0
132 *           |
133 *          |
134 *         |
135 *        |
136 *       |
137 *      |
138 *     |
139 *    |
140 *   |
141 *  |
142 * /
143 *
144 *           1
145 *           |
146 *          |
147 *         |
148 *        |
149 *       |
150 *      |
151 *     |
152 *    |
153 *   |
154 *  |
155 * /
156 *
157 *           3
158 *           |
159 *          |
160 *         |
161 *        |
162 *       |
163 *      |
164 *     |
165 *    |
166 *   |
167 *  |
168 * /
169 *
170 *           0
171 *           |
172 *          |
173 *         |
174 *        |
175 *       |
176 *      |
177 *     |
178 *    |
179 *   |
180 *  |
181 * /
182 *
183 *           1
184 *           |
185 *          |
186 *         |
187 *        |
188 *       |
189 *      |
190 *     |
191 *    |
192 *   |
193 *  |
194 * /
195 *
196 *           3
197 *           |
198 *          |
199 *         |
200 *        |
201 *       |
202 *      |
203 *     |
204 *    |
205 *   |
206 *  |
207 * /
208 *
209 *           0
210 *           |
211 *          |
212 *         |
213 *        |
214 *       |
215 *      |
216 *     |
217 *    |
218 *   |
219 *  |
220 * /
221 *
222 *           1
223 *           |
224 *          |
225 *         |
226 *        |
227 *       |
228 *      |
229 *     |
230 *    |
231 *   |
232 *  |
233 * /
234 *
235 *           3
236 *           |
237 *          |
238 *         |
239 *        |
240 *       |
241 *      |
242 *     |
243 *    |
244 *   |
245 *  |
246 * /
247 *
248 *           0
249 *           |
250 *          |
251 *         |
252 *        |
253 *       |
254 *      |
255 *     |
256 *    |
257 *   |
258 *  |
259 * /
260 *
261 *           1
262 *           |
263 *          |
264 *         |
265 *        |
266 *       |
267 *      |
268 *     |
269 *    |
270 *   |
271 *  |
272 * /
273 *
274 *           3
275 *           |
276 *          |
277 *         |
278 *        |
279 *       |
280 *      |
281 *     |
282 *    |
283 *   |
284 *  |
285 * /
286 *
287 *           0
288 *           |
289 *          |
290 *         |
291 *        |
292 *       |
293 *      |
294 *     |
295 *    |
296 *   |
297 *  |
298 * /
299 *
300 *           1
301 *           |
302 *          |
303 *         |
304 *        |
305 *       |
306 *      |
307 *     |
308 *    |
309 *   |
310 *  |
311 * /
312 *
313 *           3
314 *           |
315 *          |
316 *         |
317 *        |
318 *       |
319 *      |
320 *     |
321 *    |
322 *   |
323 *  |
324 * /
325 *
326 *           0
327 *           |
328 *          |
329 *         |
330 *        |
331 *       |
332 *      |
333 *     |
334 *    |
335 *   |
336 *  |
337 * /
338 *
339 *           1
340 *           |
341 *          |
342 *         |
343 *        |
344 *       |
345 *      |
346 *     |
347 *    |
348 *   |
349 *  |
350 * /
351 *
352 *           3
353 *           |
354 *          |
355 *         |
356 *        |
357 *       |
358 *      |
359 *     |
360 *    |
361 *   |
362 *  |
363 * /
364 *
365 *           0
366 *           |
367 *          |
368 *         |
369 *        |
370 *       |
371 *      |
372 *     |
373 *    |
374 *   |
375 *  |
376 * /
377 *
378 *           1
379 *           |
380 *          |
381 *         |
382 *        |
383 *       |
384 *      |
385 *     |
386 *    |
387 *   |
388 *  |
389 * /
390 *
391 *           3
392 *           |
393 *          |
394 *         |
395 *        |
396 *       |
397 *      |
398 *     |
399 *    |
400 *   |
401 *  |
402 * /
403 *
404 *           0
405 *           |
406 *          |
407 *         |
408 *        |
409 *       |
410 *      |
411 *     |
412 *    |
413 *   |
414 *  |
415 * /
416 *
417 *           1
418 *           |
419 *          |
420 *         |
421 *        |
422 *       |
423 *      |
424 *     |
425 *    |
426 *   |
427 *  |
428 * /
429 *
430 *           3
431 *           |
432 *          |
433 *         |
434 *        |
435 *       |
436 *      |
437 *     |
438 *    |
439 *   |
440 *  |
441 * /
442 *
443 *           0
444 *           |
445 *          |
446 *         |
447 *        |
448 *       |
449 *      |
450 *     |
451 *    |
452 *   |
453 *  |
454 * /
455 *
456 *           1
457 *           |
458 *          |
459 *         |
460 *        |
461 *       |
462 *      |
463 *     |
464 *    |
465 *   |
466 *  |
467 * /
468 *
469 *           3
470 *           |
471 *          |
472 *         |
473 *        |
474 *       |
475 *      |
476 *     |
477 *    |
478 *   |
479 *  |
480 * /
481 *
482 *           0
483 *           |
484 *          |
485 *         |
486 *        |
487 *       |
488 *      |
489 *     |
490 *    |
491 *   |
492 *  |
493 * /
494 *
495 *           1
496 *           |
497 *          |
498 *         |
499 *        |
500 *       |
501 *      |
502 *     |
503 *    |
504 *   |
505 *  |
506 * /
507 *
508 *           3
509 *           |
510 *          |
511 *         |
512 *        |
513 *       |
514 *      |
515 *     |
516 *    |
517 *   |
518 *  |
519 * /
520 *
521 *           0
522 *           |
523 *          |
524 *         |
525 *        |
526 *       |
527 *      |
528 *     |
529 *    |
530 *   |
531 *  |
532 * /
533 *
534 *           1
535 *           |
536 *          |
537 *         |
538 *        |
539 *       |
540 *      |
541 *     |
542 *    |
543 *   |
544 *  |
545 * /
546 *
547 *           3
548 *           |
549 *          |
550 *         |
551 *        |
552 *       |
553 *      |
554 *     |
555 *    |
556 *   |
557 *  |
558 * /
559 *
560 *           0
561 *           |
562 *          |
563 *         |
564 *        |
565 *       |
566 *      |
567 *     |
568 *    |
569 *   |
570 *  |
571 * /
572 *
573 *           1
574 *           |
575 *          |
576 *         |
577 *        |
578 *       |
579 *      |
580 *     |
581 *    |
582 *   |
583 *  |
584 * /
585 *
586 *           3
587 *           |
588 *          |
589 *         |
590 *        |
591 *       |
592 *      |
593 *     |
594 *    |
595 *   |
596 *  |
597 * /
598 *
599 *           0
600 *           |
601 *          |
602 *         |
603 *        |
604 *       |
605 *      |
606 *     |
607 *    |
608 *   |
609 *  |
610 * /
611 *
612 *           1
613 *           |
614 *          |
615 *         |
616 *        |
617 *       |
618 *      |
619 *     |
620 *    |
621 *   |
622 *  |
623 * /
624 *
625 *           3
626 *           |
627 *          |
628 *         |
629 *        |
630 *       |
631 *      |
632 *     |
633 *    |
634 *   |
635 *  |
636 * /
637 *
638 *           0
639 *           |
640 *          |
641 *         |
642 *        |
643 *       |
644 *      |
645 *     |
646 *    |
647 *   |
648 *  |
649 * /
650 *
651 *           1
652 *           |
653 *          |
654 *         |
655 *        |
656 *       |
657 *      |
658 *     |
659 *    |
660 *   |
661 *  |
662 * /
663 *
664 *           3
665 *           |
666 *          |
667 *         |
668 *        |
669 *       |
670 *      |
671 *     |
672 *    |
673 *   |
674 *  |
675 * /
676 *
677 *           0
678 *           |
679 *          |
680 *         |
681 *        |
682 *       |
683 *      |
684 *     |
685 *    |
686 *   |
687 *  |
688 * /
689 *
690 *           1
691 *           |
692 *          |
693 *         |
694 *        |
695 *       |
696 *      |
697 *     |
698 *    |
699 *   |
700 *  |
701 * /
702 *
703 *           3
704 *           |
705 *          |
706 *         |
707 *        |
708 *       |
709 *      |
710 *     |
711 *    |
712 *   |
713 *  |
714 * /
715 *
716 *           0
717 *           |
718 *          |
719 *         |
720 *        |
721 *       |
722 *      |
723 *     |
724 *    |
725 *   |
726 *  |
727 * /
728 *
729 *           1
730 *           |
731 *          |
732 *         |
733 *        |
734 *       |
735 *      |
736 *     |
737 *    |
738 *   |
739 *  |
740 * /
741 *
742 *           3
743 *           |
744 *          |
745 *         |
746 *        |
747 *       |
748 *      |
749 *     |
750 *    |
751 *   |
752 *  |
753 * /
754 *
755 *           0
756 *           |
757 *          |
758 *         |
759 *        |
760 *       |
761 *      |
762 *     |
763 *    |
764 *   |
765 *  |
766 * /
767 *
768 *           1
769 *           |
770 *          |
771 *         |
772 *        |
773 *       |
774 *      |
775 *     |
776 *    |
777 *   |
778 *  |
779 * /
780 *
781 *           3
782 *           |
783 *          |
784 *         |
785 *        |
786 *       |
787 *      |
788 *     |
789 *    |
790 *   |
791 *  |
792 * /
793 *
794 *           0
795 *           |
796 *          |
797 *         |
798 *        |
799 *       |
800 *      |
801 *     |
802 *    |
803 *   |
804 *  |
805 * /
806 *
807 *           1
808 *           |
809 *          |
810 *         |
811 *        |
812 *       |
813 *      |
814 *     |
815 *    |
816 *   |
817 *  |
818 * /
819 *
820 *           3
821 *           |
822 *          |
823 *         |
824 *        |
825 *       |
826 *      |
827 *     |
828 *    |
829 *   |
830 *  |
831 * /
832 *
833 *           0
834 *           |
835 *          |
836 *         |
837 *        |
838 *       |
839 *      |
840 *     |
841 *    |
842 *   |
843 *  |
844 * /
845 *
846 *           1
847 *           |
848 *          |
849 *         |
850 *        |
851 *       |
852 *      |
853 *     |
854 *    |
855 *   |
856 *  |
857 * /
858 *
859 *           3
860 *           |
861 *          |
862 *         |
863 *        |
864 *       |
865 *      |
866 *     |
867 *    |
868 *   |
869 *  |
870 * /
871 *
872 *           0
873 *           |
874 *          |
875 *         |
876 *        |
877 *       |
878 *      |
879 *     |
880 *    |
881 *   |
882 *  |
883 * /
884 *
885 *           1
886 *           |
887 *          |
888 *         |
889 *        |
890 *       |
891 *      |
892 *     |
893 *    |
894 *   |
895 *  |
896 * /
897 *
898 *           3
899 *           |
900 *          |
901 *         |
902 *        |
903 *       |
904 *      |
905 *     |
906 *    |
907 *   |
908 *  |
909 * /
910 *
911 *           0
912 *           |
913 *          |
914 *         |
915 *        |
916 *       |
917 *      |
918 *     |
919 *    |
920 *   |
921 *  |
922 * /
923 *
924 *           1
925 *           |
926 *          |
927 *         |
928 *        |
929 *       |
930 *      |
931 *     |
932 *    |
933 *   |
934 *  |
935 * /
936 *
937 *           3
938 *           |
939 *          |
940 *         |
941 *        |
942 *       |
943 *      |
944 *     |
945 *    |
946 *   |
947 *  |
948 * /
949 *
950 *           0
951 *           |
952 *          |
953 *         |
954 *        |
955 *       |
956 *      |
957 *     |
958 *    |
959 *   |
960 *  |
961 * /
962 *
963 *           1
964 *           |
965 *          |
966 *         |
967 *        |
968 *       |
969 *      |
970 *     |
971 *    |
972 *   |
973 *  |
974 * /
975 *
976 *           3
977 *           |
978 *          |
979 *         |
980 *        |
981 *       |
982 *      |
983 *     |
984 *    |
985 *   |
986 *  |
987 * /
988 *
989 *           0
990 *           |
991 *          |
992 *         |
993 *        |
994 *       |
995 *      |
996 *     |
997 *    |
998 *   |
999 *  |
1000 */

```

```

54 mtype = { CAR };
55
56 /* Cars waiting sign for each traffic light */
57 chan carsWaiting[N_TRAFFIC_LIGHTS] = [1] of { mtype };
58
59 proctype LineTrafficGenerator( byte initTlId )
60 {
61     byte tlId;
62
63     tlId = initTlId;
64
65     do
66         :: carsWaiting[tlId] ! CAR;
67     od
68 }
69
70 /* Manager messages */
71 mtype = { LOCK, INT, RELEASE };
72
73 /* Intersections lock/release requests queue.
74 * Message contains requestee traffic light identifier.
75 */
76 chan intersectionLockRequests[N_INTERSECTIONS] =
77     [N_TRAFFIC_LIGHTS] of { mtype, byte };
78 chan intersectionLockGranted[N_TRAFFIC_LIGHTS] =
79     [0] of { mtype };
80 chan intersectionReleaseRequests[N_INTERSECTIONS] =
81     [0] of { mtype };
82
83 /* Macro for obtaining intersection resource */
84 #define lockIntersection( intId, tlId ) \
85     intersectionLockRequests[intId] ! LOCK(tlId); \
86     intersectionLockGranted[tlId] ? INT
87
88 /* Macro for releasing intersection resource */
89 #define unlockIntersection( intId ) \
90     intersectionReleaseRequests[intId] ! RELEASE
91
92 /* Intersection resource manager */
93 proctype Intersection( byte initIntId )
94 {
95     byte intId, tlId;
96
97     intId = initIntId;
98
99 endInt:
100 do
101     :: intersectionLockRequests[intId] ? LOCK(tlId) ->
102         /* Handle request */
103         intersectionLockGranted[tlId] ! INT;
104
105         /* Wait for release */
106         progressGiveIntersection:
107             intersectionReleaseRequests[intId] ? RELEASE;
108     od;
109 }
110

```

```

111 /* Traffic lights states */
112 mtype = { RED, GREEN };
113
114 /* Traffic light state */
115 mtype tlColor[N_TRAFFIC_LIGHTS];
116
117 /* Main traffic light process */
118 proctype TrafficLight( byte initTlId )
119 {
120     byte tlId;
121
122     tlId = initTlId;
123
124     assert(tlColor[tlId] == RED);
125
126 endTL:
127     do
128         :: carsWaiting[tlId] ? [CAR] ->
129             /* Cars in queue */
130
131             /* Lock dependent intersections */
132             if
133                 :: tlId == SN ->
134                     lockIntersection(0, tlId);
135                     lockIntersection(1, tlId);
136                     lockIntersection(2, tlId);
137                 :: tlId == WE ->
138                     lockIntersection(0, tlId);
139                     lockIntersection(3, tlId);
140                 :: tlId == ES ->
141                     lockIntersection(2, tlId);
142                     lockIntersection(3, tlId);
143                     lockIntersection(4, tlId);
144                 :: tlId == NE ->
145                     lockIntersection(1, tlId);
146                     lockIntersection(4, tlId);
147             fi;
148
149             /* Allow passing */
150 progressPassCar:
151             atomic
152             {
153                 printf("MSC:_Traffic_light_#%d:_GREEN\n", tlId);
154                 tlColor[tlId] = GREEN;
155
156                 /* Pass car */
157                 /* Note: atomic for easier claim construction */
158                 carsWaiting[tlId] ? CAR;
159                 printf("MSC:_Traffic_light_#%d:_pass_cars\n", tlId);
160             };
161
162             /* Forbid passing */
163             atomic
164             {
165                 printf("MSC:_Traffic_light_#%d:_RED\n", tlId);
166                 tlColor[tlId] = RED;
167             };

```

```

168
169      /* Release dependent intersections */
170      if
171      :: tId = SN ->
172          unlockIntersection(2);
173          unlockIntersection(1);
174          unlockIntersection(0);
175      :: tId = WE ->
176          unlockIntersection(3);
177          unlockIntersection(0);
178      :: tId = ES ->
179          unlockIntersection(4);
180          unlockIntersection(3);
181          unlockIntersection(2);
182      :: tId = NE ->
183          unlockIntersection(4);
184          unlockIntersection(1);
185      fi;
186  od;
187 }
188
189 /* The main model function */
190 init
191 {
192     byte tId, intId;
193
194     /* Reset traffic lights colors */
195     tId = 0;
196     do
197     :: tId < N_TRAFFIC_LIGHTS ->
198         tlColor[tId] = RED;
199         tId++;
200     :: else ->
201         break;
202     od;
203
204     atomic
205     {
206         /* Start intersection managers processes */
207         intId = 0;
208         do
209         :: intId < N_INTERSECTIONS ->
210             run Intersection(intId);
211             intId++;
212         :: else ->
213             break;
214         od;
215
216         /* Start traffic lights processes */
217         tId = 0;
218         do
219         :: tId < N_TRAFFIC_LIGHTS ->
220             run TrafficLight(tId);
221             tId++;
222         :: else ->
223             break;
224         od;

```

```

225
226      /* Start cars generator process */
227      /*run CarsGenerator();*/
228      tlId = 0;
229      do
230      :: tlId < N_TRAFFIC_LIGHTS ->
231          run LineTrafficGenerator(tlId);
232          tlId++;
233      :: else ->
234          break;
235      od;
236  }
237 }
238
239 /*
240  * Correctness requirements.
241  */
242
243 /* Car crash accident definition */
244 #define accident_01 (tlColor[0] == GREEN && tlColor[1] == GREEN)
245 #define accident_02 (tlColor[0] == GREEN && tlColor[2] == GREEN)
246 #define accident_03 (tlColor[0] == GREEN && tlColor[3] == GREEN)
247 #define accident_13 (tlColor[1] == GREEN && tlColor[3] == GREEN)
248 #define accident_23 (tlColor[2] == GREEN && tlColor[3] == GREEN)
249
250 /* Car waiting at traffic light definition */
251 #define car_waiting_0 (len(carsWaiting[0]) > 0)
252 #define car_waiting_1 (len(carsWaiting[1]) > 0)
253 #define car_waiting_2 (len(carsWaiting[2]) > 0)
254 #define car_waiting_3 (len(carsWaiting[3]) > 0)
255
256 /* Traffic light is green definition */
257 #define tl_green_0 (tlColor[0] == GREEN)
258 #define tl_green_1 (tlColor[1] == GREEN)
259 #define tl_green_2 (tlColor[2] == GREEN)
260 #define tl_green_3 (tlColor[3] == GREEN)
261
262 /* Safety: Intersecting roads traffic light both never has GREEN state */
263 /*
264  * [] (!accident_01)
265  * [] (!accident_02)
266  * [] (!accident_03)
267  * [] (!accident_13)
268  * [] (!accident_23)
269  */
270
271 /* Liveness: If cars wait on traffic light, then in future traffic light
272  * became GREEN */
273 /*
274  * [] (car_waiting_0 -> <> tl_green_0)
275  * [] (car_waiting_1 -> <> tl_green_1)
276  * [] (car_waiting_2 -> <> tl_green_2)
277  * [] (car_waiting_3 -> <> tl_green_3)
278  */

```

Список литературы

- [1] И.В. Шошмина and Ю.Г. Карпов. *Введение в язык Promela и систему комплексной верификации Spin*. СПбГПУ, 2009.
- [2] Ю.Г. Карпов. *Model checking. Верификация параллельных и распределенных программных систем*. БХВ-Петербург, 2010.