Implementing of a Unified Model for Real-Time Systems

February 2023

1 Introduction

Here in this document, we discuss in detail some more benchmarks coming from different categories. We start our discussion first by comparsion between existing benchmarks in TChecker and out tool in Section 2, this is followed by Some discussion about event clock automata with diagonal constraints in Section 3, which is followed by a discussion on the general model in Section 4. We end this document with Section 5, in which we discuss some real-benchmarks. We have implemented our algorithm in Version 3 of the open-source tool TChecker[1].

The tool can handle normal-clocks, event-clocks, history-clocks, prophecy-clocks, and timers.

2 Comparison On Existing Benchmarks

We have compared the performance of our tool on the existing benchmarks. The results are presented in Table 1. For these experiments, we mapped clocks in the benchmarks to normal clocks in our tool. Since our tool also uses $\mathcal{G}-$simulation, we get exactly an equal number of states (both visited and stored) in both, the standard implementation and our implementation. Visited nodes are the nodes for which exploration is triggered, and stored nodes are the largest nodes in the final zone graph. We perform a bit worse in time because we maintain an additional internal clock (which is required for implementing general programs on transitions), and maintaining this clock makes the difference bound matrix larger, and hence operations on the difference bound matrix are slower.

Table 1: Comparison of existing benchmarks using normal clocks in our tool

Input File	**G-Sim**			**Gen**		
	Visited States	**Stored States**	**Running Time (sec)**	**Visited States**	**Stored States**	**Running Time (sec)**
CSMACD 5	850	850	0.023	850	850	0.029
CSMACD 7	7490	7490	0.187	7490	7490	0.217
CSMACD 10	144898	144898	5.727	144898	144898	6.889
dining philosophers 5	911	911	0.146	911	911	0.179
dining philosophers 6	5480	5480	4.911	5480	5480	6.410
dining philosophers 7	TIMEOUT			TIMEOUT		
fischer 5	977	727	0.011	977	727	0.013
fischer 7	11951	7737	0.263	11951	7737	0.301
fischer 10	447598	260998	29.1574	447598	260998	34.6517
fire alarm 5	46	46	0.001	46	46	0.001
fire alarm 7	148	148	0.009	148	148	0.009
fire alarm 10	1053	1053	0.167	1053	1053	0.176
fddi 5	352	129	0.020	352	129	0.028
fddi 7	1348	237	0.212	1348	237	0.317
fddi 10	10219	459	10.139	10219	459	16.797

3 Comparison on Event-Clock Automata without Diagonal Constraints

Now we compare the performance of our tool on event-clock automata. For comparison, since there is no existing tool that works on event-clock automata, we used the algorithm mentioned in [2] to convert event-clock automata to a language equivalent timed automata. Then we run a zone-graph exploration on the timed automata and compare its performance with our tool. The first benchmark that we use is $B_1(K)$ parameterized on the maximal constant $K \geq 1$. Figure 1 displays $B_1(K)$.

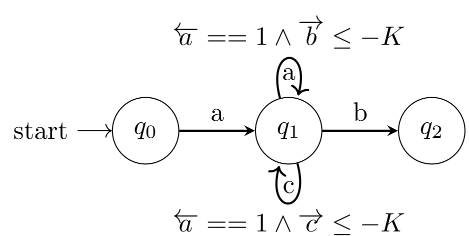


Figure 1: $B_1(K)$, The parameter K is the constant on guard in both loops of state q_1

Similar to $B_1(K)$, we have another benchmark $B_2(N)$, but in $B_2(N)$ we parameterize the number of transitions and clocks (N), and keep the maximal constant 1000. Figure 2 with $K = 1000$ gives $B_2(N)$. Taking motivation from $B_1(K)$ and $B_2(N)$ we construct another benchmark $B_3(K, N)$, where K is the maximal constant and N is the number of c loops appearing as displayed in Figure 2. So, $B_2(N) := B_3(1000, N)$, and $B_1(K) := B_3(K, 1)$.

Since the initial state has only one outgoing action on an a (from q_0 to q_1), action a is always the first action.

There is an a loop on q_1. This loop can only be taken if the time elapsed from the previous a is 1 time unit, which is enforced by the guard $\overleftarrow{a} == 1$. Once this loop is taken, a b action can only be taken after K time units, which is enforced by the guard $\overrightarrow{b} \leq -K$.

There are also c loops on q_1. Any c_i loop can only be taken if the time elapsed from the previous a is 1 time unit (enforced by the guard $\overleftarrow{a} == 1$), and after taking a c_i loop the next c_i loop can only be taken after K time units (enforced by $\overrightarrow{c_i} \leq -K$).

After the initial a every next c action or a action occurs at exactly 1 time units. This is enforced by the guard $\overleftarrow{a} == 1$ on every loop. Once a c_i action occurs the next c_i action can only occur after K time units, this is enforced by the guard $c_i \leq -K$ on the c_i loop. Once an a action occurs, b action can occur only after K time units.

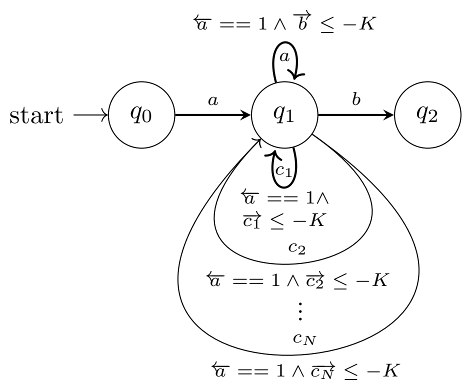


Figure 2: $B_3(K, N)$, K is the maximal constant and $N \geq 1$ is the number of c loops on q_1

Input Benchmark	G-Sim			Gen		
	Visited States	Stored States	Running Time (sec)	Visited States	Stored States	Running Time (sec)
$B_1(100)$	107	7	0.0009	3	3	0.00008
$B_1(1000)$	1007	7	0.009	3	3	0.00008
$B_1(10000)$	10007	7	0.089	3	3	0.0001
$B_1(100000)$	100007	7	0.857	3	3	0.00009
$B_2(2)$	3013	13	0.042	3	3	0.0001
$B_2(3)$	7025	25	0.142	3	3	0.0001
$B_2(4)$	15049	49	0.408	3	3	0.0002
$B_2(5)$	31097	97	1.077	3	3	0.0002
$B_2(6)$	63193	193	2.749	3	3	0.0003
$B_2(7)$	127385	385	7.045	3	3	0.0005
$B_2(100)$	TIMEOUT			3	3	0.789
$B_3(10000, 4)$	150049	49	3.999	3	3	0.0002
$B_3(5000, 5)$	155097	97	5.49699	3	3	0.0002
$B_3(10^6, 1)$	1000007	7	8.567	3	3	0.0001
$B_3(50000, 120)$	TIMEOUT			3	3	1.49

Table 2: Performance comparison on B_1, B_2, and B_3

4 Benchmarks and Results on General Model

We will now look into event-clock automata with diagonals. We note that there is no known algorithm that converts event-clock automata with diagonals to language-equivalent timed automata, so we do not have anything to compare our tool. We will now look at a simple benchmark example B_4 Figure 3.

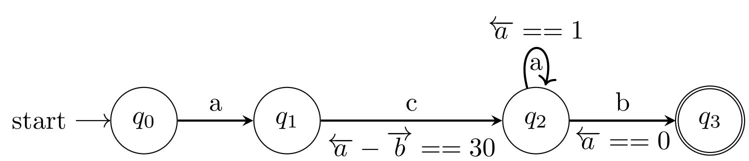


Figure 3: B_4

We now present another parameterized benchmark $B_5(I, J)$, where $1 \leq I, J$. The state q_3 in $B_5(I, J)$ is reachable if and only if J divides I. We can observe in benchmark B_5, that q_2 to q_3 edge can be taken only if $\overleftarrow{a} = \overrightarrow{b} = 0$, and also after taking the loop on q_2 the constraint $\overleftarrow{a} - \overrightarrow{b}$ decrements by J. So, the edge q_2 to q_3 can be taken if and only if J divides I.

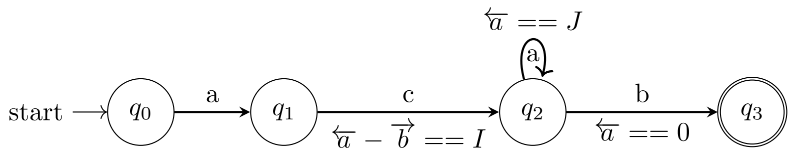


Figure 4: $B_5(I, J)$

We now present a benchmark B_6 on event-clock automata and then parameterize it in benchmark $B_7(N)$.

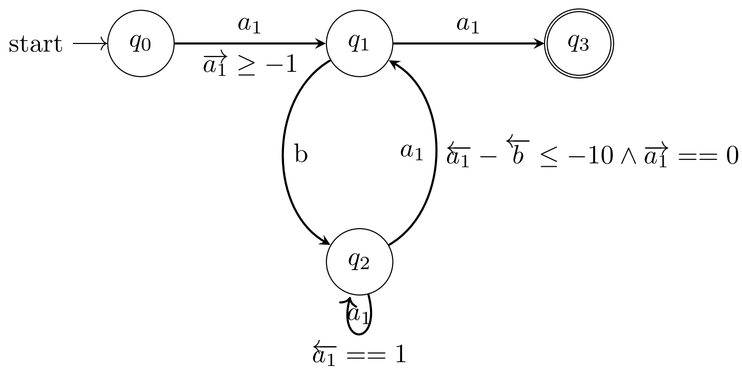


Figure 5: B_6

We now make a parameterized benchmark $B_7(N)$, where $N \geq 1$. This benchmark is depicted in Figure 6.

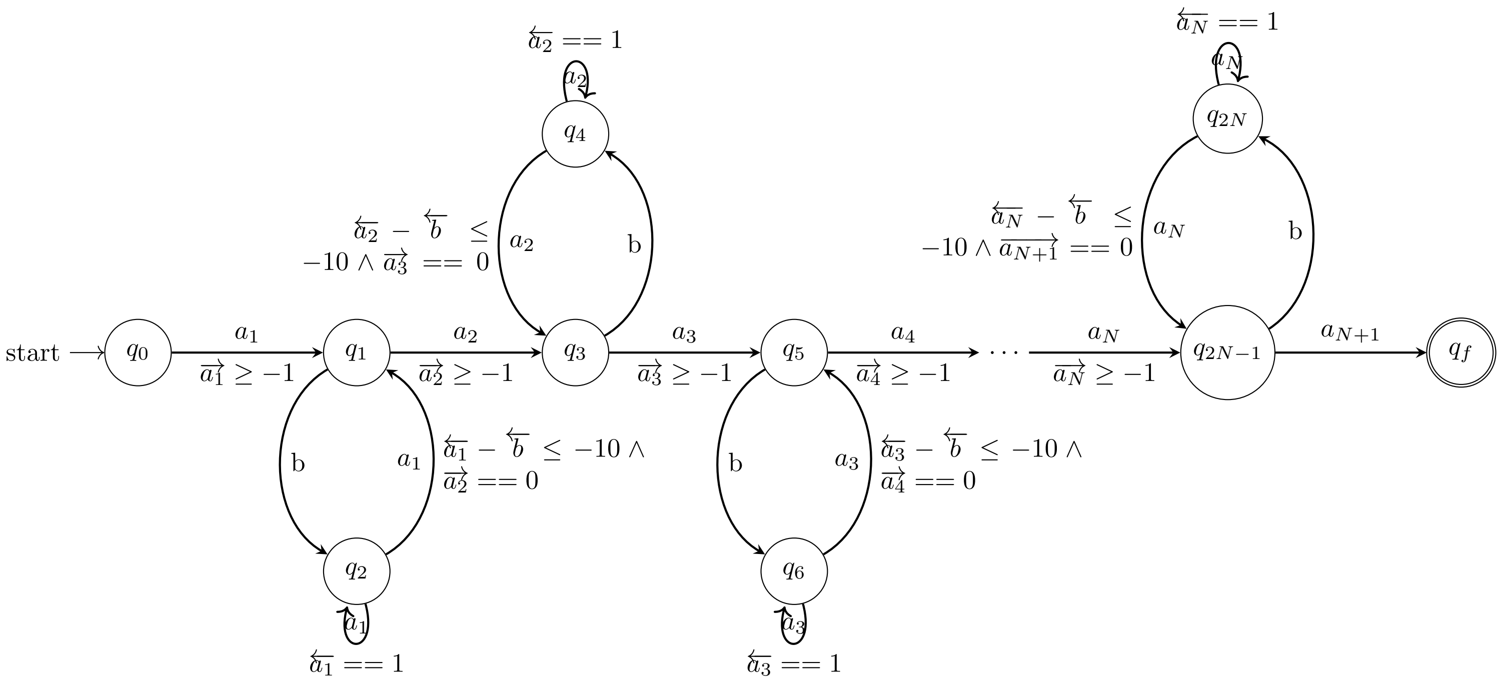


Figure 6: $B_7(N)$, where N is as shown is the number of a actions and also one minus the size of the backbone $q_0 \rightarrow_{a_1} q_1 \ldots \rightarrow_{a_{N+1}} q_f$

Now we will look into some benchmarks of the general model. We start with the benchmark $B_8(I, J)$ as given in Figure 7. It contains two prophecy clocks p_1 and p_2 and one normal clock n.

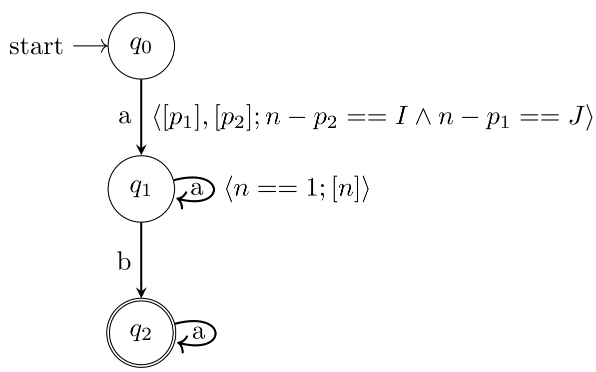


Figure 7: $B_8(I, J)$, I and J are parameters on the guard from q_0 to q_1, there are two prophecy clocks p_1 and p_2 and a normal clock n in this automaton

We give another benchmark B_9, where we parameterize on N which is the number of substructures attached to the initial node l_0 big loops $B_9(N)$. $B_9(2)$ is given in Figure 8.

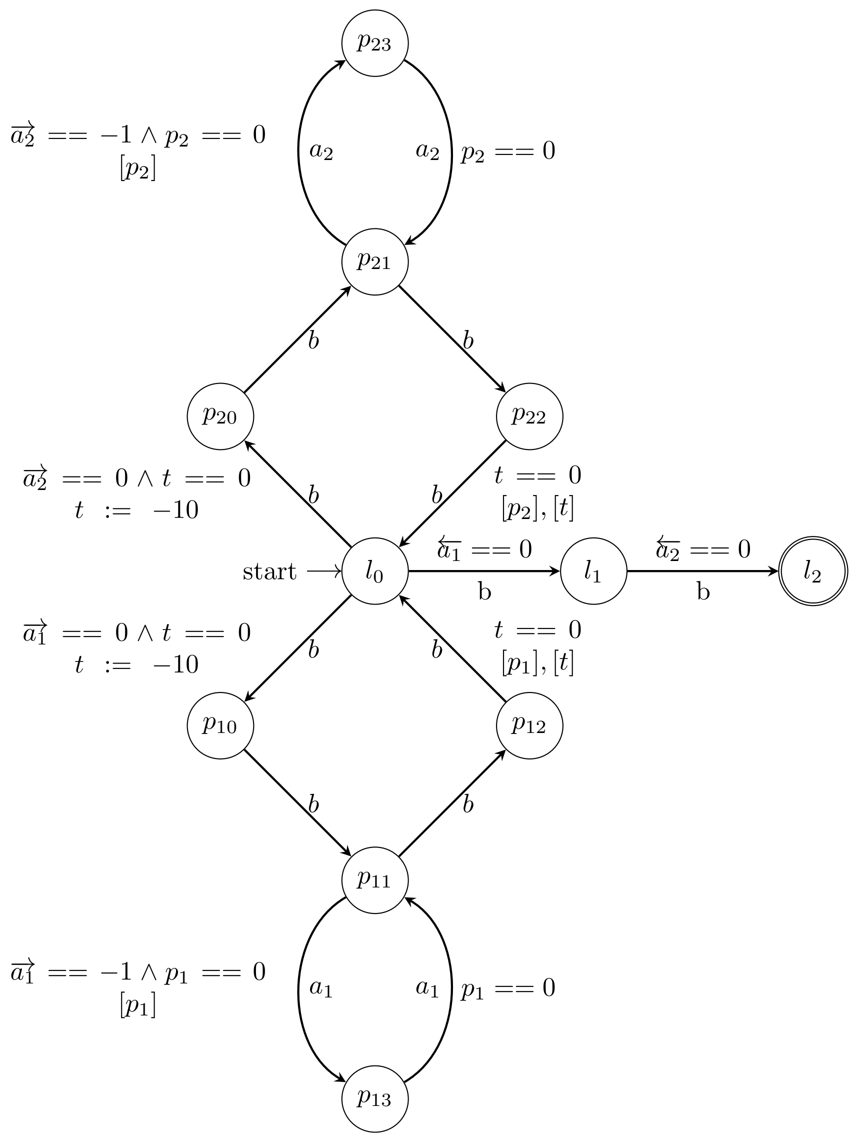


Figure 8: $B_9(2)$

We now present a benchmark B_{10} of a non X_d safe model which does not have finite zone graph.

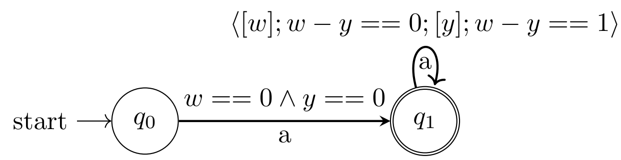


Figure 9: B_{10} Example with infinite zone graph! Clocks w and y are prophecy clocks

We now give a concrete benchmark $B_{11}(N)$ parameterized by N, where reachability is not known for every parameter N. In $B_{11}(N)$ the final state q_f is reachable for every N if and only if the collatz conjecture is true! We are just executing the steps of collatz conjecture for a particular N in $B_{11}(N)$. Note that for this benchmark we stop our search after reaching the final state (and if the final zone condition is satisfied).

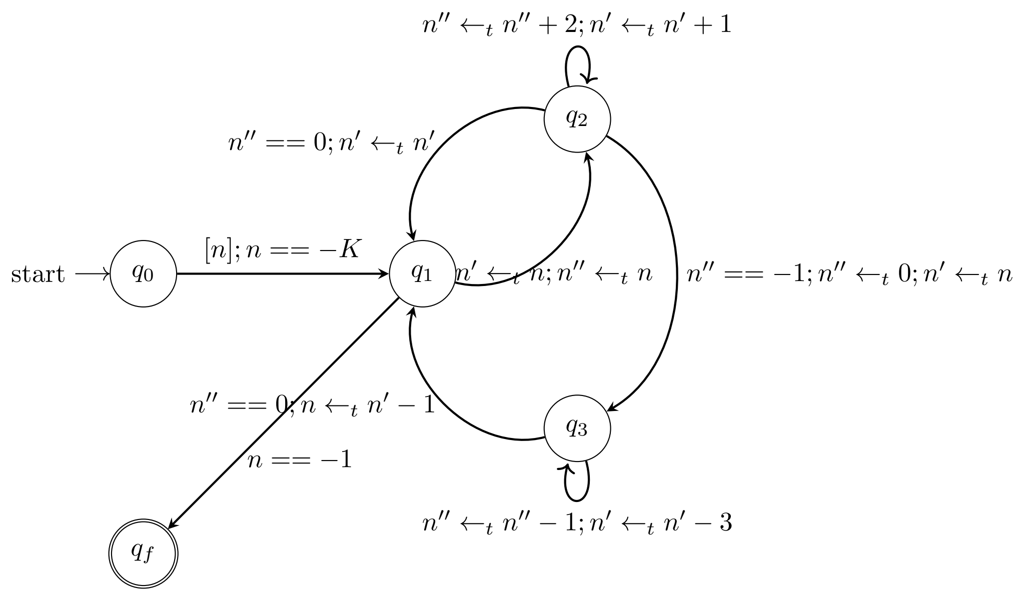


Figure 10: $B_{11}(K)$ The clocks t, n, n', n'' are all prophecy clocks. Note that there is a normal clock g which is always restricted to 0 implicitly in every guard, this is to ensure that we do not elapse any time.

5 Real-Life Benchmarks

We now show the standard phone dialling protocol modelled using only timers. We call it B_{12} The properties are modelled using normal clocks. The properties are:

1. the first digit of the number does not arrive in 30 seconds after the beginning of the dialling (picking up the receiver)
2. the current digit which is not the first does not arrive in 20 seconds after the arrival of the previous digit
3. the total time delay from the beginning of the number dialling reaches 60 seconds.

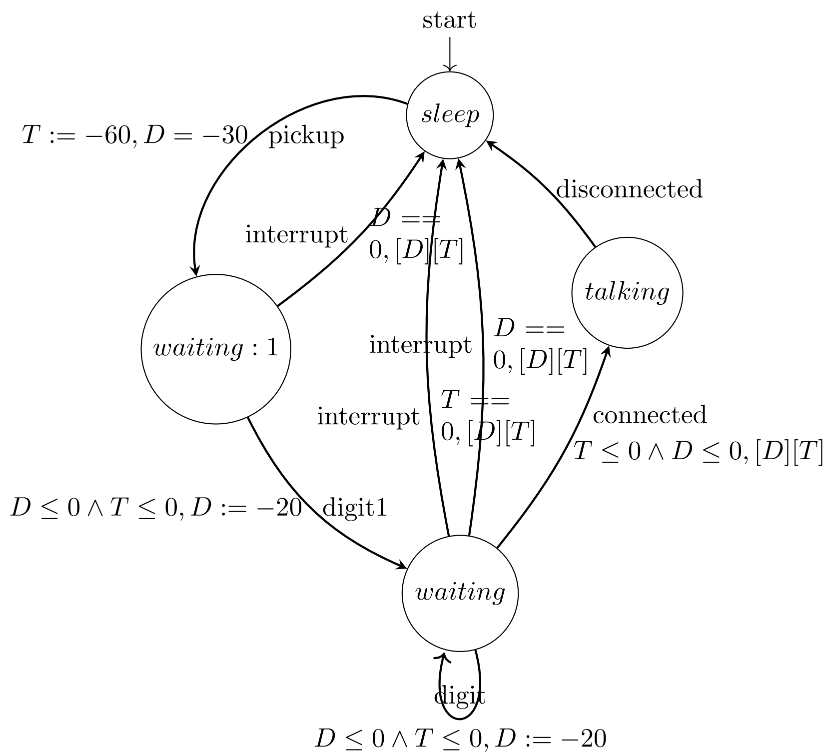


Figure 11: B_{12} without properties model of phone dialling protocol, clocks T and D are timers

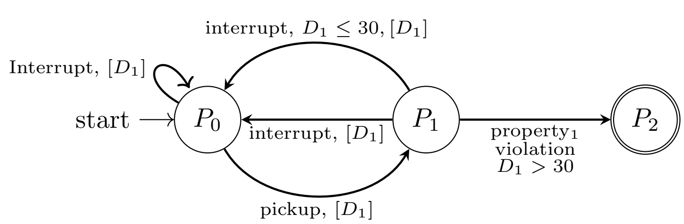


Figure 12: Property1 of B_{12}, D_1 is a normal clock

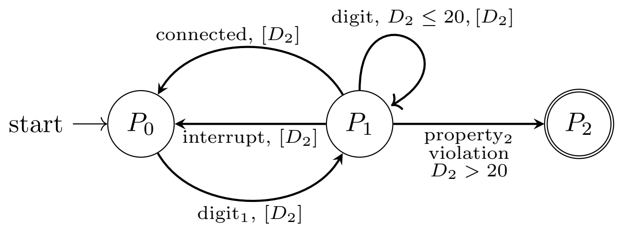


Figure 13: Property2 of B_{12}, D_2 is a normal clock

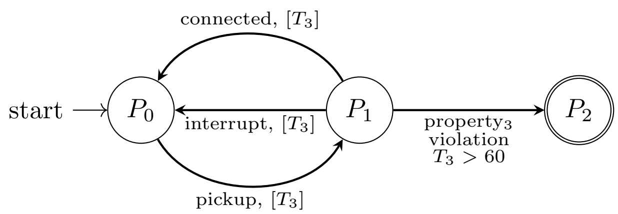
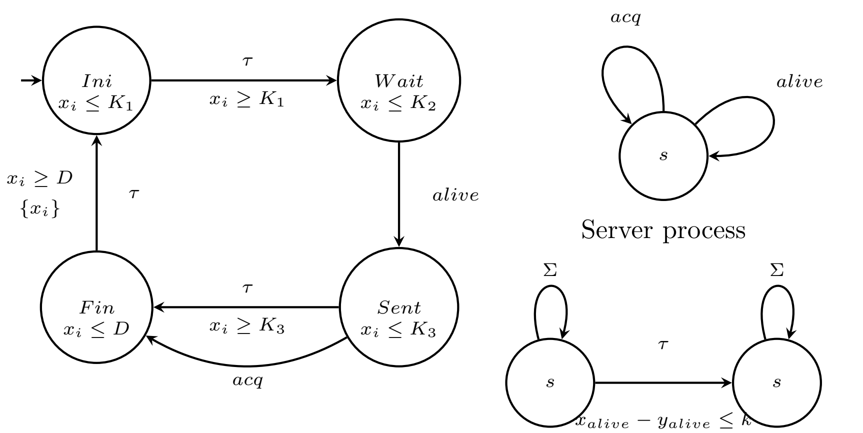


Figure 14: Property3 of B_{12}, T_3 is a normal clock

Input File	**Visited States**	**Stored States**	**Running Time (sec)**
B_4	304	304	0.198
$B_5(3000, 20)$	154	154	0.048
$B_5(5000, 37)$	139	139	0.041
$B_5(8000, 21)$	384	384	0.309
$B_5(8000, 7)$	1145	1145	2.712
B_6	14	4	0.0007
$B_7(17)$	648	53	0.476
$B_7(25)$	1252	77	2.534
$B_7(30)$	1727	92	5.903
$B_8(100, 50)$	53	53	0.017
$B_8(250, 70)$	73	73	0.0315
$B_8(400, 500)$	403	403	1.035
$B_9(2)$	71	71	0.003
$B_9(4)$	141	141	0.0107
$B_9(32)$	1121	1121	1.309
B_{10}	TIMEOUT		
$B_{11}(11)$	75	78	0.033
$B_{11}(101)$	514	519	0.509
$B_{11}(1001)$	4459	4464	38.194
B_{12}	6	4	0.0003
ALT PROTOCOL Prop1	114	114	0.038
ALT PROTOCOL Prop2	168	168	0.026
CSMACD_bounded(1)	34	26	0.0054
CSMACD_bounded(4)	4529	2068	2.597
Toy1_bounded	3	2	0.001
Toy2_bounded	23	12	0.011
Fire-Alarm(5)_forbidden_pattern	46	46	0.027
Toy1_forbidden_pattern	4	2	0.0008
Toy2_forbidden_pattern	2	1	0.0006

Table 3: Table comparing benchmarks



$Sensor_i$ ECA modelling specification

Figure 15: Fire-alarm model and specification.

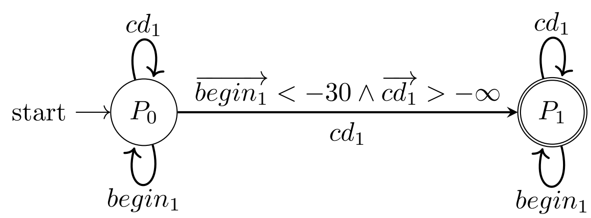
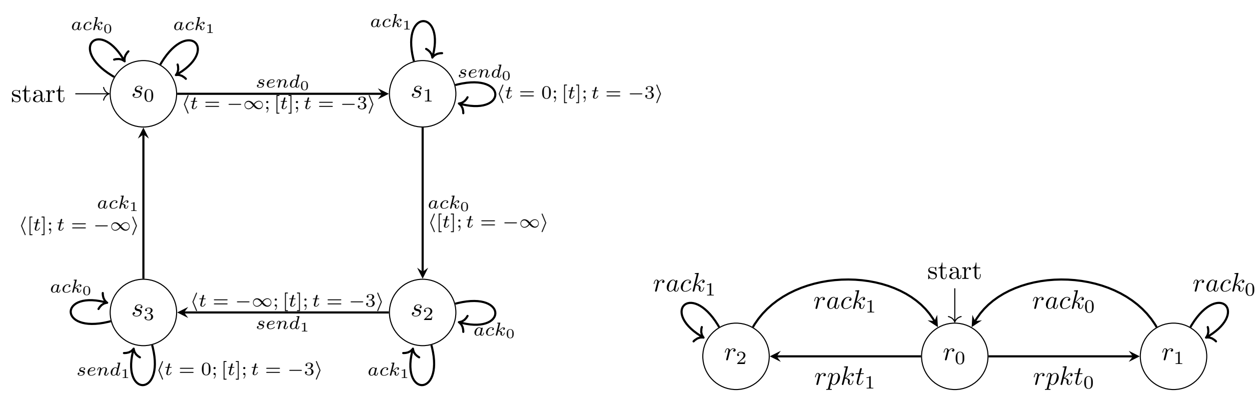


Figure 18: CSMACD Property



(a) Sender (b) Receiver

Figure 19: Alternating-Bit Protocol

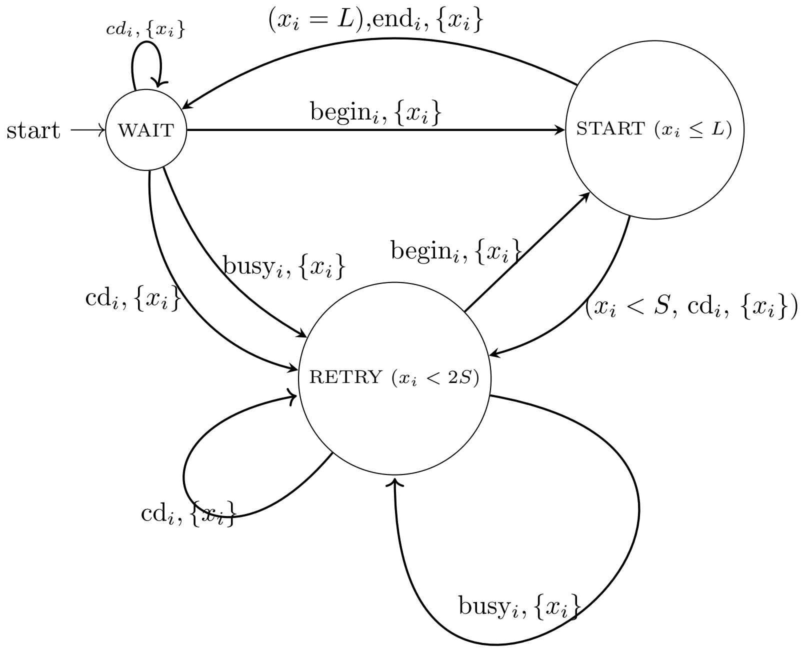


Figure 16: CSMACD Station Note that this image is different from the one modelled in TChecker (in TChecker it is modelled using committed

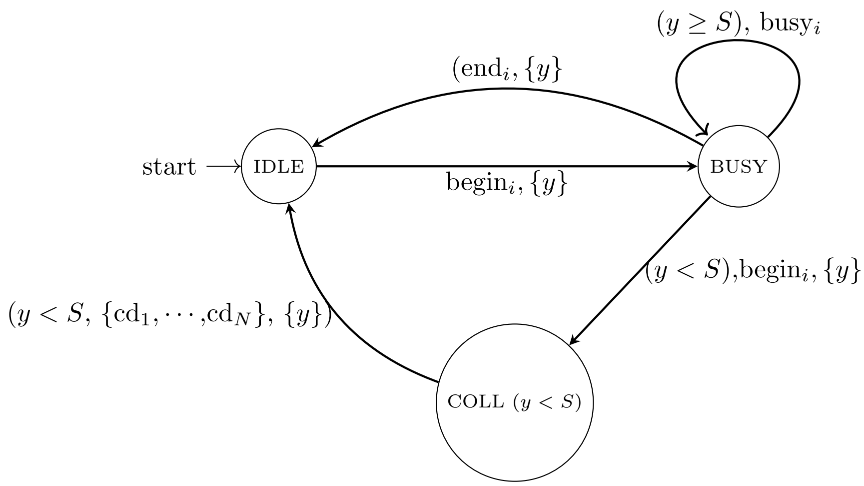
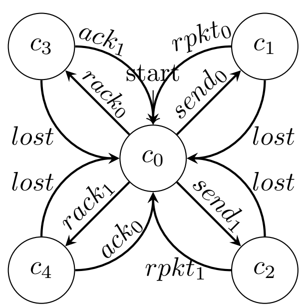
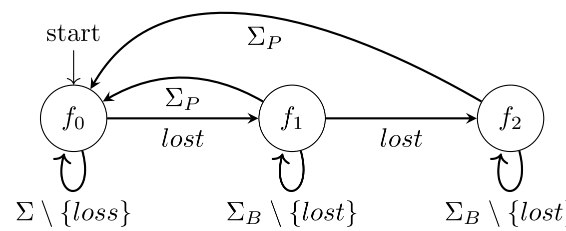


Figure 17: CSMACD Bus



(a) Channel



(b) Fair, $\Sigma_B = \{send_0, send_1, rack_0, rack_1\}, \Sigma_P = \{rpkt_0, rpkt_1, ack_0, ack_1\}, \Sigma = \Sigma_B \sqcup \Sigma_P \sqcup \{loss\}$

Figure 20: Alternating-Bit Protocol Channel and Fairness Prop

We model sending a packet with identifier $i \in \{0, 1\}$ in the sender with the action $send_i$, and receiving an acknowledgement with identifier $i \in \{0, 1\}$ in the sender with the action ack_i

This is a property that we want to check for the sender of ABP: after the sending $send_0$, the sender should receive an ack_0 before sending $send_1$.

We model the negation of this property in an ECA given in Figure 21

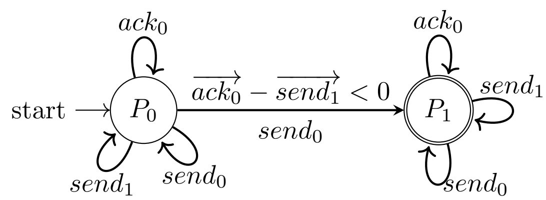


Figure 21: ABP Property1

This is another property that we want to check for the sender of ABP (bounded response property): after sending a $send_0$, the sender must receive an ack_0 within 3 time units. This property is falsified in ABP.

We model the negation of this property in an ECA given in Figure 22

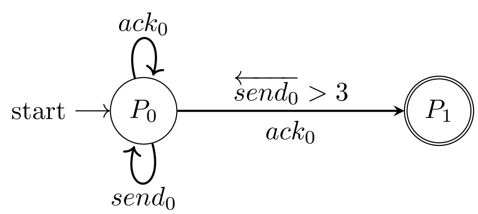


Figure 22: ABP Property2

This is the property that we want to check for process 1: for every collision detection (cd_1) except the last one, there is a begin ($begin_1$) seen within 30 time units. The following ECA given in Figure 18 gives the negation of this property.

References

[1] https://github.com/ticktac-project/tchecker

[2] event-clock automata alur