Default Report (Copy) (Copy)

LTL: RoboTrain

May 24, 2022 12:49 PM MDT

Q2 - Timing on 1st page (0 questions)

Times in seconds

#	Field	Minimum	Maximum	Mean	Std Deviation	Count
1	Timing - First Click	0	35	10	14	11
2	Timing - Last Click	0	101	20	33	11
3	Timing - Page Submit	63	120	95	15	11
4	Timing - Click Count	0	13	2	4	11

Q20 - Timing on 2nd page (trace)

Times in seconds

#	Field	Minimum	Maximum	Mean	Std Deviation	Count
1	Timing - First Click	3	106	34	25	11
2	Timing - Last Click	230	848	462	184	11
3	Timing - Page Submit	232	852	467	185	11
4	Timing - Click Count	12	191	73	61	11

Q32 - Timing on 3rd page (LTL to English)

Times in seconds

#	Field	Minimum	Maximum	Mean	Std Deviation	Count
1	Timing - First Click	2	73	19	19	11
2	Timing - Last Click	208	713	367	154	11
3	Timing - Page Submit	253	717	388	147	11
4	Timing - Click Count	16	212	74	58	11

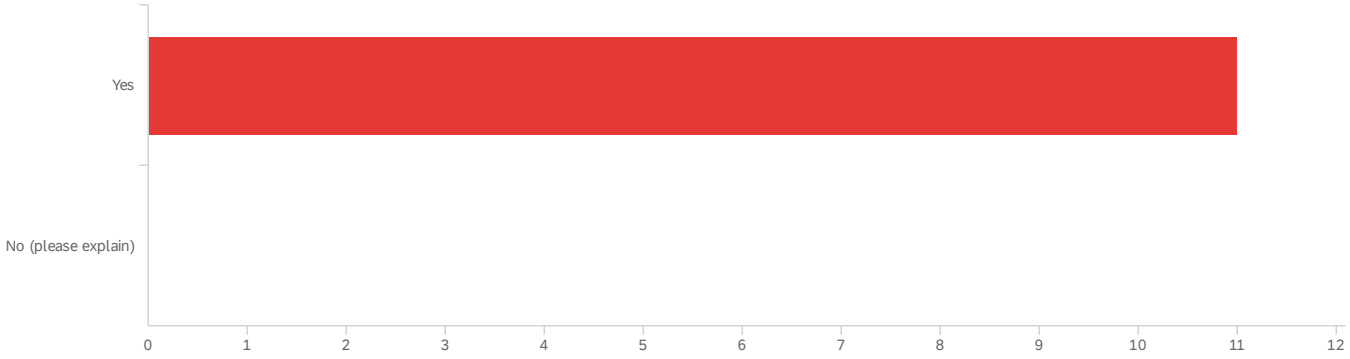
Q42 - Timing on 4th page (English to LTL)

Times in seconds

#	Field	Minimum	Maximum	Mean	Std Deviation	Count
1	Timing - First Click	2	60	19	17	11
2	Timing - Last Click	217	1057	467	228	11
3	Timing - Page Submit	223	1064	474	229	11
4	Timing - Click Count	11	181	85	58	11

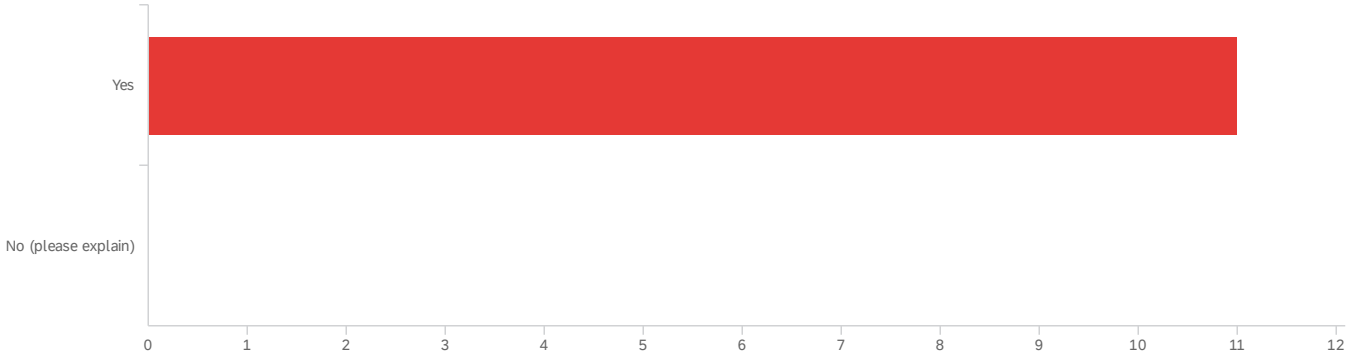
Q7 - Does the example make sense to you?*

expecting: Yes



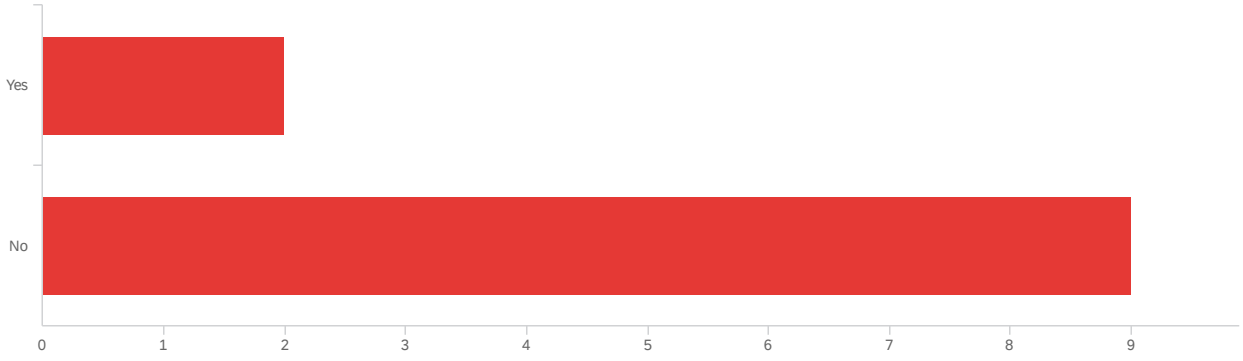
Q9 - Does the example make sense to you?*

expecting: Yes



Q11 - Q. Is the formula Engine satisfied by this trace?*

correct = No, trace = {GB} {RGB}*



Q47 - (Optional) Feel free to explain your reasoning

(Optional) Feel free to explain your reasoning

I think no because I think "engine" implies that it needs to be in the first element. "Eventually engine" would be satisfied here.

Not sure, I'm assuming an implicit always in front of engine.

The engine is not on in the first state

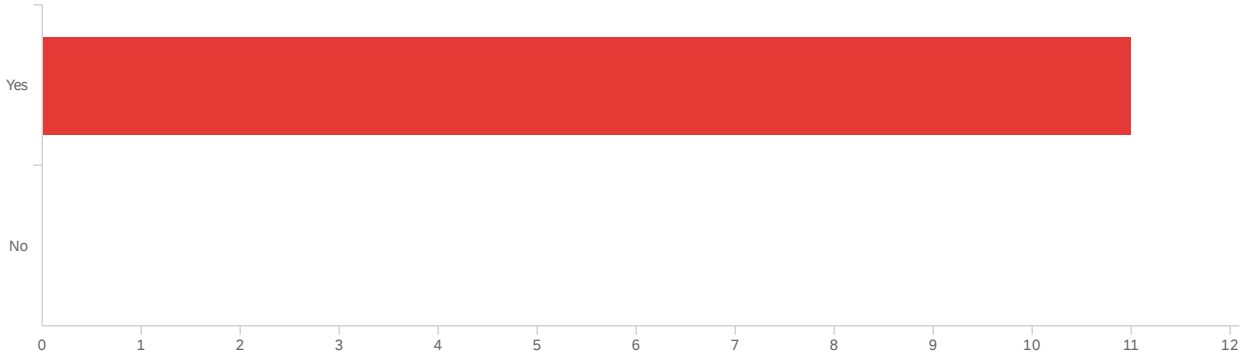
instead of {always Engine}, it's just {Engine}, which made me think that the engine only had to be on at any trace for the formula to be satisfied -- but changed my mind after the next question, because it felt that we are assuming we are at the first state

But also, I remember in class that if you don't really have a temporal modifier (always, eventually etc.) then the constraint applies specifically to the first case, which would be no.

It is not true in the first state, and constraints without LTL operators only must apply to the first state.

Q45 - Q. Is the formula next_state (next_state (next_state (Engine))) satisfied by this trace?*

Correct = yes; trace = {E} {} {} {E} {}



#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
1	Q. Is the formula next_state (next_state (next_state (Engine))) satisfied by this trace?*	1.00	1.00	1.00	0.00	0.00	11

#	Field		Choice Count
1	Yes	100.00%	11
2	No	0.00%	0
			11

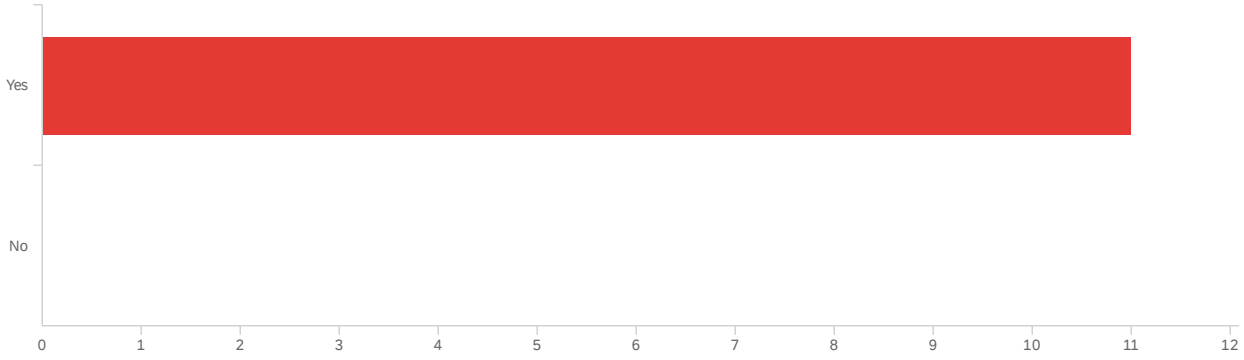
Showing rows 1 - 3 of 3

Q48 - (Optional) Feel free to explain your reasoning

(Optional) Feel free to explain your reasoning

Q13 - Q. Is the formula always (Engine implies next_state (next_state (next_state (Engine)))) satisfied by this trace?*

correct = Yes, trace = {} {RGB} {RGB} {} {RGB}*

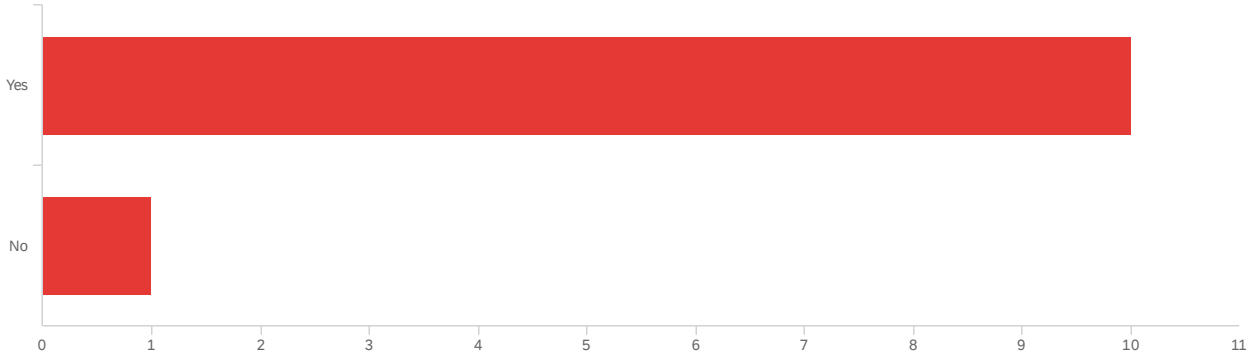


Q49 - (Optional) Feel free to explain your reasoning

(Optional) Feel free to explain your reasoning

Q14 - Q. Is the formula (next_state (Engine)) until (next_state (DoorOpen)) satisfied by this trace?*

correct = Yes, trace = {RB} {RB} {RB} {RGB} {B}*



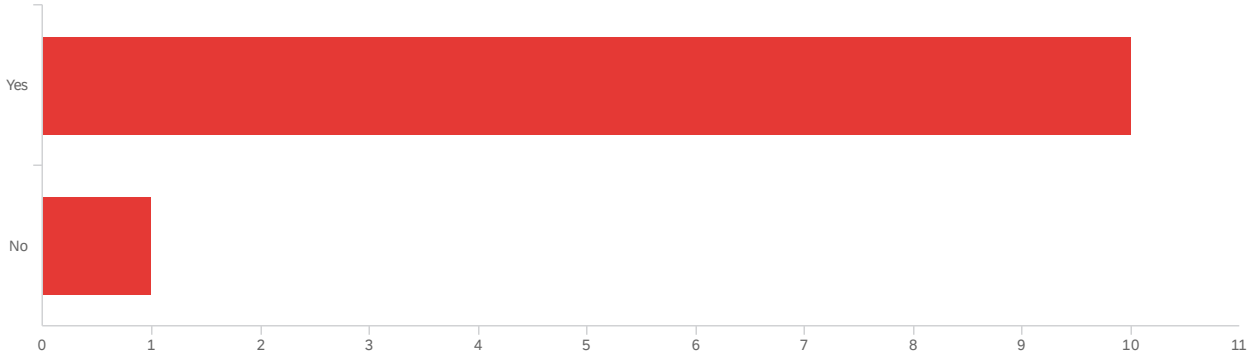
Q50 - (Optional) Feel free to explain your reasoning

(Optional) Feel free to explain your reasoning

I think this is satisfied, but also it may need to be wrapped in an always, that part is usually kind of blurry.

Q15 - Q. Is the formula { eventually (Red) } and { eventually (Green) } satisfied by this trace?*

correct = Yes, trace = {} {G} {} {} {R}*

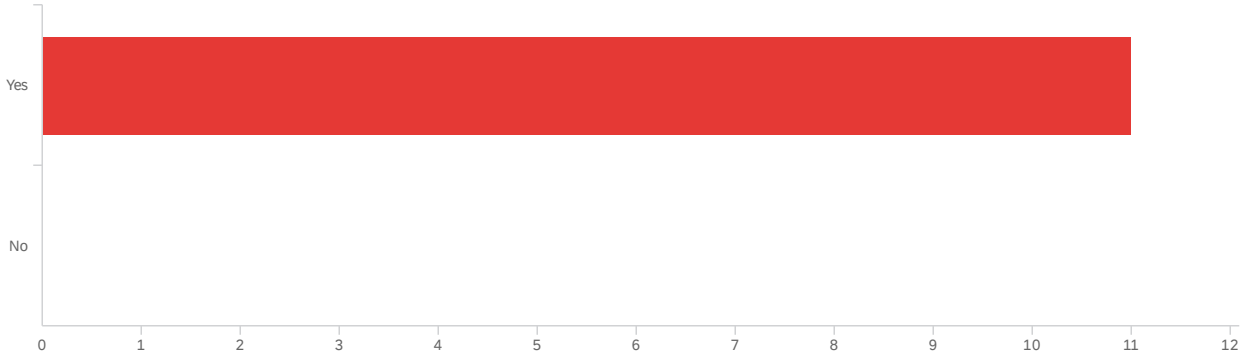


Q51 - (Optional) Feel free to explain your reasoning

(Optional) Feel free to explain your reasoning

Q16 - Q. Is the formula next_state (next_state (eventually (Engine))) satisfied by this trace?*

correct = Yes, trace = {RGB} {RGB} {RGB} {RGB} {RGB}*

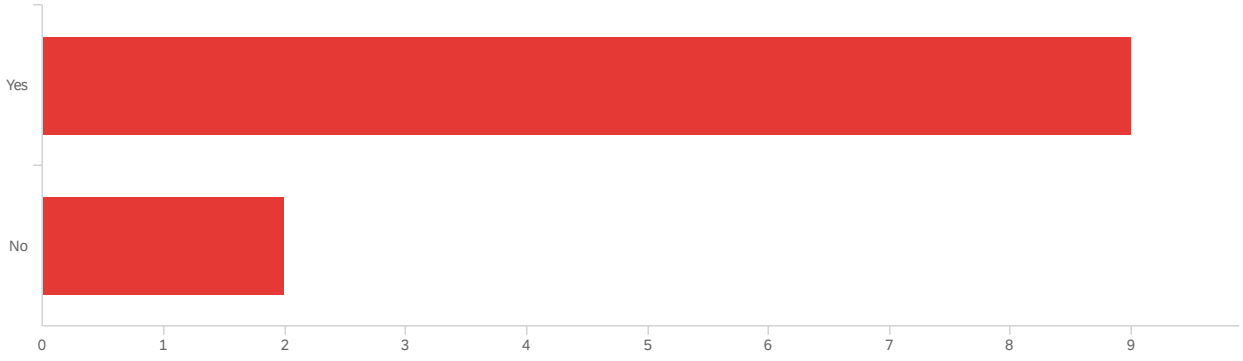


Q52 - (Optional) Feel free to explain your reasoning

(Optional) Feel free to explain your reasoning

Q17 - Q. Is the formula (Engine) until (Light) satisfied by this trace?*

correct = No, trace = {R} {R} {R} {R} {R}*



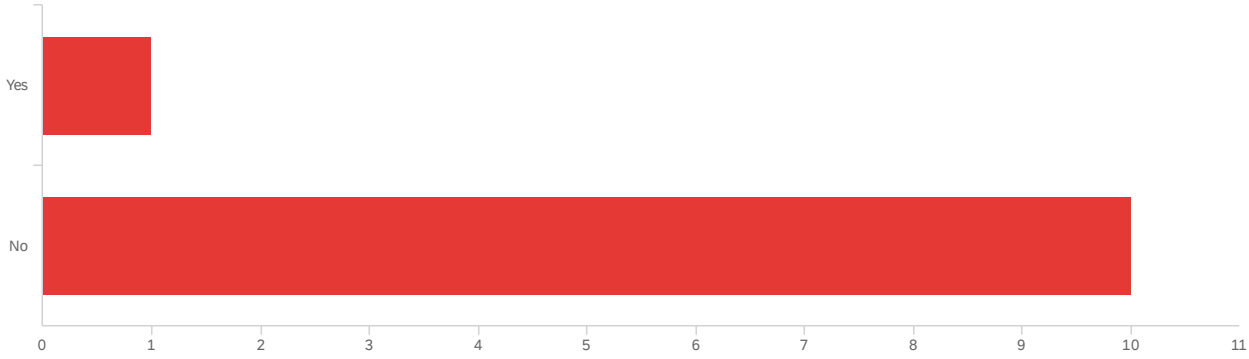
Q53 - (Optional) Feel free to explain your reasoning

(Optional) Feel free to explain your reasoning

Changed after reading page 2 example — example suggests that in A until B, B must occur

Q18 - Q. Is the formula eventually (always (Engine)) satisfied by this trace?*

correct = No, trace = {} {RGB} {} {RGB} {}

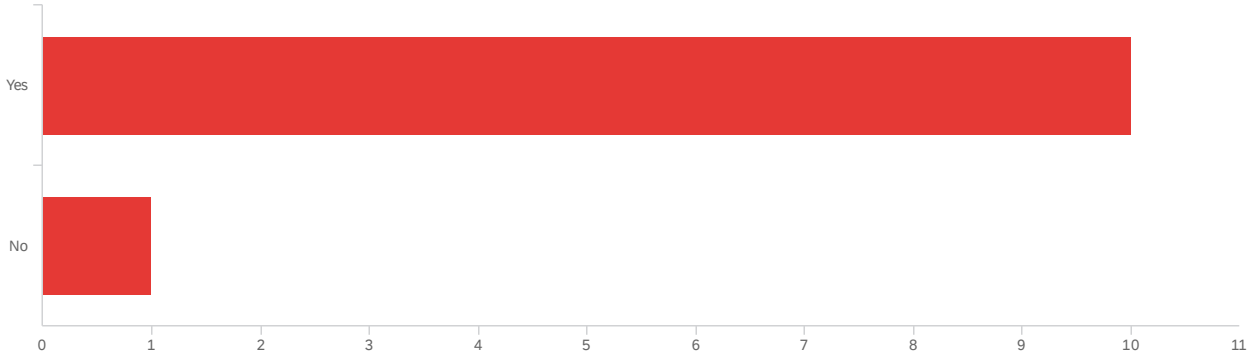


Q54 - (Optional) Feel free to explain your reasoning

(Optional) Feel free to explain your reasoning

Q19 - Q. Is the formula always (Engine implies Light) satisfied by this trace?*

correct = Yes, trace = {} {} {} {} {}*

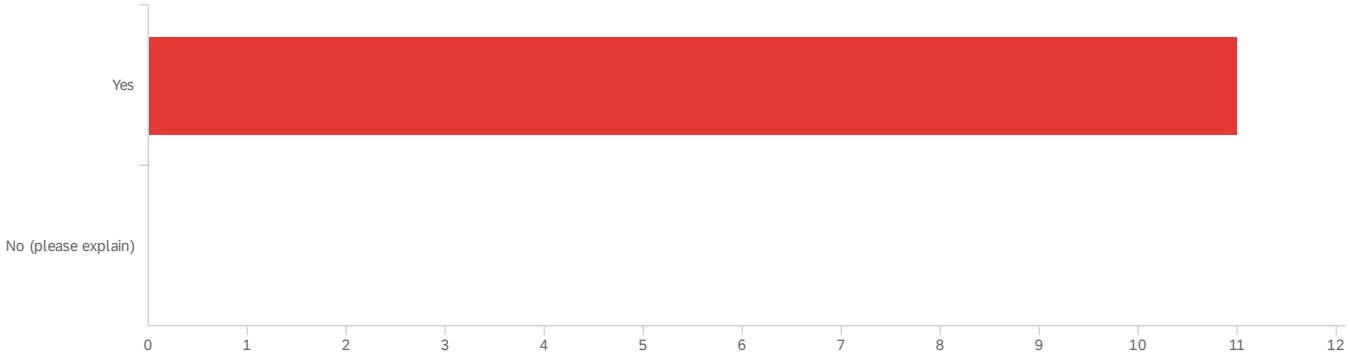


Q55 - (Optional) Feel free to explain your reasoning

(Optional) Feel free to explain your reasoning

Q23 - Does the example make sense to you?*

expected = Yes

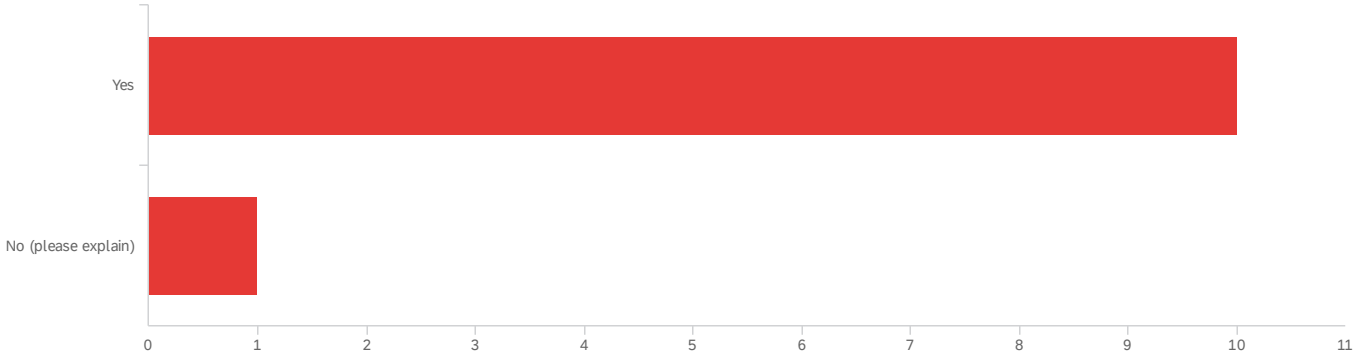


Q23_3_TEXT - No (please explain)

No (please explain)

Q25 - Does the example make sense to you?*

expected = Yes



Q25_3_TEXT - No (please explain)

No (please explain)
I'm not sure if Engine "on" after the until predicate is satisfied makes the formula is true or false.

Q27 - Red implies after (after (after (Red)))

correct = "If Red initially, then Red again in 4th state."

Engine implies next_state (next_state (next_state (Engine...
Whenever the engine is on, the engine is also on 3 states later.
If the engine is on, it is on in the next of the next of the next state.
If the engine is on, then the engine is on 3 states in the future.
Always in all states, if the engine is on then three states later it should be on.
If the engine is on, the 3rd state from that state will have the engine on.
If the engine is on, the engine will be on three states from now.
When the engine is on at a certain state, the engine will also be on 3 states after
If the engine is on, the engine will be on 3 states from now
If the Engine is on in the current state, three states forward from now, the Engine must also be on - for this to be satisfied. Or the Engine could never be on.
Whenever the engine is on, three states later the engine will be on.
If the engine is on now, it will be on three states later.

Q56 - (Optional) Feel free to explain your reasoning

(Optional) Feel free to explain your reasoning

I think this means every third state will have the same engine truth value.

3 after() -> 3 states Engine = engine is on at some state after(after(after(Engine))) engine is again on

Q28 - after (after (eventually (after (Red))))

correct = "Red at 4th state or later"

next_state (next_state (eventually (next_state (Engine)...

From the third state on, at some point the engine will be on in s+1 following some state s (s >= 3).

two states after the current state, there exists some state S that the engine is on in the next next of S.

Starting 2 states from now, we will get to a state where the engine turns on in the next state.

After two time steps, eventually the engine turns on.

There is a state such that in two states, there will eventually be a state such that the next state has the engine on.

After three states from now, the engine will eventually be on. OR After two states from now, the engine will eventually be on in the next state.

There will be a certain point where the engine is on, 2 states after the current state.

sometime after two states from now, the next state will have the engine on

Three states from initial: from this point, the Engine must eventually turn on. If the Engine is on before the third state, it does not affect the satisfiability of the statement.

After two states have passed, eventually the Engine after these the next state will be on.

From the third state onwards, eventually there will be a next state where the engine is on.

Q57 - (Optional) Feel free to explain your reasoning

(Optional) Feel free to explain your reasoning

Q29 - { eventually (Red) } implies { always (Blue) }

correct = "If ever Red, then Blue always from first state"

{ eventually (Engine) } implies { always (Light) }

If the engine is on in some state, this implies that the light will always be on in all states.

If the engine is on for some state, then the light is on for all states.

If the engine eventually turns on, then the light is always on.

If the engine turns on, the light should always be on.

If the engine is on at some point during the trace, all states will have the lights on.

If the engine is on at some point, then the light is always on.

If the engine is on at a certain point after the current state, the light should always be on.

if the engine is turned on (or going to be turned on) at any state, the light is/was always on.

If at any point the Engine turns on, the Light must be on throughout the entire trace. Otherwise, the Engine can be always off, and the trace satisfies the constraints.

At some state in the future the Engine is on, and at that point the light will always be on after that state.

If at some point the engine turns on, then every state has the light on.

Q58 - (Optional) Feel free to explain your reasoning

(Optional) Feel free to explain your reasoning

Q30 - ((Red) until (Blue)) and always (Red)

correct = "Red forever, Blue eventually"

((Engine) until (Light)) and always (Engine)

The engine will be on in all states before a light is ever on, and the engine will also always be on. This implies that the light will never be on, since the engine must always be on, and the engine can only be on until the light is on.

Engine is always on and light is never on because if the light is on then the engine should turn off.

If the light hasn't been turned on yet, the engine is on, and the engine is always on.

Engine is always on and eventually the light turns on.

The engine will always be on and the light cannot turn on.

The engine is always on, and the light eventually turns on.

Engine is always on, light doesn't matter

the engine will be on until the light turns on, and the engine is always on

The Engine must be on until the Light turns on, and the Engine must always be on. The Engine must always be on, and the Light can come on at some point and that trace will satisfy.

Engine is always on + light is never on at any point

The engine is always on, and since we engine until light, the light is never on.

Q59 - (Optional) Feel free to explain your reasoning

(Optional) Feel free to explain your reasoning

the first expression seems to be within the constraint of the second expression, so I ignored it

Q31 - always (Red implies (after (not Red) and after (after (Red))))

correct = "If Red then blinks forever"

always (Engine implies (next_state (not Engine) and n...

If the engine is on, then the engine is not on in the next state; the engine must be on, though, in the state following that.

For every state, If the engine is on, then it must be turned off in the next state, and it is on at two states after the current state.

For any state, if the engine is on: the engine has to be off for the next state, but has to be back on 2 states from now.

Once (not guaranteed) the engine turns on, it starts to alternate between turning on an off.

At any point, if the engine is on, all future states will alternate between the engine being off and on.

If the engine is on, the engine will be off in the next state, and then turn on again in the following state. (This forms an on-off cycle.)

For the entirety of the timeline, if a certain state has engine on, then the next state would not have engine on, but two states after the engine will be on. --> Engine will be on every other state, and off every other state, as long as there is engine somewhere

if the engine is on in a given state, it must be off in the next state, and on in the next next state (engine alternates between being on and off)

If the Engine ever turns on, it must turn on and off sequentially.

If the engine is on at some state, the next state afterwards it is off, but two states afterwards the engine is again on.

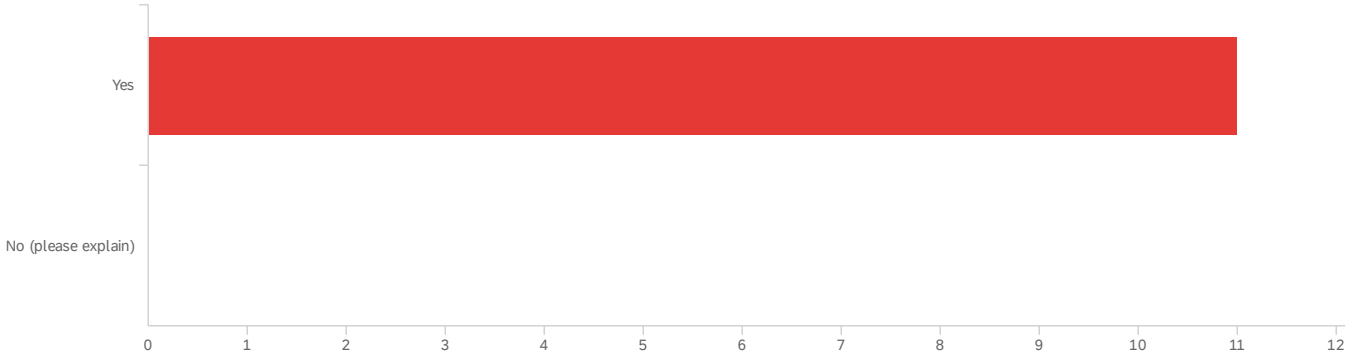
If the engine is on, then in the next state the engine is not on and two states later it turns back on. So once the engine starts then it will alternate between on and off.

Q60 - (Optional) Feel free to explain your reasoning

(Optional) Feel free to explain your reasoning

Q35 - Does the example make sense to you?*

expected = Yes



Q37 - Whenever the Red light is on, it is off in the next state and on again in the state after that.

correct = G(Red => X(!Red) and XX(Red))

Whenever the engine is on, it is off in the next state and on again in the...
always { Engine implies { next_state {not Engine} and next_state { next_state { Engine } } } }
always (engine implies (next_state (not engine and next_state(engine))))
always (Engine implies { next_state(not Engine) and next_state(next_state(Engine)) })
always {Engine => ((next_state !Engine) and (next_state next_state Engine))}
Engine implies { {next_state not engine} and {next_state next_state engine} }
always { Engine => next_state not Engine} and {next_state next_state Engine }
always { engine implies { next_state{not engine} and next_state{next_state{engine
always (engine => (next_state (not engine) and next_state(next_state(engine))))
always{ Engine implies next_state no Engine} and {next_state next_state Engine}
always (Engine implies (after(not Engine) and after(after(Engine))))
always(engine implies (after(not engine) and after(after(engine))))

Q61 - (Optional) Feel free to explain your reasoning

(Optional) Feel free to explain your reasoning

Q38 - The Red light is on in exactly one state, but not necessarily the first state.

correct = { !Red } U { Red and X(G(!Red)) } ... OR ... F(Red) and G(Red => X(G(!Red)))

The engine is on in exactly one state, but not necessarily the first state.

eventually { Engine and next_state {always{not Engine}

eventually (engine and next_state(always(not engine)))

(not Engine) until Engine and always (Engine implies always (not Engine))

(eventually Engine) and (Engine => next_state always !Engine)

{Engine implies { always {next_state not engine} and {eventually engine}

{always {Engine => next_state {not eventually Engine} and {eventually Engine}

{engine implies always {not engine and {not engine implies eventually {engine

always (engine => next_state(always not engine))

eventually{Engine}

eventually (Engine and (after(always(not Engine)))

i dont know

Q62 - (Optional) Feel free to explain your reasoning

(Optional) Feel free to explain your reasoning
Assuming always{} will be looking forward from the current state on.
not sure if it guarantees exactly one state

Q39 - The Red light cannot stay on for three states in a row.

correct = G(Red and X(Red) => XX(!Red)) ... OR ... !F(Red and X(Red) and XX(Red)) ... ETC

The engine cannot stay on for three states in a row.
always { not { Engine and next_state{Engine} and next_state{next_state{Engine
always(engine and (next_state(not engine)or next_state(next_state(not engine))))
always ((Engine and next_state(Engine)) implies (not next_state(next_state(Engine))))
always !(Engine and (next_state Engine) and (next_state next_state Engine))
engine implies { next_state next_state next_state not engine }
always Engine and {next_state Engine => next_state next_state not Engine}
always { not {engine and next_state{engine} and next_state{next_state{engine} }
always (not (engine and next_state(engine) and next_state(next_state(engine))))
always{ {Engine and next_state Engine} implies {next_state next_state no Engine
Engine implies after(after(not Engine))
not (eventually (engine and after(engine) and after(after(engine))))

Q63 - (Optional) Feel free to explain your reasoning

(Optional) Feel free to explain your reasoning

Q40 - Whenever the Red light is on, the Blue light will be on then or at some point in the future.

correct = G(Red => F(Blue))

Whenever the engine is on, the light will be on then or at some point in th...

engine implies { eventually light }
always{ Engine implies {eventually Light
always(engine implies(light or eventually(light)))
always(engine implies (light or eventually(light)))
always {Engine implies eventually{Light
always {Engine => eventually Light}
always { Engine => (Light or eventually Light) }
always { engine implies {light or eventually light} }
always (engine implies { Light or eventually Light })
always (engine => (light or eventually (light)))
(Engine and Light) or (Engine and eventually(Light))

Q64 - (Optional) Feel free to explain your reasoning

(Optional) Feel free to explain your reasoning

Assuming eventually looks forward from current state, and not starting from the beginning Also assuming eventually can be satisfied by the current state

Q41 - The Red light is on for zero or more states, and then turns off and remains off in the future.

correct = Red U G(!Red) ... OR ... F(!Red) and G(!Red => X(G(!Red)))

The engine is on for zero or more states, and then turns off and remains of...
Engine implies {next_state {always{not Engine}
(engine until not engine) and eventually (always (not engine))
(Engine until not Engine) and always (not Engine implies always (not Engine))
eventually (always !Engine)
engine implies { always {next_state not engine} }
always not Engine} => next_state {not eventually Engine
eventually {always {not engine
(engine until (not engine)) and (not engine => always (not (engine)))
always{ Engine implies eventually{always no Engine
eventually(Engine and after(always(not Engine))
always(engine implies eventually(always(not engine)))

Q65 - (Optional) Feel free to explain your reasoning

(Optional) Feel free to explain your reasoning

Assuming that "always" is looking at the current state forward

End of Report