<table>
<tr><th>ID</th><th>Natural-Language</th><th>FRETISH</th><th>Template Key</th><th>PCTL*</th></tr>
<tr><td>[P-001]</td><td>The sensor selection component shall always satisfy accurate under ideal conditions (e.g. perfect weather, perfect lighting, on the ground, perfectly marked airport, stationary, etc.)</td><td>Whenever idealConditions SensorSelection shall immediately satisfy q_hat = q</td><td>[null, holding, null, immediately]</td><td>P>=1[(G (idealConditions => (P>=1[(q_hat = q)])))]</td></tr>
<tr><td>[P-002]</td><td>Upon one or more LECs are inaccurate, sensor selection component output shall with probability greater than 95% at the next time point satisfy accurate</td><td>Upon LECs_inaccurate SensorSelection shall with probability >0.95 at the next timepoint satisfy q_hat = q</td><td>[null, regular, bound, next]</td><td>P>=1[((G (((! LECs_inaccurate) & (X LECs_inaccurate)) => (X (P>0.95[(X (q_hat = q))])))) & (LECs_inaccurate => (P>0.95[(X (q_hat = q))])))]</td></tr>
<tr><td>[P-003]</td><td>In auto-taxi mode, sensor selection component output shall with probability greater than 95% always satisfy accurate</td><td>in auto_taxi_mode SensorSelection shall with probability > 0.95 always satisfy q_hat = q</td><td>[in, null, bound, always]</td><td>P>=1[((G ((! (((! auto_taxi_mode) & (X auto_taxi_mode)))) | (X (P>0.95[((auto_taxi_mode & (X (! auto_taxi_mode))) R (q_hat = q))])))) & (auto_taxi_mode => ((auto_taxi_mode & (X (! auto_taxi_mode))) | (P>0.95[((auto_taxi_mode & (X (! auto_taxi_mode))) R (q_hat = q))]))))]</td></tr>
<tr><td>[P-004]</td><td>In auto-taxi mode, whenever the LECs disagree, the sensor selection output shall with probability > 0.99 within 5 ticks satisfy accurate</td><td>in auto_taxi_mode whenever LEC_disagreement SensorSelection shall with probability > 0.99 within 5 ticks satisfy q_hat = q</td><td>[in, holding, bound, within]</td><td>P>=1[((G ((! (((! auto_taxi_mode) & (X auto_taxi_mode)))) | (X ((auto_taxi_mode & (X (! auto_taxi_mode))) R (LEC_disagreement => (P>0.99[(F<=5 (q_hat = q))]))))))) & (auto_taxi_mode => ((auto_taxi_mode & (X (! auto_taxi_mode))) | ((auto_taxi_mode & (X (! auto_taxi_mode))) R (LEC_disagreement => (P>0.99[(F[<=5] (q_hat = q))])))))))]</td></tr>
<tr><td>[P-006]</td><td>In auto-takeoff mode, whenever runway incursion occurs, the sensor selection output shall with probability > 0.99 at the next time point satisfy detection of runway incursion.</td><td>in auto_takeoff_mode whenever q_k SensorSelection shall with probability > 0.99 at the next timepoint satisfy incursionDetected</td><td>[in, holding, bound, next]</td><td>P>=1[((G ((! (((! auto_takeoff_mode) & (X auto_takeoff_mode)))) | (X ((auto_takeoff_mode & (X (! auto_takeoff_mode))) R (q_k => (P>0.99[((auto_takeoff_mode & (X (! auto_takeoff_mode))) | ((X incursionDetected) & (! (auto_takeoff_mode & (X (! auto_takeoff_mode))))))]))))))) & (auto_takeoff_mode => ((auto_takeoff_mode & (X (! auto_takeoff_mode))) | ((auto_takeoff_mode & (X (! auto_takeoff_mode))) R (q_k => (P>0.99[((auto_takeoff_mode & (X (! auto_takeoff_mode))) | ((X incursionDetected) & (! (auto_takeoff_mode & (X (! auto_takeoff_mode))))))])))))))]</td></tr>
<tr><td>[P-007]</td><td>after auto-land mode, the sensor selection output shall with probability > 0.99 eventually satisfy detection of correct runway exit</td><td>after auto_land_mode SensorSelection shall with probability > 0.99 eventually satisfy detect_correct_exit</td><td>[after, null, bound, eventually]</td><td>P>=1[(((! (auto_land_mode & (X (! auto_land_mode)))) U ((auto_land_mode & (X (! auto_land_mode))) & (X (P>0.99[(F detect_correct_exit)])))) | (G (! (auto_land_mode & (X (! auto_land_mode))))))]</td></tr>
<tr><td>[P-008]</td><td>before auto-takeoff mode, the sensor selection output shall with probability > 0.99 eventually satisfy detection of correct runway entry</td><td>before auto_takeoff_mode SensorSelection shall with probability > 0.99 eventually satisfy detect_correct_entry</td><td>[before, null, bound, eventually]</td><td>P>=1[(((! auto_takeoff_mode) & (X auto_takeoff_mode)) | (P>0.99[((! ((! auto_takeoff_mode) & (X auto_takeoff_mode))) U detect_correct_entry)]))]</td></tr>
<tr><td>[P-009]</td><td>Upon out-of-distribution_event OODDetector shall with probability ≥ 0.9 after 5 time units satisfy InputDataIsOOD</td><td>upon ood_event OODDetector shall with probability ≥ 0.99 after 5 ticks satisfy f_k</td><td>[null, regular, bound, after]</td><td>P>=1[((G (((! ood_event) & (X ood_event)) => (X (P>=0.99[((G<=5 (! f_k)) & (F<=(5+1) f_k))])))) & (ood_event => (P>=0.99[((G[<=5] (! f_k)) & (F[<=5+1] f_k))])))]</td></tr>
<tr><td>[P010]</td><td>In auto-land mode, Upon out-of-distribution_event OODDetector shall with probability ≥ 0.99 after 5 time units satisfy InputDataIsOOD</td><td>in auto_land_mode upon ood_event OODDetector shall with probability ≥ 0.99 after 5 ticks satisfy f_k</td><td>[in, regular, bound, after]</td><td>P>=1[((G ((! (((! auto_land_mode) & (X auto_land_mode)))) | (X (((auto_land_mode & (X (! auto_land_mode))) R (((! ood_event) & ((X ood_event) & (! (auto_land_mode & (X (! auto_land_mode)))))) => ((X (P>=0.99[((G<=5 (! f_k)) & (F<=(5+1) f_k))])) & (! (auto_land_mode & (X (! auto_land_mode))))))) & (ood_event => (P>=0.99[((G[<=5] (! f_k)) & (F[<=5+1] f_k))]))))))) & (auto_land_mode => ((auto_land_mode & (X (! auto_land_mode))) | (((auto_land_mode & (X (! auto_land_mode))) R (((! ood_event) & ((X ood_event) & (! (auto_land_mode & (X (! auto_land_mode)))))) => ((X (P>=0.99[((G[<=5] (! f_k)) & (F[<=5+1] f_k))])) & (! (auto_land_mode & (X (! auto_land_mode))))))) & (ood_event => (P>=0.99[((G[<=5] (! f_k)) & (F[<=5+1] f_k))]))))))]</td></tr>
<tr><td>[P-011]</td><td>Upon inaccurate, the Runway Detector with probability greater than 90% remain inaccurate for 10 time units.</td><td>upon q_hat - q > epsilon & q - q_hat < epsilon RunwayDetector shall with probability ≥ 0.9 for 10 ticks satisfy q_hat - q > epsilon & q - q_hat < epsilon</td><td>[null, regular, bound, for]</td><td>P>=1[((G (((! (((q_hat - q) > epsilon) & ((q - q_hat) < epsilon))) & (X (((q_hat - q) > epsilon) & ((q - q_hat) < epsilon)))) => (X (P>0.9[(G<=10 (((q_hat - q) > epsilon) & ((q - q_hat) < epsilon)))])))) & ((((q_hat - q) > epsilon) & ((q - q_hat) < epsilon)) => (P>0.9[(G[<=10] (((q_hat - q) > epsilon) & ((q - q_hat) < epsilon)))])))]</td></tr>
<tr><td>[P012]</td><td>Upon a runway incursion, the Runway Intrusion Detector, with probability greater than 99.99%, detects the incursion before an unsafe separation distance is reached.</td><td>upon q_k RunwayIntrusionDetector shall with probability > 0.9999 before unsafe_sep_distance satisfy incursionDetected</td><td>[null, regular, bound, before]</td><td>P>=1[((G (((! q_k) & (X q_k)) => (X (P>0.9999[(incursionDetected R (! unsafe_sep_distance))])))) & (q_k => (P>0.9999[(incursionDetected R (! unsafe_sep_distance))])))]</td></tr>
</table>

TABLE I: FRETISH and generated PCTL* formulas for RTX perception system case study.

ID	Natural-Language	FRETISH	Template Key	PCTL*
[R013]	*Upon out-of-distribution_event OODDetector shall with probability $\geq$ 0.9 after 5 time units satisfy InputDataIsOOD*	upon ood_event OODDetector shall with probability $\geq$ 0.99 after 5 ticks satisfy f_k	[null, regular, bound, after]	P>=1[((G (((! ood_event) & (X ood_event)) => (X (P>=0.99[((G<=5 (! f_k)) & (F<=6 f_k))])))) & (ood_event => (P>=0.99[((G[<=5] (! f_k)) & (F[<=6] f_k))])))]
[R014]	*In auto-land mode, Upon out-of-distribution_event OODDetector shall with probability $\geq$ 0.99 after 5 time units satisfy InputDataIsOOD*	in auto_land_mode upon ood_event OODDetector shall with probability $\geq$ 0.99 after 5 ticks satisfy f_k	[in, regular, bound, after]	P>=1[((G ((! (((! auto_land_mode) & (X auto_land_mode)) & (! false))) | (X (((auto_land_mode & (X (! auto_land_mode))) R (((! ood_event) & ((X ood_event) & (! (auto_land_mode & (X (! auto_land_mode)))))) => ((X (P>=0.99[((G<=5 (! f_k)) & (F<=6 f_k))])) & (! (auto_land_mode & (X (! auto_land_mode))))))) & (ood_event => (P>=0.99[((G[<=5] (! f_k)) & (F[<=6] f_k))])))))) & (auto_land_mode => ((auto_land_mode & (X (! auto_land_mode))) | (((auto_land_mode & (X (! auto_land_mode))) R (((! ood_event) & ((X ood_event) & (! (auto_land_mode & (X (! auto_land_mode)))))) => ((X (P>=0.99[((G[<=5] (! f_k)) & (F[<=6] f_k))])) & (! (auto_land_mode & (X (! auto_land_mode))))))) & (ood_event => (P>=0.99[((G[<=5] (! f_k)) & (F[<=6] f_k))]))))))]
[R015]	*Upon inaccurate, the Runway Detector with probability greater than 90% remain inaccurate for 10 time units.*	upon q_hat - q > epsilon & q - q_hat < epsilon RunwayDetector shall with probability > 0.9 for 10 ticks satisfy q_hat - q > epsilon & q - q_hat < epsilon	[null, regular, bound, for]	P>=1[((G (((! (((q_hat - q) > epsilon) & ((q - q_hat) < epsilon))) & (X (((q_hat - q) > epsilon) & ((q - q_hat) < epsilon)))) => (X (P>0.9[(G<=10 (((q_hat - q) > epsilon) & ((q - q_hat) < epsilon)))])))) & ((((q_hat - q) > epsilon) & ((q - q_hat) < epsilon)) => (P>0.9[(G[<=10] (((q_hat - q) > epsilon) & ((q - q_hat) < epsilon)))])))]
[R016]	*Upon a runway incursion, RunwayIntrusionDetector, with probability greater than 99.99%, detects the incursion before an unsafe separation distance is reached.*	upon q_k RunwayIntrusionDetector shall with probability > 0.9999 before unsafe_sep_distance satisfy incursionDetected	[null, regular, bound, before]	P>=1[((G (((! q_k) & (X q_k)) => (X (P>0.9999[((incursionDetected | false) R (! unsafe_sep_distance))])))) & (q_k => (P>0.9999[((incursionDetected | false) R (! unsafe_sep_distance))])))]
[R017]	*Whenever a runway incursion, RunwayIntrusionDetector, with probability greater than 99.99%, detects an incursion within 10 time units.*	whenever q_k RunwayIntrusionDetector shall with probability > 0.9999 within 10 ticks satisfy incursionDetected	[null, holding, bound, within]	P>=1[(G (q_k => (P>0.9999[(F<=10 incursionDetected)])))]

TABLE II: FRETISH and generated PCTL* formulas for RTX perception system component.