|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Property**  **Type** | **Property Description** | **TCTL Property** | **Verification Result** | | **Verification Time** | |
| **UPPAAL** | **FDR** | **UPPAAL** | **FDR** |
| FootBot | Safety | Deadlock-free. | A[] not deadlock | **Sat.** | **Sat.** | **0.004s** | **0.7468s** |
| No outputs conflict. | A[] not (CFootBot\_cycle.Output\_Conflict || Movement\_cycle.Output\_Conflict || SimSMovement\_cycle.Output\_Conflict) | **Sat.** | **Sat.** | **0.002s** | **0.7468s** |
| Reachability | State *SMove* is reachable. | E<> SimSMovement.entered\_SMove | **Sat.** | **Sat.** | **0.000s** | **0.4327s** |
| Alpha  Algorithm | Safety | Deadlock-free. | A[] not deadlock | **Sat.** | **Sat.** | **44.990s** | **67.526s** |
| No outputs conflict. | A[] not (AggregationSoftware\_cycle.Output\_Conflict || MovementC\_cycle.Out put\_Conflict || …) | **Sat.** | **Sat.** | **32.940s** | **68.321s** |
| No more than *RC* time units are spent in the state *Receive*. | A[] Communication.j0 imply Communication.RCC <= Communication.RC | **Sat.** | **Sat.** | **30.820s** | **72.349s** |
| The state machine *Communication* starts to enter the state *Receive* exactly every *RC* units. | A[] (Communication.entered\_Broadcast && Communication.RCC>0) imply Comm unication.RCC == Communication.RC | **Sat.** | **Sat.** | **30.220s** | **71.203s** |
| No more than *360/av* time units are spent in the state *Turning*. | A[] Movement.tj1 imply Movement.MBC <= 360/Movement.av | **Sat.** | **Sat.** | **31.640s** | **78.900s** |
| Reachability | State *SMove* is reachable. | E<> Movement.entered\_SMove | **Sat.** | **Sat.** | **1.690s** | **47.100s** |
| State *Turn180* is reachable. | E<> Movement.entered\_Turn180 | **UnSat.** | **UnSat.** | **30.630s** | **54.672s** |
| Liveness | The behaviour of communication should consist of a *broadcast* followed by a *receive* event. | A<> (broadcast\_Communication.b == true && broadcast\_Communication.id != Communication.id) imply receive\_Communication\_CommunicationC == true | **Sat.** | **Sat.** | **0.010s** | **55.549s** |
| The robot should start *moving*, and after every *obstacle*, *move* at least once. | obstacle\_ePuck.b == true --> move\_ePuck.b == true | **Sat.** | **Sat.** | **31.380s** | **61.801s** |
| Initially, and after exactly *RC* time units, when a *broadcast.id* happens, the events *broadcast* and *receive* are offered before *RC* time units elapse. | (Communication.entered\_Broadcast && Communication.RCC>0) --> (broadcast\_Communication\_CommunicationC == true || receive\_Communication\_CommunicationC == true) | **Sat.** | **Sat.** | **66.800s** | **66.636s** |

We used a machine with quad-core Intel i7 5500U, and 8 GB of RAM in an Ubuntu system.

We ran each experiment 10 times, and the verification time given is the average.