Model-based development and formal verification of a ROS2 multi-robot system using Timed Rebeca

Hiep Hong Trinh[†*], Marjan Sirjani[‡*], Federico Ciccozzi[*] and Mikael Sjödin[*]

Abstract— **Model-based development allows faster prototyping, earlier experimentation and validation of design intents. For a multi-agent system with complex asynchronous interactions and concurrency, model checking is a stronger and automated mechanism for verifying desired system properties. Timed Rebeca is an actor-based modelling language supporting both concurrent and time semantics, accompanied with a model-checking compiler. These capabilities allow using Timed Rebeca to correctly model ROS2 node topography, recurring physical signals, motion primitives and other timed and time-convertible behaviours. In this work we modelled a multiple autonomous mobile robots system in Timed Rebeca then developed corresponding ROS2 simulation code, ensured semantic synchronization between the model and the code, and set up experiments to use the model for revealing problems that do not always show up in simulation and verifying different properties (goal reachability, collision freedom, deadlock freedom, arrival times). The biggest challenges lie in abstracting complex information in robotics, bridging the gap between a discrete model and a continuous system and minimizing the state space, while maintaining the model's accuracy. We devised different discretization strategies for different kinds of information, identifying the 'enough' thresholds of abstraction, and applying efficient optimization techniques to boost computations to reduce model checking time. The benefits of formal verification are clear, however applying them in practice is difficult; our approach is a well explained, easy to understand, working solution in a realistic context.**

I. SUMMARY

A model is a purposeful abstraction of a physical system that allows engineers to reason about some aspects of that system by focusing only on relevant details while ignoring extraneous ones [2]. Formal verification is "the use of mathematical techniques to ensure that a design conforms to some precisely expressed notion of functional correctness" [1]. Model checking is a formal method that allows automatic verification of a finite-state concurrent system by using a discrete model of the system [3]. Model checking always gives a consistent yes or no answer; if it fails, it will always fail the same way.

Rebeca[1] is an actor-based language for modelling asynchronous, concurrent systems. Timed Rebeca adds language constructs for modelling timing constraints of real-time systems including execution time, expiry time and scheduled time [4], [7]. Rebeca has an object-based syntax and provides a model-checking engine and an IDE to help build and check models conveniently [6]. The software controlling a robotic system, which may comprise multiple robots working together, has to be predictable and reliable in terms of functional and safety. Testing through simulation or real deployment can only be done in a later stage of development while exhaustive testing is impossible; furthermore testing is extremely hard for a concurrent system. Modelling and model-checking help to systematically verify the system's logic and timing correctness early in the process, thus reducing development cost and increasing safety assurance. For reactive systems like ROS2 applications, a dedicated modelling language and tool is needed.

In this work[2], we used Timed Rebeca to holistically model a multiple autonomous mobile robots system[3], developed corresponding ROS2 simulation code on RViz2[4], synchronized the model and the code in semantics, and ran model-checking to test behavioral algorithms, discover working parameters and their safety thresholds, verify different properties (goal reachability, deadlock freedom, collision freedom, arrival times). The simulation program was then executed multiple times under the same settings to prove that finally it encountered the same problems that were predicted by model checking. Experiments showed that some system problems are not always detected through simulation but consistently revealed by model-checking (see Table I, Fig.1, Fig.2).

TABLE I

NON-WORKING CASES

Parameters	**Case C2a**		**Case C2b**		**Case C2c**	
Scan rate	140		100		140	
Speed limit	0.91		1.275		0.91	
Stop zone	0.3		0.3		0.3	
Robot	***Speed***	***Wait***	***Speed***	***Wait***	***Speed***	***Wait***
R1	0.9	1500	0.5	1500	0.9	1500
R2	0.9	2000	0.5	1500	0.8	2000
R3	0.9	2500	0.5	1500	0.7	2500
R4	0.9	3000	0.5	1500	0.6	3000
R5	0.9	1500	0.5	1500	1.0	1500
Analysis result	Assertion failed (collision)		Assertion failed (collision)		Assertion failed (collision)	
States	3074		6816		1740	
Transitions	6602		15293		3727	
Simulations	5		5		5	
Simulation results	2/5 passed 3/5 failed		3/5 passed 2/5 failed		0/5 passed 5/5 failed	

[†]Principal investigator hiep.hong.trinh@mdu.se

[‡]Coordinating author marjan.sirjani@mdu.se

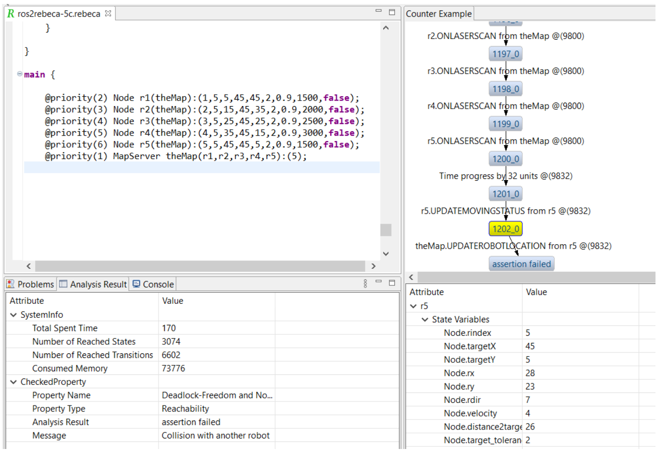
[*]School of IDT, Mälardalen University (Västerås, Sweden)

[1] https://rebeca-lang.org/

[2] Mainly based on the thesis report in [8]

[3] https://github.com/thhiep/ros2rebeca_model (public access)

[4] https://github.com/thhiep/ros2rebeca_code (limited access)

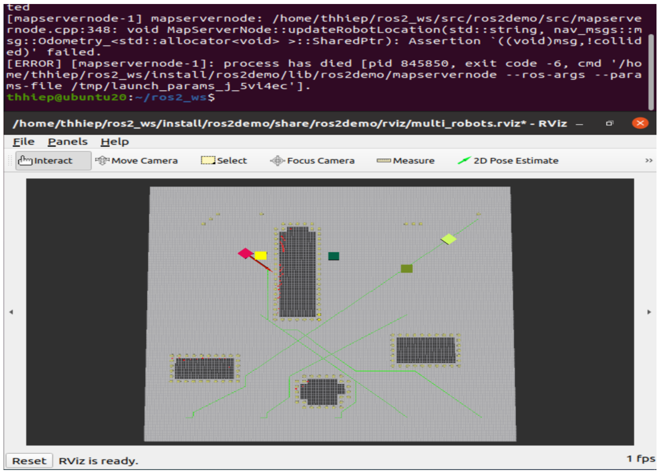

ros2rebeca-5c.rebeca
Counter Example
r2.ONLASERSCAN from theMap @(9800)
1197_0
r3.ONLASERSCAN from theMap @(9800)
1198_0
r4.ONLASERSCAN from theMap @(9800)
1199_0
r5.ONLASERSCAN from theMap @(9800)
1200_0
Time progress by 32 units @(9832)
1201_0
r5.UPDATEMOVINGSTATUS from r5 @(9832)
1202_0
theMap.UPDATEROBOTLOCATION from r5 @(9832)
assertion failed
Problems
Analysis Result
Console

```
        }

    }

main {

        @priority(2) Node r1(theMap):(1,5,5,45,45,2,0.9,1500,false);
        @priority(3) Node r2(theMap):(2,5,15,45,35,2,0.9,2000,false);
        @priority(4) Node r3(theMap):(3,5,25,45,25,2,0.9,2500,false);
        @priority(5) Node r4(theMap):(4,5,35,45,15,2,0.9,3000,false);
        @priority(6) Node r5(theMap):(5,5,45,45,5,2,0.9,1500,false);
        @priority(1) MapServer theMap(r1,r2,r3,r4,r5):(5);
```

Attribute	Value
SystemInfo	
Total Spent Time	170
Number of Reached States	3074
Number of Reached Transitions	6602
Consumed Memory	73776
CheckedProperty	
Property Name	Deadlock-Freedom and No...
Property Type	Reachability
Analysis Result	assertion failed
Message	Collision with another robot

Attribute	Value
r5	
State Variables	
Node.rindex	5
Node.targetX	45
Node.targetY	5
Node.rx	28
Node.ry	23
Node.rdir	7
Node.velocity	4
Node.distance2targe	26
Node.target_toleran	2

Fig. 1. Screenshots of model checking results (non-working)



```
ted
[mapservernode-1] mapservernode: /home/thhiep/ros2_ws/src/ros2demo/src/mapserve
rnode.cpp:348: void MapServerNode::updateRobotLocation(std::string, nav_msgs::m
sg::Odometry_<std::allocator<void> >::SharedPtr): Assertion `((void)msg,!collid
ed)' failed.
[ERROR] [mapservernode-1]: process has died [pid 845850, exit code -6, cmd '/ho
me/thhiep/ros2_ws/install/ros2demo/lib/ros2demo/mapservernode --ros-args --para
ms-file /tmp/launch_params_j_5vi4ec'].
thhiep@ubuntu20:~/ros2_ws$
```

Fig. 2. Screenshots of simulation results (non-working)

The hardest challenges in modelling and model-checking a robotic system, especially a multi-robot one, are: abstracting complex robotic information, dealing with the gap when using a discrete model to model continuous changes, identifying the 'enough' thresholds of abstraction to minimize the state space while maintaining the model's accuracy, optimizing computations to reduce model checking time. We devised different discretization strategies for different kinds of information and set up experiments to test the 'enough' thresholds and verify behavioural equivalence between the model and the program. We also devised a human-like safe walking and backoff algorithm for avoiding collision and getting out of congestion.

In summary, our techniques for modelling different parts of a multiple mobile robots system are:

- Project static and mobile obstacles onto a 2D occupancy grid, and reason why grid resolution matters modelling accuracy
- Use A* algorithm to generate paths across cells, mapping robot real coordinate to grid cell and robot angle to only 08 compass directions
- Abstract a robot's shape to a rectangle to consider dimension and direction in collision detection based on intersection test
- Discretize movement step from micro to per cell, convert speeds to time taken for traveling or rotating and model velocities by scheduling the next pose of the robot after travel time: $updateMovingStatus()after(time\_travel);$
- Model laser scans and obstacle detection in a discrete manner with proven precision and performance, which can be a reusable modelling pattern.

The 'enough' thresholds are identified and tested as follows:

- On an occupancy grid, it is enough to model the linear movement of the robot from cell to cell and the rotating movement by capturing only 08 directions ($dir * 45°$) , given that:
- Grid resolution must be small enough so that half of the robot length always covers the middle point: $ROBOT\_LENGTH > CELL\_WIDTH * \sqrt{2}$
- Laser scan rate must be at least twice faster than the robot speed to ensure timely detection of obstacles. This agrees with the Nyquist-Shannon sampling theorem [5]. In terms of time, the time a robot takes an action should be at least twice longer than the period of a signal it depends on: $action\_time > 2{*}signal\_period$
- To model laser scans, 2°is enough for beam step for detecting adjacent obstacles without skipping or overlapping scans.

Other techniques for minimizing the state space and optimizing computations:

- Use a middle script to help with writing and debuging the model's code, and precompute data (like occupancy grid, trigonometric values of angles) to avoid repeated run-time computations
- Set message priorities to skip non-changing events, which only lengthen the model-checking time

Overall, we demonstrate a round-trip model-based engineering process, devising efficient modelling patterns that address the discrete-continuous gap and enable model-checking, and finally delivering a baseline Timed Rebeca model and ROS2 codebase for a multiple mobile robots system. We showcase the values of using models in prototyping, anticipating problems and testing algorithms and parameters, assuring functionality, safety and performance qualities in robotic software development.

ACKNOWLEDGMENT

The publication of this work was funded by the ORPHEUS project. Parts of its implementation were funded by ORPHEUS and KKS.

REFERENCES

[1] P. Bjesse, "What is formal verification?" *SIGDA Newsl.*, vol. 35, no. 24, p. 1–es, dec 2005.

[2] A. Brown, "Model driven architecture: Principles and practice," *Software and System Modeling*, vol. 3, pp. 314–327, 08 2004.

[3] E. M. Clarke, O. Grumberg, D. Kroening, D. A. Peled, and H. Veith, *Model checking, 2nd Edition*. MIT Press, 2018. [Online]. Available: https://mitpress.mit.edu/books/model-checking-second-edition

[4] A. H. Reynisson, M. Sirjani, L. Aceto, M. Cimini, A. Jafari, A. Ingólfsdóttir, and S. H. Sigurdarson, "Modelling and simulation of asynchronous real-time systems using Timed Rebeca," *Sci. Comput. Program.*, vol. 89, pp. 41–68, 2014. [Online]. Available: https://doi.org/10.1016/j.scico.2014.01.008

[5] C. E. Shannon, "Communication in the presence of noise," 1948.

[6] M. Sirjani, "Power is overrated, go for friendliness! expressiveness, faithfulness, and usability in modeling: The actor experience," in *Principles of Modeling - Essays Dedicated to Edward A. Lee on the Occasion of His 60th Birthday*, ser. Lecture Notes in Computer Science, M. Lohstroh, P. Derler, and M. Sirjani, Eds., vol. 10760. Springer, 2018, pp. 423–448. [Online]. Available: https://doi.org/10.1007/978-3-319-95246-8_25

[7] M. Sirjani, E. A. Lee, and E. Khamespanah, "Model checking software in cyberphysical systems," in *44th IEEE Annual Computers, Software, and Applications Conference, COMPSAC 2020, Madrid, Spain, July 13-17, 2020*. IEEE, 2020, pp. 1017–1026. [Online]. Available: https://doi.org/10.1109/COMPSAC48688.2020.0-138

[8] H. Trinh, "Model based development and verification of ROS2 robotic applications using Timed Rebeca, Master Thesis, Malardalen University," 2023.