The Landscape of Formal Methods for Robotics

Marie Farrell, **Matt Luckcuck**, Louise A. Dennis, Clare Dixon and Michael Fisher

Department of Computer Science, University of Liverpool, UK

20th of March 2019

Structure

1. Methodology
2. What is a Robotic System?
3. General Software Engineering Techniques for Robotic Systems
4. Robotic Systems' Challenges
5. Formalisms, Tools and Approaches for Robotic Systems
6. Conclusions

Introduction

Aim

We describe the current state of formal methods being applied to robotics.

This tutorial is based on the survey paper *Formal Specification and Verification of Autonomous Robotic Systems: A Survey*, which looks at the last ten years of literature.
The current version is available on ArXiv: 1807.00048.

Methodology

Methodology: Scope

Survey Scope

- systems that (eventually) have some physical effect on the world
- systems that both affect and are controlled by humans
- full range of autonomy
- formal properties concerning the behaviour of autonomous robotic systems
- formal techniques, not (for example) differential equations

Methodology: Research Questions

RQ1: What are the challenges when formally specifying and verifying the behaviour of (autonomous) robotic systems?

RQ2: What are the current formalisms, tools, and approaches used when addressing the answer to RQ1?

RQ3: What are the current limitations of the answers to RQ2 and are there developing solutions aiming to address them?

Methodology: Search Criteria

- Search Queries: formal modelling, formal specification and formal verification of (autonomous) robotic systems
- 5 pages deep on Google Scholar results (21/05/2018)
- surveyed 156 papers with 63 deemed to be in scope
- restricted to last ten years (2007–2018)

What is a Robotic System?

What is a Robotic System?

Multi-dimensional:

- Embedded System
- Cyber-Physical System
- Real-Time System
- Hybrid System
- Adaptive System
- Autonomous System

What is a Robotic System?

Multi-dimensional:

- Embedded System
- Cyber-Physical System
- Real-Time System
- Hybrid System
- Adaptive System
- Autonomous System

A machine that implements Artificial Intelligence and interacts with the physical world.

General Software Engineering Techniques for Robotic Systems

General Software Engineering Techniques for Robotic Systems

Robot Software Engineering

Our survey covered *formal* methods, but there were also some non-formal software engineering techniques specifically addressing robotic systems.

General Software Engineering Techniques for Robotic Systems

- **Testing and Simulation:** field-tests using the real robots and/or simulations
- **Middleware Architectures:** ROS, OPRoS, OpenRTM, Orocos and G^{en}oM
- **Domain Specific Languages:** describing robotic systems, often aimed at particular subdomains (e.g. robot motion)
- **Graphical Notations:** Statecharts (ArmarX, restricted Finite State Machines), RoboFlow, etc.
- **MDE/XML:** AutomationML, BRICS Component Model, etc.

Robotic Systems' Challenges

What are the Challenges?

We partitioned the challenges currently being tackled into two sets:

External Challenges:

- Modelling the Physical Environment
- Trust and Certification Evidence

Internal Challenges:

- Agent-Based Systems
- Multi-Robot Systems
- Self-Adaptive and Reconfigurable Systems

Modelling the Physical Environment

Challenge:

- How to specify and verify the behaviour of the robot working in a dynamic and often unknown environment that is further complicated by differing and/or degraded sensor accuracy.




Modelling the Physical Environment

Current Solutions:

- Ignore the environment![a]
- Assume that the environment it is static and known, prior to deployment[b]
- Use predicates representing sensor data to abstract away from the environment[c]

[a] Savas Konur, Clare Dixon, and Michael Fisher. "Analysing Robot Swarm Behaviour via Probabilistic Model Checking". In: *Robotics and Autonomous Systems* 60.2 (2012), pp. 199–213.

[b] Salar Moarref and Hadas Kress-Gazit. "Decentralized control of robotic swarms from high-level temporal logic specifications". In: *Int. Symp. Multi-Robot Multi-Agent Syst.* IEEE, 2017.

[c] Michael Fisher, Louise A Dennis, and Matt Webster. "Verifying Autonomous Systems". In: *Commun. ACM* 56.9 (2013), pp. 84–93.

Modelling the Physical Environment

Formal Methods must bridge the *reality gap*:

1. Model the environment using e.g. Probabilistic Temporal Logic (PTL)[a]

2. Monitor the environment using e.g. Timed Automata[b]

[a] M. Webster et al. "Toward Reliable Autonomous Robotic Assistants Through Formal Verification: A Case Study". In: *IEEE Transactions on Human-Machine Systems* 46.2 (2016), pp. 186–196.

[b] Adina Aniculaesei et al. "Towards the Verification of Safety-critical Autonomous Systems in Dynamic Environments". In: *Electron. Proc. Theor. Comput. Sci.* 232 (2016), pp. 79–90.

Trust and Certification Evidence

Robotic systems operate in areas that are:

1. Saftey-Critical e.g. nuclear/aerospace

2. Require public trust

Trust and Certification Evidence

Challenges:

- Formal verification providing appropriate trust and certification evidence
- Determining suitable formal methods for particular types of robotic system.

Trust and Certification Evidence

Current Solutions:

- Automatic generation of safety cases e.g. the AUTOCERT tool for a pilotless aircraft[a]

- Formalising and verifying domain specific rules e.g. using Isabelle/HOL to formalise rules for vehicle overtaking[b]

[a] Ewen Denney and Ganesh Pai. "Automating the assembly of aviation safety cases". In: *IEEE Transactions on Reliability* 63.4 (2014), pp. 830–849.

[b] Albert Rizaldi et al. "Formalising and monitoring traffic rules for autonomous vehicles in Isabelle/HOL". In: *Integr. Form. Methods*. Vol. 10510. LNCS. 2017, pp. 50–66.

Agent-Based Systems

- A model of autonomy.
- An agent encapsulates the system's decision-making capability into one component.
- It helps to provide *rational* autonomy (can explain its reasoning) which is crucial for certification and trust purposes

Challenge:

Ensuring that agents are verifiable.

Agent-Based Systems

Current Approaches:

- Belief-Desire-Intention (BDI) model of agency[a].
- Model Checker for Multi-Agent Systems (MCMAS)[b].
- Alloy for verifying multi-agent systems[c].

---

[a] Mark D'Inverno et al. "The dMARS Architecture: A Specification of the Distributed Multi-Agent Reasoning System". In: *Auton. Agent. Multi. Agent. Syst.* 9.1/2 (2004), pp. 5–53.

[b] Jiyoung Choi, Seungkeun Kim, and Antonios Tsourdos. "Verification of heterogeneous multi-agent system using MCMAS". In: *Int. J. Syst. Sci.* 46.4 (2015), pp. 634–651.

[c] Rodion Podorozhny et al. "Verification of Multi-agent Negotiations Using the Alloy Analyzer". In: *Integr. Form. Methods*. Vol. 4591. LNCS. 2007, pp. 501–517.

Multi-Robot Systems

There are many different kinds of Multi-Robot System including:

- Swarms of homogeneous robots
- Teams of heterogeneous robots

Multi-Robot Systems: Swarms

Challenges:

- Linking the formal specification and verification used at the microscopic (individual robots) level and macroscopic (whole system) level.
- How to resolve the state space explosion problem when model-checking large swarms.

Multi-Robot Systems: Swarms

Current Approaches:

- Temporal logics and model-checking to specify and verify swarms at different levels of abstraction[a]
- Using techniques such as symmetry reduction or abstracting the swarm to a single robot helps to mitigate the state space explosion problem[b]

[a] Alan F.T. Winfield et al. "On formal specification of emergent behaviours in swarm robotic systems". In: *Int. J. Adv. Robot. Syst.* 2.4 (2005), pp. 363–370.

[b] Savas Konur, Clare Dixon, and Michael Fisher. "Analysing Robot Swarm Behaviour via Probabilistic Model Checking". In: *Robotics and Autonomous Systems* 60.2 (2012), pp. 199–213.

Multi-Robot Systems: Heterogeneous Teams

Challenge:

How to link the formal methods used for the specification and verification of individual robots and the overall behaviour of the team

Multi-Robot Systems: Heterogeneous Teams

Current Approaches:

- A methodology for automating the development of robot teams using LTL-X and model-checking[a].
- FOL formalisation of beliefs and intentions to allow a robot to predict the plan of another agent[b].

---

[a] Marius Kloetzer, Xu Chu Ding, and Calin Belta. "Multi-robot deployment from LTL specifications with reduced communication". In: *Decis. Control Eur. Control Conf.* IEEE. 2011, pp. 4867–4872.

[b] Kartik Talamadupula et al. "Coordination in human-robot teams using mental modeling and plan

Self-Adaptive and Reconfigurable Systems

- Self-adaptive systems are driven by and respond to changes in the environment
- Reconfigurable systems sense their environment and *decide* on how best to reconfigure themselves
- Reconfigurability requires the system to *autonomously* make a decision and this autonomous behaviour must be verified

Challenges:

- Ensuring 'correct' choice of configuration
- Ensuring each configuration is 'correct'

Adaptation, Reconfigurability and Autonomy

Current Approaches:

- Model-checking at runtime for self-adaptive systems[a]
- Agent-based systems to model autonomy that are verified using temporal logics and model-checkers e.g. probabilistic model-checking of autonomous mine detector robot[b]

[a] Betty H.C. Cheng et al. "Using models at runtime to address assurance for self-adaptive systems". In: *Models@run.time.* Vol. 8378. LNCS. 2014, pp. 101–136.

[b] Paolo Izzo, Hongyang Qu, and Sandor M. Veres. "A stochastically verifiable autonomous control architecture with reasoning". In: *Conf. Decis. Control* (2016), pp. 4985–4991.

Formalisms, Tools and Approaches for Robotic Systems

Formalisms for Robotic Systems

Summary

- Temporal logics most prevalent formalism
  - Specifying properties
- Discrete Event Systems (state-transition systems) second most used
  - Often to specify systems

Formalisms for Robotic Systems

Formalism	Total
Temporal Logics	34
Discrete Event Systems	22
Discrete Event Systems (minus Temporal Logics)	11
Model-Oriented Specification	5
Process Algebra	3
Ontologies	4
Other Formalisms	12

Formalisms for Robotic Systems: Temporal Logic

- Used for specifying dynamic properties about a system over linear or branching time
- Extensions include: Linear-Time Temporal Logic (LTL), Computation Tree Logic (CTL), Probabilistic Temporal Logic (PTL), Probabilistic Computation Tree Logic (PCTL), LTL-X (LTL minus the ‘next’ operator), etc.

Temporal Logic Examples

- Automatically building PTL models of the safety rules and environment of a domestic robot assistant[a].
- Using LTL specifications to synthesise robot motion automata[b].

[a] Paul Gainer et al. "CRutoN: Automatic Verification of a Robotic Assistant's Behaviours". In: *Int. Work. Form. Methods Ind. Crit. Syst.* Vol. 10471. LNCS. 2017, pp. 119–133.

[b] Sertac Karaman and Emilio Frazzoli. "Sampling-based motion planning with deterministic μ-calculus specifications". In: *Conf. Decis. Control.* Ed. by John Baillieul and Lei Guo. IEEE. IEEE, 2009, p. 8.

Formalisms for Robotic Systems: Discrete Event Systems

- Used to specify behaviour during the design phase or used as input to a tool which usually checks them for properties specified in another formal language (e.g. temporal logic).

Formalisms for Robotic Systems: Discrete Event Systems

Discrete Event Systems Examples

- An extension of Petri Nets to capture robot plans which can be executed to find a sequence of transitions from the start to goal markers[a].
- Capture communication between ROS nodes using Timed Automata[b].

[a] V A Ziparo et al. "Petri Net Plans: A Formal Model for Representation and Execution of Multi-robot Plans". In: *Auton. Agents Multiagent Syst.* Vol. 23. AAMAS. 2008, pp. 79–86.

[b] Raju Halder et al. "Formal verification of ROS-based robotic applications using timed-automata". In: *Proceedings - 2017 IEEE/ACM 5th International FME Workshop on Formal Methods in Software Engineering, FormaliSE 2017* (2017), pp. 44–50.

Formalisms for Robotic Systems: Model-Oriented Formalisms

- Specify a system as a collection of data and a set of operations that manipulate that data.
- Well suited to capturing complicated data structures but only provide limited features for capturing behaviour.

Examples of its use:

- Z model that describes an arbitrary self-adaptive system[a].
- Event-B specifications are integrated with probabilistic properties to derive reconfigurable architectures for an on-board satellite system[b].

[a] Danny Weyns and Sam Malek. "FORMS: a formal reference model for self-adaptation". In: *Int. Conf. Auton. Comput.* ACM, 2010, pp. 205–214.

[b] Anton Tarasyuk et al. "Formal development and assessment of a reconfigurable on-board satellite system". In: *Int. Conf. Computer Safety, Reliability, and Security*. Vol. 7612. LNCS. 2012, pp. 210–222.

Formalisms for Robotic Systems: Process Algebras

- Define the behaviours of a system in terms of events and the interactions of processes.
- Suited for specifying concurrent systems.

Formalisms for Robotic Systems: Process Algebras

Process Algebra Examples

- Combination of Finite State Processes Process Algebra and π-calculus to specify multi-agent systems[a].
- RoboChart provides a formal semantics, based on CSP, for a timed state machine notation[b].

[a] Nadeem Akhtar. "Contribution to the Formal Specification and Verification of a Multi-Agent Robotic System". In: *Eur. J. Sci. Res.* 117.1 (2014), p. 2014.

[b] Pedro Ribeiro et al. "Modelling and verification of timed robotic controllers". In: *LNCS* 10510 (2017), pp. 18–33.

Formalisms for Robotic Systems: Ontologies

- Used to specify the key concepts, properties, relationships and axioms of a given domain so that it is possible to reason over the information that it represents and infer new information.

Formalisms for Robotic Systems: Ontologies

Examples of its use:

- Describe the robot environment, describe and reason about actions and for the reuse of domain knowledge[a].
- KnowRob is a knowledge processing system for autonomous personal robot assistants[b].

[a] Craig Schlenoff et al. "An IEEE standard ontology for robotics and automation". In: *Intell. Robot. Syst.* IEEE. 2012, pp. 1337–1342.

[b] Moritz Tenorth and Michael Beetz. "KnowRob—knowledge processing for autonomous personal robots". In: *Intell. Robot. Syst.* IEEE. 2009, pp. 4261–4266.

Formalisms for Robotic Systems: Others

Examples of other formalisms

- KLAIM is a formal language to capture properties about distributed systems, it has a stochastic extension, STOKLAIM[a].
- Dynamic logic for the specification and verification of hybrid programs to describe the discrete and continuous navigation behaviour of a ground robot[b].
- Propositional dynamic logic as a verification logic for agent-based systems[c].

---

[a] Edmond Gjondrekaj et al. "Towards a formal verification methodology for collective robotic systems". In: *Form. Eng. Methods*. Vol. 7635 LNCS. Springer, 2012, pp. 54–70.

[b] Stefan Mitsch, Khalil Ghorbal, and André Platzer. "On Provably Safe Obstacle Avoidance for Autonomous Robotic Ground Vehicles". In: *Robot. Sci. Syst.* (2013).

[c] K V Hindriks and J.-J. Ch. Meyer. "Toward a programming theory for rational agents". In: *Auton. Agent. Multi. Agent. Syst.* 19.1 (2009), pp. 4–29.

Tools for Robotic Systems

Summary

- Model checkers were the most used tool
  - Temporal Logics and Discrete Event Systems
- Other toolsets for specific logics or approaches were the second most common

Tools for Robotic Systems

Type of Tool	Tool	Total	Type Total
Model-Checkers	Prism	4	25
	NuSMV	2	
	Uppaal	3	
	SAL	1	
	SPIN	5	
	Beryl	2	
	Aldebaran	1	
	Dfinder	4	
	Unspecified	3	
Program Model Checkers	AJPF	4	7
	MCMAS	3	
Theorem Provers	KeyMaera	2	3
	SteP	1	
Others	Bio-PEPA Tool Suite	1	14
	TmeNET	1	
	TuLiP	1	
	LTLMoP	2	
	Alloy	2	
	Evaluator	1	
	minisat	1	
	MissionLab (VIPARS)	1	
	RV-BIP	1	
	Community Z Tools	3	

Approaches to Formally Verifying Robotic Systems

Summary

- "Approach" meaning the tool(s) or technique(s) used to verify the system
- Most used was model-checking
  - Including program model-checking
- Formal software development frameworks were the next most popular

Approaches to Formally Verifying Robotic Systems

Approach	Total
Model-Checking	32
Formal Software Frameworks /Architectures	10
Integrated Formal Methods	8
Theorem Proving	3
Runtime Monitoring	3

Approaches: Model-Checking

- Can be used with temporal logics, process algebras and programs.
- Model-checkers are automatic, making them easy to use and the approach is relatively easy to explain to stakeholders.
- Some can handle timing and others, probabilities.
- RQ3: Suffers from state space explosion problem.

Approaches: Model-Checking

Model-Checking Examples

- Büchi Automata have been used to represent the robot's environment and model-checked for an accepting path satisfying an LTL specification[a].
- Model-checking used to find traces of a transition system describing the behaviour of a robot team that satisfy an LTL-X formula[b].

[a] Meng Guo, Karl Johansson, and Dimos Dimarogonas. "Revising Motion Planning under Linear Temporal Logic Specifications in Partially Known Workspaces". In: *Robot. Autom.* IEEE. IEEE, 2013, pp. 5025–5032.

[b] Marius Kloetzer, Xu Chu Ding, and Calin Belta. "Multi-robot deployment from LTL specifications with reduced communication". In: *Decis. Control Eur. Control Conf.* IEEE. 2011, pp. 4867–4872.

Approaches: Frameworks for Verifiable Robotic Software

- Toolsets and design guides for developing verifiable robotic systems.
- RQ3: no real consensus between the approaches.

Approaches: Frameworks for Verifiable Robotic Software

Frameworks Examples

- Behaviour Interaction Priority (BIP) is a toolset for modelling component-based real-time software, with a notation based on finite state machines[a].
- Averest provides tools for verifying temporal properties of synchronous programs that are written in the Quartz language[b].

[a] Ananda Basu et al. "Rigorous System Design Using the BIP Framework". In: *Software* 28.3 (2011), pp. 41–48.
[b] Martin Proetzsch and Karsten Berns. "Formal verification of safety behaviours of the outdoor robot ravon". In: *Informatics Control. Autom. Robot.* 2007, pp. 157–164.

Approaches: Integrated Formal Methods

- The integration of multiple formal methods, or a formal method with a semi- or non-formal approach, that complement each other.
- This becomes a necessary approach as systems become more complex and critical.
- RQ3: Currently no generic framework for integrating formal methods for robotics.

Approaches: Integrated Formal Methods

Examples of its use:

- Combination of spatial reasoning, AJPF and Uppaal to verify an agent controlling a car[a].
- Combination of CSP and B (CSP||B) to verify cooperation between vehicles and the abstract behaviour of the physical vehicle[b].

[a] Maryam Kamali, Sven Linker, and Michael Fisher. "Modular verification of vehicle platooning with respect to decisions, space and time". In: (2018), pp. 18–36.

[b] Samuel Colin et al. "Using CSP || B components: application to a platoon of vehicles". In: *International Workshop on Formal Methods for Industrial Critical Systems*. Springer. 2008, pp. 103–118.

Approaches: Theorem Proving

- Produces a formal proof of the correctness of the software system.
- Proofs can be used to provide robust trust and certification evidence.
- RQ3: Not as usable as other approaches and tools are generally difficult to use.

Approaches: Theorem Proving

Examples of its use:

- Isabelle/HOL to formalise a subset of the German traffic rules for vehicle overtaking[a].
- KeYmaera hybrid theorem prover to verify that a robot would not collide with stationary or moving obstacles and maintain a suitable distance from obstacles[b].

[a] Albert Rizaldi et al. "Formalising and monitoring traffic rules for autonomous vehicles in Isabelle/HOL". In: *Integr. Form. Methods*. Vol. 10510. LNCS. 2017, pp. 50–66.

[b] Stefan Mitsch, Khalil Ghorbal, and André Platzer. "On Provably Safe Obstacle Avoidance for Autonomous Robotic Ground Vehicles". In: *Robot. Sci. Syst.* (2013).

Approaches: Runtime Monitoring

- **Monitor**: consumes events from the system and compares them to the expected behaviour. If they differ, then it can invoke mitigating activities e.g. warn the user.
- Can be easier to verify.
- Can help mitigate the problem of the 'reality gap' when used to complement offline verification.

Approaches: Runtime Monitoring

Examples of its use:

- Used to recognise anomalous environmental interactions and so highlight when the previous formal verification done on an autonomous robotic system becomes invalid[a].
- ROSRV is a runtime verification framework for robotics systems deployed on ROS[b].

[a] Angelo Ferrando et al. "Recognising assumption violations in autonomous systems verification". In: *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems. 2018, pp. 1933–1935.

[b] Jeff Huang et al. "ROSRV: Runtime Verification for Robots". In: *Runtime Verif.* 2014, pp. 247–254.

Conclusions

RQ1: Challenges

RQ1: What are the challenges when formally specifying and verifying the behaviour of (autonomous) robotic systems?

External Challenges:

- Modelling the Physical Environment
- Trust and Certification Evidence

Internal Challenges:

- Agent-Based Systems
- Multi-Robot Systems
- Self-Adaptive and Reconfigurable Systems

RQ2: Current Formalims, Tools, and Approaches

RQ2: What are the current formalisms, tools, and approaches used when addressing the answer to RQ1?

Answer RQ2

- Temporal logics, discrete event systems and model-checkers are the most prominent formalisms and approaches in the literature.
- Why?
  - Temporal logics and discrete event systems allow abstract specification, which is useful early in the development process.
  - Model-checking is easy to explain to stakeholders who do not have experience with formal methods.

RQ3: Limitations

RQ3: What are the current limitations of the answers to RQ2 and are there developing solutions aiming to address them?

Answer RQ3

- Formal Methods aren't well integrated into robotic systems engineering
  - Some tool-chains tackling the whole process
- Tool support *for mere mortals*…
  - Getting better, but needs testing/trials with *real users*
- Lots of notations and tools but barely any interoperability
- Formalising the *last link* between the formal model and the program code

Questions?

ArXiv Preprint:

Luckcuck M., Farrell M., Dennis L., Dixon C., & Fisher M. (2018). *Formal Specification and Verification of Autonomous Robotic Systems: A Survey*. ArXiv: 1807.00048.

ROBOTICS AND AI IN NUCLEAR