MAPmAKER: A Tool for Performing Multi-Robot LTL Planning Under Uncertainty

Anonymous Author(s)

ABSTRACT

Robot applications are increasingly asking for decentralized techniques that allow for tractable automated planning. Another aspect that state-of-the-art robot applications must consider is *partial knowledge* about the environment in which the robots are operating and the associated uncertainty with the outcome of the robots' actions.

Current planning techniques used for teams of robots that should perform complex missions do not systematically address these challenges: they are either based on centralized solutions and hence not scalable, they consider rather simple missions, such as A-to-B travel, or do not work in partially known environments. We present a planning solution that decomposes the team of robots into subclasses, considers complex high-level missions given in temporal logic, and at the same time works when only *partial knowledge* of the environment is available. We prove the correctness of the solution and evaluate its effectiveness on a set of realistic examples.

ACM Reference format:
Anonymous Author(s). 2018. MAPmAKER: A Tool for Performing Multi-Robot LTL Planning Under Uncertainty. In *Proceedings of 40th International Conference on Software Engineering, Gothenburg, Sweden, May 27–June 3, 2018 (ICSE 2018),* 5 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

A *planner* is a software component that receives as input a model of the robotic application and computes a set of actions (a *plan*) that, if performed, allows the achievement of a desired mission [25]. As done in some recent works in the robotics community (see for example [4, 5, 14, 15, 20, 22, 46]), in this work we assume that a robot application is defined using finite transition systems and each robot of the team has to achieve a mission, indicated as *local mission*, that is specified as an LTL property. As opposed to more traditional specification means, such as consensus or trajectory tracking in robot control, A-to-B travel in robot motion planning, or STRIPS or PDDL problem formulations in robot task planning, LTL allows us to specify a rich class of temporal goals that include e.g., surveillance, sequencing, safety, or reachability.

Several works studied centralized planners that are able to manage *teams* of robots that collaborate to achieve a certain goal (a global mission) [21, 28, 36]. Others studied how to decompose a global mission into a set of local missions to be achieved by each robot of the team [17, 17, 39, 42]. These local missions have been recently exploited by *decentralized* planners [42], i.e., planners that instead of evaluating the global mission over the whole team of robots, analyze the satisfaction of local missions inside a subset of the team of robots. In this way, the problem of finding a collective team behavior is decomposed into sub-problems that avoid the expensive fully centralized planning.

Another aspect that current planners must consider is partial knowledge about the environment in which the robots should operate. Partial knowledge in software development has been strongly studied by the software engineering community. For example, partial models have been used to support requirement analysis and elicitation [26, 31, 32], to help designers in producing a model of the system that satisfies a set of desired properties [2, 16, 43, 44] and to verify whether already designed models possess some properties of interest [7, 8, 30]. However, most of the existing planners assume that the environment in which the robots are deployed is known [10]. This assumption does not usually hold in real word scenarios [24]. In real world applications it is usually the case that only *partial knowledge* about the environment in which the robots are operating is present. Several works studied planners that work when only partial information about the environment in which the robots operate is available (e.g., [11, 13, 37]). However, literature considering *decentralized* planners with only partial knowledge about the robot application and temporal logic goals is rather limited [17].

Organization. Section 2 introduces robotic applications by highlighting the status of current planners. Section 3 describes the MAPmAKER approach. Section 4 presents the MAPmAKER tool. Section 5 concludes with final remarks.

2 LIMITATIONS OF CURRENT PLANNERS

Decentralized solutions. –Kind of state of the art, relate the tool with others and make emphasis on the differences –

Decentralized planning problem has been studied for known environments [17, 39, 42]. However, planners for partially known environments do not usually employ decentralized solutions [11, 13, 37].

Dealing with partial knowledge in planning. Planning in partially known environments is handled in different ways. (1) Several works (e.g., [3, 6, 9, 12, 13, 23, 33, 34, 38, 45]) consider probabilities within the planning algorithm. In [24], to plan trajectories the authors use a high-level planner that exploits an abstraction of the hybrid system and the mission to compute high-level plans. The low-level planner uses the dynamics of the hybrid system and the suggested high-level plans to explore the state space for feasible solutions. Every time an unknown obstacle is encountered, the high-level planner modifies the coarse high-level plan online by accounting for the geometry of the discovered obstacle. Within this framework,

ICSE 2018, Gothenburg, Sweden

DOI: 10.1145/nnnnnnn.nnnnnnn

MAPmAKER can be considered as a high-level planner that is able to use an abstraction of the hybrid system that contains partial information, i.e., encode unknown obstacles. (2) Some approaches analyzed how to update plans when new information about known model of a robotic application is detected (e.g., [17]). Differently, in our approach portions of the model of the robotic application are partially known, partial knowledge is reduced as true and false evidence about partial information is detected. Other works (e.g., [1]), aim at detecting how to explore totally unknown environments. (3) Plan synthesis is a particular instance of controller synthesis. (4) MAPmAKER can be classified on the boundary between reactive synthesis [9, 27, 41] techniques and iterative planning [18, 29].

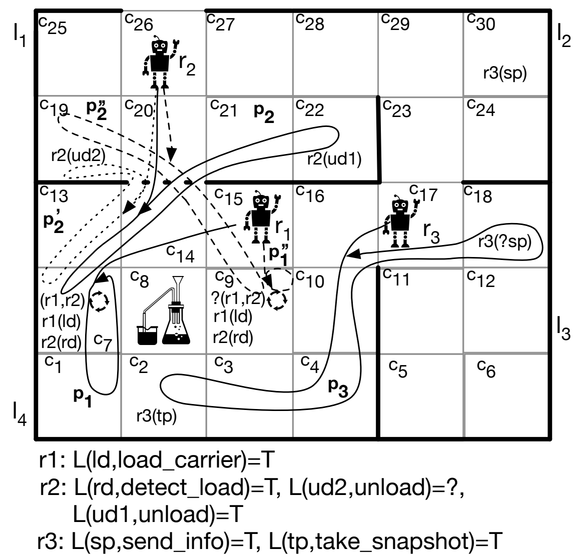

Figure 1: An example showing the model of the robots and their environment. Plans computed by MAPmAKER are represented by trajectories marked with arrows.

3 THE MAPMAKER APPROACH

–Explain the tool from a high-level point of view–

!!

Specific contributions. Specific contributions are detailed in the following: (1) we define the concept of *partial robot model*, which allows the description of the behavior of the robots and its environment when only partial information is available. Specifically, a partial robot model allows considering three types of partial information: partial knowledge about the execution of transitions (possibility of changing the robot location), on service provision (whether the execution of an action succeed in providing a service) and on the meeting capabilities (whether a robot can meet with another); (2) we define the concept of local mission satisfaction for partial robot models; (3) we define the distributed planning problem for partially specified robots; (4) we propose a distributed planning algorithm and we proved its correctness; (5) we evaluate the proposed algorithm on a robot application obtained from the RobotCup Logistics League competition [19] and on a robotic application working in an apartment of about 80 m^2, which is part of a large residential facility for senior citizens [40]. The results show the effectiveness of the proposed algorithm.

!!

A set $R = \{r_1, r_2, r_3\}$ of robots is deployed in the environment graphically described in Fig. 2. This environment represents a building made by four rooms $L = \{l_1, l_2, l_3, l_4\}$, which has been affected by an earthquake. The environment is further partitioned in cells, each labeled with an identifier in $c_1, c_2, \ldots, c_{30}$. Robots r_1, r_2, and r_3 are placed in their initial locations. Each robot is able to move from one cell to another, by performing action *mov*. The robots are also able to perform the following actions. Robot r_1 is able to load debris of the building by performing action *ld*. In Fig. 2 the cells in which a robot r can perform an action α are marked with the label $r(\alpha)$. Robot r_2 can wait until another robot loads debris on it by performing action *rd* and can unload debris by performing one of the two actions *ud*1 and *ud*2. Actions *ud*1 and *ud*2 use different actuators. Specifically, action *ud*1 uses a gripper while action *ud*2 exploits a dump mechanism. Robot r_3 is able to take pictures by performing action *tp* and send them using a communication network through the execution of action *sp*. Symbols $r_1(ld)$, $r_2(rd)$, $r_2(ud1)$, $r_2(ud2)$, $r_3(tp)$, and $r_3(sp)$ are used in Fig. 2 to mark the regions where actions can be executed by the robots, while movement actions are not reported for graphical reasons. Each action may be associated with a service, which is a high-level functionality provided by the robot when an action is performed. For example, actions *ld*, *rd*, *tp*, and *sp* are associated with the services *load_carrier*, *detect_load*, *take_snapshot*, and *send_info*, respectively. Actions *ud*1 and *ud*2 are associated with service *unload*. The labels $L(\pi, \alpha) = \top$ below Fig. 2 are used to indicate that a service π is associated with action α. Robots must meet and synchronously execute actions. In this example, robots r_1 and r_2 must meet in cell c_7 and synchronously execute actions *ld* and *rd*, respectively. The cells where meeting is requested are marked with rotating arrows marked with the identifiers of the robots that must meet, meaning that, in order to meet, the robots must be on the same cell to meet.

The *mission* the team of robots has to achieve is to check whether toxic chemicals have been released by the container located in l_4. We assume that the mission is specified through a set of *local missions* assigned to each robot of the team and described in Linear Time Temporal Logic (LTL). An LTL formula is obtained by composing actions with standard LTL operators: X (next), F (eventually), G (always) and U (until) [35]. In our example the mission can be specified by means of the following local missions: $\phi_1 = \mathsf{G}(\mathsf{F}(\textit{load\_carrier}))$, $\phi_2 = \mathsf{G}(\mathsf{F}(\textit{detect\_ load} \wedge \mathsf{F}(\textit{unload})))$, $\phi_3 = \mathsf{G}(\mathsf{F}(\textit{take\_snapshot} \wedge \mathsf{F}(\textit{send\_info})))$, which are assigned to robot r_1, r_2 and r_3, respectively. The formulae specify that periodically robot r_1 loads debris on r_2 (by performing action *load_carrier*), robot r_2 receives debris (when action *detect_ load* occurs) and brings them to an appropriate unload area (by performing action *unload*), and robot r_3 continuously takes pictures (by performing action *take_snapshot*) and sends them using the communication network (by performing action *send_info*). Informally, while r_3 continuously takes pictures and sends them using the communication network, r_1 and r_2 remove debris to allow r_3 having a better view on the container. The pictures allow verifying whether toxic chemicals have been released by the container.

The presence of partial knowledge about the robots and their environment is described in the following.

Partial knowledge about the actions execution. The robots can move between cells separated by grey lines, while they cannot cross black bold lines. It is unknown whether it is possible to move between cells c_{14} and c_{20} since the structure may have been affected by collapses. This is indicated using a dashed black bold line. It is also unknown whether robot r_3 can send pictures using a communication network in location l_3 and specifically in cell c_{18}, i.e., whether action s_p can be performed. Locations of the environment where it is unknown if an action can be provided are marked with the name of the action preceded by symbol ?.

Unknown service provisioning. There are cases in which actions can be executed but there is uncertainty about service provisions. For example, actions $ud1$ and $ud2$ of robot r_2 unload the robot. Action $ud2$ will always be able to provide the *unload* service, while it is unknown whether $ud1$ is actually able to provide this service since its effectiveness depends on the size of the collected debris. In Fig. 2, the label $L(ud1, unload)$ =? indicates that there is partial knowledge about the provision of the *unload* service when action $ud1$ is performed.

Unknown meeting capabilities. It is unknown whether robots r_1 and r_2 can meet in one cell of the environment. For example, a collapse in the roof of the building may forbid the two robots to concurrently execute services ld and rd, i.e., there is not enough space for $r1$ to load $r2$. Unknown meeting capabilities are indicated with rotating arrows labeled with the symbol ?. For example, in Fig. 2, it is unknown whether robots r_1 and r_2 are able to meet in cell c_9.

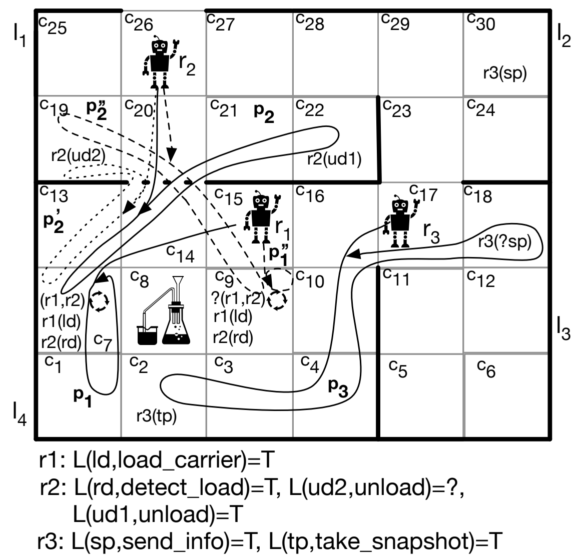

Figure 2: An example showing the model of the robots and their environment. Plans computed by MAPmAKER are represented by trajectories marked with arrows.

4 THE MAPMAKER TOOL

–Explain the tool in detail, maybe including a scope– (add figure for the tool? maybe in the previous section?)

!! **Contribution.** This work presents MAPmAKER (Multi-robot plAnner for PArtially Known EnviRonments), a *novel decentralized* planner for partially known environments. Given a team of robots and a local mission for each robot, MAPmAKER partitions the set of robots into classes based on dependencies dictated by the local missions of each robot. For each of these classes, it explores the state space of the environment and the models of the robot searching for definitive and possible plans. A *definitive plan* is a sequence of actions that ensure the satisfaction of the local mission for each robot. A *possible plan* is a sequence of actions that may satisfy the local mission due to some unknown information about the model of the robots or the environment in which they are deployed. MAPmAKER chooses the plan that allows the achievement of the mission by performing the lower number of actions, but other policies can also be used. !!

A set $R = \{r_1, r_2, r_3\}$ of robots is deployed in the environment graphically described in Fig. 2. This environment represents a building made by four rooms $L = \{l_1, l_2, l_3, l_4\}$, which has been affected by an earthquake. The environment is further partitioned in cells, each labeled with an identifier in $c_1, c_2, \ldots, c_{30}$. Robots r_1, r_2, and r_3 are placed in their initial locations. Each robot is able to move from one cell to another, by performing action mov. The robots are also able to perform the following actions. Robot r_1 is able to load debris of the building by performing action ld. In Fig. 2 the cells in which a robot r can perform an action α are marked with the label $r(\alpha)$. Robot r_2 can wait until another robot loads debris on it by performing action rd and can unload debris by performing one of the two actions $ud1$ and $ud2$. Actions $ud1$ and $ud2$ use different actuators. Specifically, action $ud1$ uses a gripper while action $ud2$ exploits a dump mechanism. Robot r_3 is able to take pictures by performing action tp and send them using a communication network through the execution of action sp. Symbols $r_1(ld)$, $r_2(rd)$, $r_2(ud1)$, $r_2(ud2)$, $r_3(tp)$, and $r_3(sp)$ are used in Fig. 2 to mark the regions where actions can be executed by the robots, while movement actions are not reported for graphical reasons. Each action may be associated with a service, which is a high-level functionality provided by the robot when an action is performed. For example, actions ld, rd, tp, and sp are associated with the services *load_carrier*, *detect_load*, *take_snapshot*, and *send_info*, respectively. Actions $ud1$ and $ud2$ are associated with service *unload.* The labels $L(\pi, \alpha) = \top$ below Fig. 2 are used to indicate that a service π is associated with action α. Robots must meet and synchronously execute actions. In this example, robots r_1 and r_2 must meet in cell c_7 and synchronously execute actions ld and rd, respectively. The cells where meeting is requested are marked with rotating arrows marked with the identifiers of the robots that must meet, meaning that, in order to meet, the robots must be on the same cell to meet.

The *mission* the team of robots has to achieve is to check whether toxic chemicals have been released by the container located in l_4. We assume that the mission is specified through a set of *local missions* assigned to each robot of the team and described in Linear Time Temporal Logic (LTL). An LTL formula is obtained by composing actions with standard LTL operators: X (next), F (eventually), G (always) and U (until) [35]. In our example the mission can be specified by means of the following local missions: $\phi_1 = \mathsf{G}(\mathsf{F}(load\_carrier))$, $\phi_2 = \mathsf{G}(\mathsf{F}(detect\_\ load \wedge \mathsf{F}(unload)))$, $\phi_3 = \mathsf{G}(\mathsf{F}(take\_snapshot \wedge \mathsf{F}(send\_info)))$, which are assigned to robot r_1, r_2 and r_3, respectively. The formulae specify that periodically robot r_1 loads debris on r_2 (by performing action *load_carrier*), robot r_2 receives debris (when action *detect_ load* occurs) and brings them to an appropriate unload area (by performing action *unload*), and robot r_3 continuously takes pictures (by performing action *take_snapshot*) and sends them using the communication network (by performing action *send_info*). Informally, while r_3 continuously takes pictures and sends them using the communication network, r_1 and r_2 remove debris to allow r_3 having a better view on the container. The pictures allow verifying whether toxic chemicals have been released by the container.

The presence of partial knowledge about the robots and their environment is described in the following.

Partial knowledge about the actions execution. The robots can move between cells separated by grey lines, while they cannot cross black bold lines. It is unknown whether it is possible to move between cells c_{14} and c_{20} since the structure may have been affected by collapses. This is indicated using a dashed black bold line. It is also unknown whether robot r_3 can send pictures using a communication network in location l_3 and specifically in cell c_{18}, i.e., whether action s_p can be performed. Locations of the environment where it is unknown if an action can be provided are marked with the name of the action preceded by symbol ?.

Unknown service provisioning. There are cases in which actions can be executed but there is uncertainty about service provisions. For example, actions $ud1$ and $ud2$ of robot r_2 unload the robot. Action $ud2$ will always be able to provide the *unload* service, while it is unknown whether $ud1$ is actually able to provide this service since its effectiveness depends on the size of the collected debris. In Fig. 2, the label $L(ud1, unload) =?$ indicates that there is partial knowledge about the provision of the *unload* service when action $ud1$ is performed.

Unknown meeting capabilities. It is unknown whether robots r_1 and r_2 can meet in one cell of the environment. For example, a collapse in the roof of the building may forbid the two robots to concurrently execute services ld and rd, i.e., there is not enough space for $r1$ to load $r2$. Unknown meeting capabilities are indicated with rotating arrows labeled with the symbol ?. For example, in Fig. 2, it is unknown whether robots r_1 and r_2 are able to meet in cell c_9.

5 CONCLUSIONS

–General conclusions (maybe use the same from the last paper but removing the discussion about the results)–

This work presented MAPmAKER, a novel decentralized planner for partially known environments. MAPmAKER solves the decentralized planning problem when partial robot applications are analyzed. We evaluated MAPmAKER by considering the robot application model of the RoboCup Logistics League competition [19] and an apartment of about 80 m^2, which is part of a large residential facility for senior citizens [40]. The results show that the effectiveness of MAPmAKER is triggered when the computed possible plans are actually executable in the real model of the robotic application. Furthermore, in several cases, MAPmAKER was able to achieve missions that could not be completed by classical planners.

Future work and research directions include (1) the study of appropriate policies to select between definitive and possible plans, (2) the use of more efficient planners to speed up plan computation. These may be based for example on symbolic techniques.

REFERENCES

[1] B. C. Akdeniz and H. I. Bozma. 2015. Exploration and topological map building in unknown environments. In *IEEE International Conference on Robotics and Automation (ICRA)*. 1079–1084.
[2] Aws Albarghouthi, Arie Gurfinkel, and Marsha Chechik. 2012. From under-approximations to over-approximations and back. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 157–172.
[3] C. Amato, G. Konidaris, G. Cruz, C. A. Maynor, J. P. How, and L. P. Kaelbling. 2015. Planning for decentralized control of multiple robots under uncertainty. In *IEEE International Conference on Robotics and Automation (ICRA)*. 1241–1248.
[4] Amit Bhatia, Lydia E Kavraki, and Moshe Y Vardi. 2010. Motion planning with hybrid dynamics and temporal goals. In *Conference on Decision and Control (CDC)*. IEEE, 1108–1115.
[5] Amit Bhatia, Lydia E Kavraki, and Moshe Y Vardi. 2010. Sampling-based motion planning with temporal goals. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 2689–2696.
[6] S. Bhattacharya, R. Ghrist, and V. Kumar. 2015. Persistent Homology for Path Planning in Uncertain Environments. *IEEE Transactions on Robotics* 31, 3 (2015), 578–590.
[7] Glenn Bruns and Patrice Godefroid. 1999. Model checking partial state spaces with 3-valued temporal logics. In *International Conference on Computer Aided Verification*. Springer, 274–287.
[8] Marsha Chechik, Benet Devereux, Steve Easterbrook, and Arie Gurfinkel. 2004. Multi-valued symbolic model-checking. *ACM Transactions on Software Engineering and Methodology* 12, 4 (2004), 1–38.
[9] Yushan Chen, Jana Tumová, and Calin Belta. 2012. LTL robot motion control based on automata learning of environmental dynamics. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 5177–5182.
[10] A. G. Cunningham, E. Galceran, R. M. Eustice, and E. Olson. 2015. MPDM: Multipolicy decision-making in dynamic, uncertain environments for autonomous driving. In *International Conference on Robotics and Automation (ICRA)*. 1670–1677.
[11] Jonathan F Diaz, Alexander Stoytchev, and Ronald C Arkin. 2001. Exploring Unknown Structured Environments.. In *FLAIRS Conference*. AAAI Press, 145–149.
[12] Xu Chu Dennis Ding, Stephen L Smith, Calin Belta, and Daniela Rus. 2011. LTL control in uncertain environments with probabilistic satisfaction guarantees. *IFAC Proceedings Volumes* 44, 1 (2011), 3515–3520.
[13] Noel E Du Toit and Joel W Burdick. 2012. Robot motion planning in dynamic, uncertain environments. *IEEE Transactions on Robotics* 28, 1 (2012), 101–115.
[14] Georgios E Fainekos, Antoine Girard, Hadas Kress-Gazit, and George J Pappas. 2009. Temporal logic motion planning for dynamic robots. *Automatica* 45, 2 (2009), 343–352.
[15] Georgios E Fainekos, Hadas Kress-Gazit, and George J Pappas. 2005. Temporal logic motion planning for mobile robots. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 2020–2025.
[16] Michais Famelis, Rick Salay, and Marsha Chechik. 2012. Partial models: Towards modeling and reasoning with uncertainty. In *International Conference on Software Engineering*. IEEE, 573–583.
[17] Meng Guo and Dimos V Dimarogonas. 2015. Multi-agent plan reconfiguration under local LTL specifications. *The International Journal of Robotics Research* 34, 2 (2015), 218–235.
[18] Meng Guo, Karl H Johansson, and Dimos V Dimarogonas. 2013. Revising motion planning under linear temporal logic specifications in partially known workspaces. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 5025–5032.
[19] Christian Deppe Ulrich Karras, Tobias Neumann, Tim Niemueller Alain Rohr, Wataru Uemura, Daniel Ewert, Nils Harder, Sören Jentzsch, Nicolas Meier, and Sebastianyear Reuter. 2016. RoboCup Logistics League Rules and Regulations. (2016).
[20] Marius Kloetzer and Calin Belta. 2008. A fully automated framework for control of linear systems from temporal logic specifications. *IEEE Trans. Automat. Control* 53, 1 (2008), 287–297.

[21] Marius Kloetzer, Xu Chu Ding, and Calin Belta. 2011. Multi-robot deployment from LTL specifications with reduced communication. In *Conference on Decision and Control and European Control Conference (CDC-ECC)*. IEEE, 4867–4872.

[22] Hadas Kress-Gazit, Georgios E Fainekos, and George J Pappas. 2007. Where's waldo? sensor-based temporal logic motion planning. In *International Conference on Robotics and Automation*. IEEE, 3116–3121.

[23] Hanna Kurniawati, Yanzhu Du, David Hsu, and Wee Sun Lee. 2011. Motion planning under uncertainty for robotic tasks with long time horizons. *The International Journal of Robotics Research* 30, 3 (2011), 308–323.

[24] Morteza Lahijanian, Matthew R Maly, Dror Fried, Lydia E Kavraki, Hadas Kress-Gazit, and Moshe Y Vardi. 2016. Iterative temporal planning in uncertain environments with partial satisfaction guarantees. *IEEE Transactions on Robotics* 32, 3 (2016), 583–599.

[25] Jean-Claude Latombe. 2012. *Robot motion planning*. Vol. 124. Springer.

[26] Emmanuel Letier, Jeff Kramer, Jeff Magee, and Sebastian Uchitel. 2008. Deriving event-based transition systems from goal-oriented requirements models. *Automated Software Engineering* 15, 2, 175–206.

[27] Scott C Livingston, Richard M Murray, and Joel W Burdick. 2012. Backtracking temporal logic synthesis for uncertain environments. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 5163–5170.

[28] Savvas G Loizou and Kostas J Kyriakopoulos. 2005. Automated planning of motion tasks for multi-robot systems. In *Conference on Decision and Control and European Control Conference (CDC-ECC)*. IEEE, 78–83.

[29] Matthew R Maly, Morteza Lahijanian, Lydia E Kavraki, Hadas Kress-Gazit, and Moshe Y Vardi. 2013. Iterative temporal motion planning for hybrid systems in partially unknown environments. In *International conference on Hybrid systems: computation and control*. ACM, 353–362.

[30] Claudio Menghi, Paola Spoletini, and Carlo Ghezzi. 2016. Dealing with Incompleteness in Automata-Based Model Checking. In *Formal Methods*, Vol. 9995. Springer, 531–550.

[31] Claudio Menghi, Paola Spoletini, and Carlo Ghezzi. 2017. COVER: Change-based Goal Verifier and Reasoner.. In *REFSQ Workshops*. Springer.

[32] Claudio Menghi, Paola Spoletini, and Carlo Ghezzi. 2017. Integrating Goal Model Analysis with Iterative Design. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 112–128.

[33] Venkatraman Narayanan and Maxim Likhachev. 2015. Task-oriented planning for manipulating articulated mechanisms under model uncertainty. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 3095–3101.

[34] Alexandros Nikou, Jana Tumova, and Dimos D Dimarogonas. 2017. Probabilistic Plan Synthesis for Coupled Multi-Agent Systems. *arXiv preprint arXiv:1704.01432* (2017).

[35] Amir Pnueli. 1977. The temporal logic of programs. In *Foundations of Computer Science*. IEEE, 46–57.

[36] Michael Melholt Quottrup, Thomas Bak, and RI Zamanabadi. 2004. Multi-robot planning: A timed automata approach. In *International Conference on Robotics and Automation*, Vol. 5. IEEE, 4417–4422.

[37] Nicholas Roy, Geoffrey Gordon, and Sebastian Thrun. 2006. Planning under uncertainty for reliable health care robotics. In *Field and Service Robotics*. Springer, 417–426.

[38] Nicholas Roy, Geoffrey Gordon, and Sebastian Thrun. 2006. *Planning under Uncertainty for Reliable Health Care Robotics*. Springer, 417–426.

[39] Philipp Schillinger, Mathias Bürger, and Dimos Dimarogonas. 2016. Decomposition of Finite LTL Specifications for Efficient Multi-Agent Planning. In *International Symposium on Distributed Autonomous Robotic Systems*.

[40] The Angen research and innovation apartment: official website 2014. (2014). http://angeninnovation.se

[41] Wolfgang Thomas et al. 2002. *Automata, logics, and infinite games: a guide to current research*. Vol. 2500. Springer Science & Business Media.

[42] Jana Tumova and Dimos V Dimarogonas. 2016. Multi-agent planning under local LTL specifications and event-based synchronization. *Automatica* 70 (2016), 239–248.

[43] Sebastian Uchitel, Dalal Alrajeh, Shoham Ben-David, Victor Braberman, Marsha Chechik, Guido De Caso, Nicolas D'Ippolito, Dario Fischbein, Diego Garbervetsky, Jeff Kramer, et al. 2013. Supporting incremental behaviour model elaboration. *Computer Science-Research and Development* 28, 4 (2013), 279–293.

[44] Sebastian Uchitel, Greg Brunet, and Marsha Chechik. 2009. Synthesis of partial behavior models from properties and scenarios. *IEEE Transactions on Software Engineering* 35, 3 (2009), 384–406.

[45] Eric M Wolff, Ufuk Topcu, and Richard M Murray. 2012. Robust control of uncertain Markov decision processes with temporal logic specifications. In *Annual Conference on Decision and Control (CDC)*. IEEE, 3372–3379.

[46] Tichakorn Wongpiromsarn, Ufuk Topcu, and Richard M Murray. 2010. Receding horizon control for temporal logic specifications. In *International conference on Hybrid systems: computation and control*. ACM, 101–110.