Integrating Diagnostic Reasoning in Plan Execution Monitoring for Cognitive Factories

Esra Erdem Volkan Patoglu Zeynep G. Saribatur
Faculty of Engineering and Natural Sciences, Sabancı University, İstanbul, Turkey
Email: {esraerdem,vpatoglu,zgsaribatur}@sabanciuniv.edu

***Abstract*—We propose a novel method for diagnostic reasoning using state-of-the-art automated reasoners (e.g., ASP solvers), and its integration in execution monitoring of plans of multiple teams of heterogeneous robots in a cognitive factory setting. The idea of the proposed execution monitoring algorithm is, when some changes or discrepancies are detected, to make appropriate decisions based on their causes. These causes (e.g., broken robots/components) are found by the proposed diagnostic reasoning method, by synergistically integrating knowledge representation, hypothetical reasoning, geometric reasoning, and learning from earlier experiences. Based on these causes, if necessary, new hybrid plans (task plans integrated with feasibility checks) are computed to reach the manufacturing goals by allowing repairs of robots/components.**

I. Motivation and Methods

We focus on reliable and fault tolerant operation of cognitive factories [6, 13], consisting of multiple teams of heterogeneous and reconfigurable robots, where teams collaborate for efficient use of shared resources towards achieving a common goal. In these dynamic environments, there may be changes in the environment (e.g., a change in the location of obstacles), or in the manufacturing orders (e.g., an increase in the orders). These changes can be detected by sensors or by direct communication between the teams and the monitoring agent. Also, there may be unexpected changes (e.g., broken robots, components) that are hard to detect directly from observations/communications, and that require deeper reasoning to find out. Identifying such changes is essential to prevent future plan failures or to recover from potential failures.

We introduce a general execution monitoring framework for cognitive factories, integrated with hybrid diagnostic reasoning, to ensure that the computed optimal plans are executed in a reliable and fault tolerant way, even under such changes. When the monitoring agent detects changes, it informs the relevant team about them so that appropriate decisions can be made to reach the manufacturing goals. In most cases, when these changes are relevant for successful execution of the rest of the plan, this decision involves replanning at the local level. When local replanning fails to resolve the problem, then global replanning may be triggered.

Local replanning is done in different ways depending on the sort of the changes. For instance, if some new obstacles are detected in the environment, replanning from the current state to a goal state has to consider complex, temporal state/transition constraints to avoid collisions of robots with the obstacles. If the manufacturing order is changed, then replanning has to consider the new goals. Otherwise, if a discrepancy between an observed state and the expected state is detected, then replanning has to take into account the causes of the discrepancy (i.e., broken robot components) as well as possible repairs. We propose to use diagnostic reasoning to compute causes of discrepancies to improve replanning and allow repairs.

Diagnostic reasoning is a crucial part of our execution monitoring framework, in particular, for replanning and repairs. For instance, suppose that a robot is broken; without a diagnosis, the team does not know about the condition of this robot and finding a new plan from the current state to a goal may not be possible at all, unless the robot is repaired. Therefore, we put more emphasis on the diagnostic reasoning aspect of our execution monitoring algorithm.

We introduce a novel generic method for diagnostic reasoning integrated with hypothetical reasoning, geometric reasoning, and learning. Our method starts with the robotic action domain description used to compute a task plan, the part of the task plan executed from the initial state until the current state, and a set of observations about the current state. The robotic domain is formalized in answer set programming (ASP) [3]—a nonmonotonic knowledge representation formalism—where feasibility checks are embedded in the action preconditions and state/transition constraints. Our method first applies a systematic generic transformation over the robotic action domain description by utilizing defaults, to be able to perform hypothetical reasoning for diagnosis. Then it computes a smallest set of diagnoses (e.g., broken robots) by means of hypothetical reasoning over the modified formalism.

With such a diagnostic reasoning capability, teams can find potential causes of a detected discrepancy more accurately. Our method also provides further details on a diagnosis (e.g., the actions that could not be executed due to broken robots/components). It enables the monitoring agent(s) to learn from earlier diagnoses and failures the likelihoods of robot components being broken, to utilize feasibility checks as needed, and to decide when broken robots need repairs for successful completion of the specified manufacturing orders.

II. Experimental Evaluations

We performed some experiments over cognitive factory scenarios to investigate the usefulness of diagnosis and repairs.

These scenarios take place in a cognitive toy factory workspace, where a team of multiple robots collectively works toward completion of an assigned manufacturing task. The workspace includes static obstacles. The team is heteroge-

TABLE I
EVALUATION OF DIAGNOSIS AND REPAIRS

Scenario	Number of replanning and total CPU time		
	w/ diagnosis		w/o diagnosis
	w/ repair	w/o repair	
1 charger, 1 wet, 1 dry Discrepancy at Step 5 Diagnosis cardinality=1	3* 10.38	11 9.71	17 82.22
1 charger, 1 wet, 2 dry Discrepancy at Step 12 Diagnosis cardinality=1	3* 0.52	21 18.17	41 5.49
1 charger, 2 wet, 2 dry Discrepancy at Step 8 Diagnosis cardinality=1	2* 4.95	17* 28.51	25 99.68
1 charger, 2 wet, 2 dry Discrepancy at Step 12 Diagnosis cardinality=2	8* 13.41	13 23.04	25 53.14
2 charger, 2 wet, 2 dry Discrepancy at Step 6 Diagnosis cardinality=1	4* 11.78	17 79.38	44 152.07
2 charger, 2 wet, 2 dry Discrepancy at Step 14 Diagnosis cardinality=2	5* 5.00	9* 11.67	36 25.7

* indicates termination with a feasible plan, while others terminate due to non-existence of a feasible plan.

neous, composed of robots with different capabilities. All robots are holonomic and can move from any grid cell to another one following straight paths.

In experiments, we used the ASP solver CLASP [8] (Version 2.1.3), on a Linux server with 16 Intel E5-2665 CPU cores with 2.4GHz and 64GB memory. The results are presented in Table I. There are several interesting observations from these results. 1) If no information is available about the broken robots/components (without diagnosis), no feasible plan could be computed for any of the instances. In comparison to the cases with diagnosis, more iterations and more computation time are required to terminate (due to the non-existence of a feasible plan). 2) When some information is available about broken robots/components (with diagnosis), considering repairs of the broken components allows for computation of feasible plans. Since the number of replanning iterations get smaller with the introduction of repair actions, the computation time also decreases. 3) In most of the cases, we observe that as discrepancy is detected at a later stage, the replanning time decreases. Similarly, as the team size and the cardinality of the diagnosis increase, the replanning time increases.

A video of a sample cognitive factory scenario with two teams, annotated with execution monitoring and diagnosis aspects, is available at http://cogrobo.sabanciuniv.edu/?p=999.

III. RELATED WORK AND CONCLUSION

Our work on diagnosis is similar to conflict-based diagnosis in AI [9, 12]. It is also different since consider dynamic domains with actions and change, rather than a static system like circuits. Also our logical framework is nonmonotonic.

Our work is related to diagnostic reasoning in ASP and action languages [1, 2, 5, 6] due to the common underlying nonmonotonic formalisms. It is different in several ways, with respect to the definition of a diagnosis (e.g., point of failure vs. broken components) and the logical representations (e.g., with/without dummy actions like "break").

Another line of research related to ours is about integration of diagnosis and model-based monitoring [4, 10, 11]. In our work, we consider a dynamic environment that involves multiple robots rather than a specific dynamic system. The behaviors of robots are represented in a more generic form, since it allows planning and explanation generation in addition to consistency checks.

In comparison with these related work, it is important to emphasize the novelty of our hybrid diagnostic reasoning algorithm, both from the perspective of AI and from the perspective of robotics. Thanks to the expressive formalism and solvers of ASP, unlike the related approaches, our algorithm generates diagnoses without introducing dummy actions (since defaults and non-determinism are possible in ASP), it can optimize these diagnoses (since optimization statements are supported by ASP), it can detect multiple interacting faults (since concurrency of actions and ramifications are possible in ASP), and it utilizes feasibility checks as needed (since external atoms are supported by ASP). Also, our algorithm is applicable to robotic applications in dynamic domains.

For further details about the methods, experimental evaluations, and discussion about the related work, we refer the reader to our ICRA 2015 paper [7].

ACKNOWLEDGEMENTS

We acknowledge TUBITAK Grants 111E116, 113M422, 114E491.

REFERENCES

[1] Marcello Balduccini and Michael Gelfond. Diagnostic reasoning with A-Prolog. *TPLP*, 3(4–5):425–461, 2003.

[2] Chitta Baral, Sheila McIlraith, and Tran Cao Son. Formulating diagnostic problem solving using an action language with narratives and sensing. In *Proc. of KR*, 2000.

[3] Gerhard Brewka, Thomas Eiter, and Miroslaw Truszczynski. Answer set programming at a glance. *Commun. ACM*, 54(12): 92–103, 2011.

[4] Daniel Dvorak and Benjamin Kuipers. Model-based monitoring of dynamic systems. In *Proc. of IJCAI*, 1989.

[5] T. Eiter, E. Erdem, W. Faber, and J. Senko. A logic-based approach to finding explanations for discrepancies in optimistic plan execution. *Fundamenta Informaticae*, 79:25–69, 2007.

[6] Esra Erdem, Kadir Haspalamutgil, Volkan Patoglu, and Tansel Uras. Causality-based planning and diagnostic reasoning for cognitive factories. In *Proc. of ETFA*, 2012.

[7] Esra Erdem, Volkan Patoglu, and Zeynep G. Saribatur. Integrating hybrid diagnostic reasoning in plan execution monitoring for cognitive factories with multiple robots. In *Proc. of ICRA*, 2015. Finalist for Best Conference Paper Award, Finalist for Best Cognitive Robotics Paper Award.

[8] Martin Gebser, Benjamin Kaufmann, Andr Neumann, and Torsten Schaub. clasp: A conflict-driven answer set solver. In *Proc. of LPNMR*, 2007.

[9] Johan De Kleer, Alan K. Mackworth, and Raymond Reiter. Characterizing diagnoses and systems. *Artif. Intell.*, 56(2):197–222, 1992.

[10] Franz Lackinger and Wolfgang Nejdl. Integrating model-based monitoring and diagnosis of complex dynamic systems. In *Proc. of IJCAI*, 1991.

[11] Steven James Levine and Brian Charles Williams. Concurrent plan recognition and execution for human-robot teams. In *Proc. of ICAPS*, 2014.

[12] Raymond Reiter. A theory of diagnosis from first principles, 1987.

[13] Zeynep G. Saribatur, Esra Erdem, and Volkan Patoglu. Cognitive factories with multiple teams of heterogeneous robots: Hybrid reasoning for optimal feasible global plans. In *Proc. of IROS*, 2014.