Approaches for action sequence representation in robotics: a review

Hirenkumar Nakawala[1], *Member, IEEE*, Paulo J. S. Gonçalves[2], *Senior Member, IEEE*, Paolo Fiorini[1], *Fellow, IEEE*, Giancarlo Ferrigno[3], and Elena De Momi[3], *Senior Member, IEEE*

***Abstract*— Robust representation of actions and its sequences for complex robotic tasks would transform robot's understanding to execute robotic tasks efficiently. The challenge is to understand action sequences for highly unstructured environments and to represent and construct action and action sequences. In this manuscript, we present a review of literature dealing with representation of action and action sequences for robot task planning and execution. The methodological review was conducted using Google Scholar and IEEE Xplore, searching the specific keywords. This manuscript gives an overview of current approaches for representing action sequences in robotics. We propose a classification of different methodologies used for action sequences representation and describe the most important aspects of the reviewed publications. This review allows the reader to understand several options that do exist in the research community, to represent and deploy such action representations in real robots.**

I. INTRODUCTION

Robot actions are defined as controlling rules, explicit inverse models, behaviours, routines, or policies that could help doing robotic tasks to reach a specific goal [1]. In a short time, robots are becoming competent in performing human activities, e.g. surgery, cleaning, cooking and so forth in the area of service robotics as well as in the industrial robotics [2]. To perform different activities, robots need to execute sequential or parallel action sequence to achieve a real-time task planning, for example to do a "pick-and-place" task or to assist in performing a "cleaning" task. Thereby, robust representation of action sequences is of importance to design efficient reasoning strategies for robot task planning and execution.

In real-world scenarios, to accomplish a task goal, the objects are perceived in the environment, the object specific action is determined, and then is executed, where the action sequence is specified in the robot planner [3]. There are often several ways to execute an action sequence to achieve a specific goal. During the sequential actions execution, the first action should be executed successfully to execute the second one. However, unexpected behaviour is often seen with the robots during task execution, which characterizes underlying deficiency of the planning systems to understand subtle differences between the actions [2]. Particularly in industrial robotics, the planning system considers robot's current actions without its dependency on the previous or next actions and environmental factors. It is often seen that spatiotemporal actions are performed under very high degree of uncertainties e.g. in the service robotics domain. Efficient representation of actions or actions sequences is a bottleneck for designing robust cognitive reasoning strategies and is of special interest to artificial intelligence (AI) community.

Reviewing action sequences representation approaches could be useful to researchers understanding possible approaches for action sequences representation for successful execution of a complex robotic task and planning program as well as achievement of minimal execution duration. In recent years, robotic action sequences are represented using different methodologies, for example using the logic-based methods, knowledge representation techniques, to name a few. This manuscript gives an overview of current approaches for representing actions and its sequences concerning robotic tasks.

II. LITERATURE REVIEW

The review collected the scientific papers which investigate different approaches and methods for action sequences representation for robotic task planning and execution. A search of the literature was done using Google Scholar and IEEE Xplore search engines. Different combinations of keywords were used: "action sequences", "knowledge representation", "logic", "robot", "natural language processing", "open-source frameworks", "ROS", "visual representation languages", and "robot planning".

This section comprises six categories: Situation calculus; Ontologies and other knowledge representation techniques; Temporal and dynamic logic; Action sequences representation in planning languages; State Machines, Petri Nets, and System Flow Charts; and in the last subsection we presented other methodologies, that are also used for representation of action and action sequences. Each subsection starts by defining each category, the methods therein and examples of application found in the literature. Table 1 summarizes the approaches presented in this literature review, where we also indicated the available ROS packages for each of them.

Situation calculus

In situational calculus [4][5], actions or change in the situation either represent a change in the belief about the world or beliefs about a changing state of the world in the form of first-order logic, where the named "actions"

[1]Hirenkumar Nakawala and Paolo Fiorini are with Department of Computer Science, University of Verona, Via S. Francesco, 22, 37129, Verona, Italy. hirenkumarchandrakant.nakawala@univr.it

[2]Paulo J. S. Gonçalves is with Instituto Politcnico de Castelo Branco, Castelo Branco, Portugal, and IDMEC, Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal. paulo.goncalves@ipcb.pt

[3]Giancarlo Ferrigno and Elena De Momi are with Department of Electronics, Information and Bioengineering (DEIB), Politecnico di Milano, Piazza Leonardo da Vinci, 32, 20133, Milan, Italy. giancarlo.ferrigno@polimi.it

represent an entity, which changes. Another first order term called "situation" is used to represent a sequence of actions. In situational calculus, constant S_0 represents the "initial situation", where actions have not occurred yet [4]. After the "initial situation", an action sequence follows the "situation" term in an order from right to left. "Actions" are represented with functional symbols such as "$put\,(A, B)$" and "situations" are represented as first-order terms such as "$do(put\,(A,\ B)\,,\ s)$" which concludes a situation s during the action "put". Possible actions which could be executed in the situation are represented by a binary predicate symbol "$Poss$". For example, a robot r can put the object x in situation s only and only if the robot r is holding the object x, is not heavy and is smaller than the table z, and the robot r is located next to table z.

$$Poss\ \ (put\,(r,\ x)\,,\ s) \quad \equiv\ [(\forall x)\ holding\,(r,\ x,\ s)] \wedge \neg heavy(x) \wedge\ smallerThan\,(x,\ z, s) \wedge nextTo\,(r,\ z, s)$$

However, the above axiom is affected by a problem called "Qualification problem" [6], where the axiom could not infer a possibility to put the object on the table by a robot. The latter could be solved by the non-monotonic solutions e.g. "circumscription" [6]. The predicates whose values become different from situation to situation is called "fluents" [7][8]. To represent the world dynamics, "action pre-condition" and "action effects" are represented. The "action pre-condition" has to be true to execute the action and "action effects" represent the "fluents" which change because of the action. Such specification leads to the "frame problem" [4], where inference engine needs to know which fluents are unaffected by the particular action. The latter creates a problem of combinatorial explosion, and system needs to reason with all of such axioms. A possible solution to this problem depends on the parsimonious representation of action sequences and on the domain expert who specifies the rules. It is important to specify the solution to frame problem as it provides modularity and accuracy in the presented knowledge. There are three main proposals to solve the frame problem:

1) Pednault proposal [9]: This proposal provides a systematic mechanism for generating frame axioms from effect axioms for deterministic actions. However, this proposal misses the parsimonious presentation.
2) The David/Haas/Schubert proposal [10]: Explanation closure axioms provide compact representation. While Schuberts proposal argues that, it is impossible to solve this problem. In closure axioms, the action theory which contains the effects of actions can be manually constrainted by adding closure axioms to theory.
3) Another option is to use successor state axioms [5] where the domain expert enumerates possible effect axioms, where fluents value can be changed.

Benefits of using situation calculus are that the primitive actions can be represented efficiently. However, complex actions, which depend on conditions, iterations and with non-deterministic choice, could not be represented. The "projection" task [11] defines a situation, where performing a given sequence of actions would be true. The "legality" task [4] follows the "projection" task and conforms if a condition would hold after performing a sequence of actions, but not whether that sequence can, in fact, be properly executed. The researchers call the situation "legal" if it is an initial situation or the results of performing an action whose preconditions are satisfied starting in a legal situation, which is determined by the "legality" task. Thus, situation calculus is an important representation for designing discrete event scenarios with high-level robot control.

Finzi et al. [12] have represented Temporal Flexible Situation Calculus (TFSC) which is a declarative temporal model. TFSC explicitly represents processes of the robot system [13]. GOLOG (alGOl in LOGic) [14] is an important programming language, where actions are reduced to primitive actions which refer to actions in the real world, like picking and moving the objects. Instead of machine states, in GOLOG, the reasoning strategy employed involving world states. Execution of the complex actions require logical reasoning about how world changes. Over the past years, GOLOG has been extended to be used in realistic domains, especially in robotics such as concurrent actions [15], time [16], continuous change [17], an explicit knowledge representation [18], sensing [19] and stochastic actions [20].

Ontologies and other knowledge representation techniques

With the evolution of knowledge-based systems, action sequences are formalized using the separation of terminology, called "TBox" from the logical assertions, called "ABox" where real-world instances are specified in action libraries [21]. Action libraries contain a set of actions that are frequently used within a given domain. Tenorth et al. [22] represented actions based on the logical inference steps, which were implemented in Description Logic (DL) and Web Ontology Language (OWL). Abstract structure for an action in a task is specified using class restrictions in OWL. The class abstraction and restrictions include objects references, which are to be manipulated along with object locations. Class restrictions were obtained from a comprehensive ontology, called "KnowRob" [23], which contains a taxonomy of more than 130 actions commonly observed in everyday activities.

In another work [24], three knowledge levels were defined for a class called "action knowledge", i.e. *primitive behaviour level*, which includes motion as preconditions and action effects as well as feature extraction algorithms for perception; *task level* which defines long-term goals for the symbolic preconditions and action effects; and the *subtask level*, which defines functions. The authors have specified "preActors" which perceives objects that an action uses as inputs and "postActors" which are generated as its results. The "preActors" includes the role of an object in determining

action, initial locations and states. The change in the state of an object is specified as "objectOfStateChange". Perceptual actions can detect an object, and actions can substantially change objects by transforming them into another one, destroying or adjoining them with another one. In this work, the robot uses effect axioms to predict the outcome of actions as the resulting world state. Similarly, the ontology and production rules have been used in [25] for the execution planning in surgery based on the perceived instruments. However, the task execution has not been done by the robots. Kunze et al. [26] have introduced a concept of "Action tree". "Action tree" is a hierarchical tree, which consists of sub-actions associated with dependencies on robots capabilities and considers robots experience on performing some action. The "Action tree" is specified in a semantic robot description language (SRDL). The dictionary of primitive actions ("DictAct") [27] is defined with four attributes, e.g. pre-condition, post-condition, input and results. In another work of knowledge representation in action sequencing [28] four levels of hierarchical structure was represented, i.e. control primitive level, control skill level, skill level and task level, where an action sequence is considered as a process. With "RoboEarth" action language [29], the action recipes consisting formal and grounded representation are stored in a database called "Action recipe database", and shared through "RoboEarth". However, these action recipes can only be used to execute pre-programmed tasks, and lack flexibility if applied in the dynamic environment because it cannot predict the state of the environment as well as of the robot. The cost of carrying out these action is also not specified so the action recipes could not be optimized during the execution planning. A hybrid formalism based on DL, called "NeoClassic" [30] , is used to define actions as a sequence of a start instance and an end instance on a process, where action sequences are represented by the situations occurring concurrently, i.e. with the preconditions and the effect of the actions.

Temporal and dynamic logic

Temporal and dynamic logic was used for translating the robot instructions into action goals and action sequences [31]. When applied together, temporal logic translates provided robot instructions into action goal, and then dynamic logic is used to detect the action sequence, which used in execution planning. Here, instead of translating instructions into actions, the instructions are transformed into propositions and then an action sequence is derived. Recently, Kress-Gazit et al. [32] have specified temporal logic of robot actions for motion and task planning and for generating robot controllers.

Action sequences representation in planning languages

Various action representations have been applied for the execution planning in robotics. For example, STRIPS plan language attempts to find a sequence of operators, which proves the given goal. STRIPS is a declarative description of actions, or operations, pre- and post-conditions, and effects, where the description of the world are specified in first-order predicate calculus [33]. An action applies to any state that specifies the precondition. Otherwise, the action does not affect. Results of executing an applicable action a in a state s', is same as s, where the action has started, except that any positive literal P, in the effect of a, is added to s', and any negative literal $\neg P$ is removed from s'. STRIPS only permits positive literals and conjuctions. However, representation with STRIPS was found to be insufficient to represent real world action sequences and complex predicates. As an advancement to STRIPS, Action Description Language (ADL) was developed. ADL expressiveness lies between STRIPS and situational calculus, which supports an open-world assumption. Pednault [34] stated that STRIP could be more expressive if effects of an operator is conditional. The ADL is finally evolved to ADL-C which can represent static and dynamic laws. Contrary to STRIPS, ADL also allows negative literals and disjunctions. One of the first autonomous mobile robots, "Shakey", used symbolic representations to determine action sequences that would achieve its goal [35]. The symbolic representation allows dealing with continuous belief state and the complex problems could be handled efficiently. The Planning Domain Description Language (PDDL) is used to describe this abstract symbolic action, its goals, and states [36]. Hierarchical Task Network (HTN) [37] decomposes action description into hierarchical tasks which could correspond to STRIPS actions. HTN is structured hierarchically from goals to the low-level commands. To optimize the performance, an order of actions is changed at different levels in the hierarchy. HTN modifies itself to adapt to action sequences, where the planner is integrated into the optimization process.

State Machines, Petri Nets, and System Flow Charts

Programming robots to perform the tasks, i.e. what the robot should do, in unknown environments, is complex. For simple systems and with a low autonomy level, task scripting is enough. However, the programmer must adequately ensure that the robot can evolve from failure modes, e.g. a robot stop due to an obstacle. When an increase of robot autonomy is needed, the robot must be capable to plan, execute, re-plan the tasks that need to be done. Task planning is the answer for the previous issue, also allowing to scale for more complex scenarios. ROS (Robot Operative System), as the standard choice for development of robotic applications currently have a set of functionality for task plan execution, e.g. SMACH ("State MACHines") [38] (that implements state machines) and Petri Net Plans (PNP)-ROS [39] (that implements Petri Nets). Other frameworks do exist as shown in Table 1, e.g. the System Flow Chart (SFC) framework [40]. SMACH is a python library that allow to design complex robot tasks and can change the robot behaviour. It is based on hierarchical state machines. In this work, the framework is defined using "State" and "Container". "State" defines the current execution step and its outcomes, that will enable further states. "Container" represents collections of one or more robot states that implement the robot execution

policies. The main drawback of this functionality is that the implementation is very code intensive and can be tricky to develop very complex systems. PNP-ROS is in fact a bridge between the PNP library and ROS, that enables execution of PNP in ROS. PNP comprises basic structures, that can represent the execution phases of actions needed to perform a given task, e.g. "initiation", "execution", "termination" and "transitions". The latter are very important because it represents the events that have conditions to control state triggering. The PNP library contains an execution engine, a tool to generate the plans "PNPgen", and bridges to Nao and Pepper robots, along the ROS bridge (PNP-ROS). "PNPgen" is a tool that translates the output of some planning system to PNP, e.g. from the "ROSPlan" framework, or Hierarchical Agent-based Task Planner (HATP), or Markov Decision Process (MDP) policies. To control the actions execution and ROS topics, it uses the "actionlib" protocol, which is also defined in ROS. The framework allows definition of the execution rules: interrupts (conditions that can determine the interruption of an action), recovery (how to recover from an interrupt), social and/or ethical norms, parallel execution. Simple example rules can be defined:

"**If** $BatteryLow$ **during** $RobotOperation$ **then** $Recharge$; $failed\_plan$ "

"**If** $PersonCloseToRobot$ **during** $RobotMotion$ **then** $skip\_action$; $WaitForNoPersonCloseToRobot$; $RestartAction$"

In [41], a 3-layered petri net model is proposed, which comprises the environment, action executor, and finally the action coordination layers. The environment layer represents the environment discrete-state event-driven dynamics, that results from the human robot actions and the physical environment. The action executor layer starts to represent the actions, e.g. changes in the environment and the conditions that they can occur. The action coordination layer is the one where the task plans, composition of actions, are defined. System Flow Chart (SFC) is a well-known visual programming language for PLC's, Programmable Controller. In [40] and [42] present the developing stages of a knowledge-based framework, to define and control the execution of tasks, using SFC's. Moreover, the system is capable, from high-level tasks description, to develop the SFC's, e.g. the robot code to be executed (actions, sequence of actions and its transitions).

Other representations

Motion description language (MDL) [43] provides a basis for the control of robots using sets of behaviours or robotic actions, timers, and events. MDL captures composition of this behaviour, which is a reaction to environmental events. "Maneuver automata" [44], which can be seen as a subset of MDLs, use regular language expressions to solve the motion task. Here, each string in the language maps to a possible motion behaviour of the system. Each symbol specifies a motion primitive that belongs to a library of necessary motions. However, the problem with symbolic representation is an incomplete knowledge representation, where the robot action sequences are not generated. In [45], authors have presented "action mode representation". In "action mode representation", an action mode is considered as a unit of behaviour. For example, for autonomous mobile robot use case, a behaviour, i.e. robot moving to follow one motion, is represented with many action modes and transitions between them. "action mode representation" is useful, for defining action pre-conditions, where it is used to represent sensor-based behaviours. A hypergraph, i.e. AND/OR graphs [46] were used to represent the action sequences. In AND/OR graph, the nodes represent actions. Planning is done using the tree search with the objective to reduce the number of nodes in the representation. Recently, temporal AND/OR graphs [47] have also been used to represent the action sequences for a robot tasks, e.g. assembly planning. However, such graphs could not be represented automatically which has limited application in real-time systems. There are several studies [48] [49] [50] focusing on the learning action sequences based on robot observations or demonstrations, which overcomes the limitation of learning sequential as well as concurrent actions. However, the representation obtained from demonstrations could be difficult to generalize. A hierarchical representation is introduced to generalize an action sequence according to the explanation-based theory [51] from a single demonstration and to combine elementary operations to accomplish more complex compound tasks. Temporal Action Logic (TAL) [52] presents a narrative based formalism for reasoning about actions and change where narratives include action type specifications, causal or dependencies constraints, observations at a specific point in time, and action occurrences through duration. Influence diagrams [53] have also been used to represent the action sequences, where actions are represented as decision nodes, outcome states (including goals) as probabilistic state nodes, and preferences using value nodes. Depending on the node types they connect, the arcs in the influence diagram represent probabilistic information dependence.

III. DISCUSSION AND CONCLUSION

Following a growing need for autonomous systems, new techniques have been proposed to handle the robot actions and its sequences. Research studies have been performed for optimizing, understanding and managing the action sequences. We presented a critical review on the action representation based on the different approaches e.g. logic and knowledge representation techniques, and so forth focusing on works of modelling action sequences for task planning and execution. This review, summarized in Table 1, allows a more in-depth understanding and different approaches representing the action sequences in the robotics field. From this review the reader understands several options that do exist in the research community, to represent and deploy such action representations and frameworks in real robots. For example the review clearly states if exists a ROS implementation, and

TABLE I

AN OVERVIEW OF ACTION SEQUENCES REPRESENTATION APPROACHES

No.	Publication	Representation approach	ROS implementations & packages	Applications
1.	McCarthy *et al.* 1969; Levesque *et al.* 1998	Situation calculus; GOLOG programming lnaguage	No	Service and industrial robotics
2.	Thielscher 1998	Situation calculus; Fluent calculus	Yes "CRAM_language"	Service and industrial robotics
3.	Levesque *et al.* 1997	GOLOG programming language	No	Service and industrial robotics
4.	Tenorth *et al.* 2013; Kunze *et al.* 2011	Ontology and Prolog predicates	Yes "knowrob"	Service robotics
5.	Lim *et al.* 2011; Ji *et al.* 2012;	Ontology	No	Service robotics
6.	Aramaki *et al.* 1999	Hierarchical task representation; state space model	No	Industrial robotics
7.	Janssen *et al.* 2014	Ontology	Yes "CRAM_language"	Service robotics
8.	Chella *et al.* 2001	Description Logics	No	Service and industrial robotics
9.	Dzifeak *et al.* 2009	Temporal and Dynamic logics	No	Service robotics
10.	Fikes *et al.* 1971	STRIPS plan langauge	Yes "hrl_clickable_world"; "rhbp - ROS Hybrid Behaviour Planner"	Service and industrial robotics
11.	Pednault 1987	Action Description Language (ADL)	No	Industrial robotics
12.	Fox *et al.* 2003	PDDL 2.1	Yes "ROSPlan"	Service and industrial robotics
13.	Erol *et al.* 1994	Hierarchical Task Network (HTN)	Yes "behavior_tree"	Service and industrial robotics
14.	Bohren *et al.* 2010	State machine	Yes "SMACH"	Service and industrial robotics
15.	Ziparo *et al.* 2011	Petri Nets	Yes "PNP_ros"	Service and industrial robotics
16.	Brockett *et al.* 1998	Motion Description Language (MDL);	No	Service and industrial robotics
17.	Frazzoli *et al.* 2005	Maneuver Automata	No	Service and industrial robotics
18.	Suzuki *et al.* 1991	Action mode representation	No	Service and industrial robotics
19.	DeMello *et al.* 1996	AND/OR graph;	No	Service and industrial robotics
20.	DeJong 1996	Temporal action logics	No	Service and industrial robotics
21.	Ogasawan *et al.* 1993	Influence diagrams	No	Service and industrial robotics

its package names. Based on the initial development framework and on the follow-up implementations, applications do exist for service and/or industrial applications.

Although is difficult and nearly impossible to have the exact number of robots that have implemented each action representation framework, a good indicator can be the number of citations of each seminal paper in Google Scholar. According to the literature review, the most popular action representation framework, implemented in ROS, is "KnowRob" with 244 paper citations. "CRAM" and "SMACH" follows with more than one hundred seminal paper citations. "PNP_ros" is also popular in the research community with 76 citations.

ACKNOWLEDGMENT

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, grant agreement No. 742671. This work was supported by European Union's Horizon 2020 research and innovation programme under grant agreement No. H2020-ICT-2016-732515. This work was supported by FCT through IDMEC under LAETA, project UID/EMS/50022/2013, and project 0043-EUROAGE-4-E (POCTEP Programa Interreg V-A Spain-Portugal).

REFERENCES

[1] F. Stulp, M. Beetz, Refining the execution of abstract actions with learned action models. Journal of Artificial Intelligence Research. 32: 487 523, 2008.

[2] R. de Kleijn, G. Kachergis, B. Hommel, Everyday robotic action: lessons from human action control. Front. neurorobot., 2014.

[3] P. Haazebroek, S. van Dantzig, B. Hommel, A computational model of perception and action for cognitive robots. Presented at Cong Process. 12(4): 355-365, 2011.

[4] J. McCarthy and P. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, Machine Intelligence, 4: pp. 463-502, 1969.

[5] H. Levesque, F. Pirri and R. Reiter, Foundations for the situation calculus. Electronics Transactions on Artificial Intelligence, 2(3-4), pp. 159-178, 1998.

[6] J. McCarthy, Human-level AI: the logical road, available at http://jmc.stanford.edu/articles/logicalai.html., 2018. Accessed on 08.02.2018

[7] M. Thielscher, Introduction to the fluent calculus, Electronics transactions on artificial intelligence, 2 (3-4), pp. 179-192, 1998.

[8] M. Thielscher, Reasoning robots - the art and science of programming robotic agents. Volume 33 of applied logic series, Springer, Dordrecht.

[9] E.P.D. Pednault, ADL: Exploring the Middle Ground between STRIPS and the Situation Calculus. In R.J. Brachman, H. Levesque, and R. Reiter, editors, Presented at the First International Conference on Principles of Knowledge Representation and Reasoning (KR'89), pp. 324-332. Morgan Kaufmann Publishers, Inc., San Mateo, California, 1989.

[10] L.K. Schubert, Monotonic Solution of the Frame Problem in the Situation Calculus: an Efficient Method for Worlds with Fully Specified Actions. In H.E. Kyberg, R.P. Loui, and G.N. Carlson, editors, Know ledge Representation and Defeasible Reasoning, pages 23, Kluwer Academic Press, Boston, Mass., 1990.

[11] R. Reiter, The projection problem in the situation calculus: a soundness and completeness result, with an application to database updates, Presented at artificial Intelligence planning systems, the first conference on (AIPS 1992), pp. 198-203, 1992.

[12] A. Finzi, F. Pirri, Representing flexible temporal behaviours in the situation calculus. Presented at IJCAI, 2005.

[13] M. Gianni, P. Papadakis, F. Pirri, M. Liu, F. Pomerleau, F. Colas, K. Zimmermann, T. Svobod, T. Petricek, G.J.M Kruijff, et al. A unified framework for planning and execution-monitoring of mobile robots. Presented in a workshop at the Twenty-Fifth AAAI Conference on Artificial Intelligence, 2011.

[14] H. Levesque, R. Reiter, Y. Lesprance, F. Lin, and R. Scherl, GOLOG: A logic programming language for dynamic situations, Journal of Logic Programming, 31: 59-84, 1997.

[15] G. De Giacomo, Y. Lesprance, ConGolog A concurrent programming language based on the situation calculus, Artificial Intelligence, 121 (1-2): pp. 109-169, 2000.

[16] R. Reiter, Sequential, temporal GOGOL. Presented at Principles of Knowledge Representation and Reasoning: the Sixth International Conference (KR'98), pages 547–556, Trento, Italy, 1998.

[17] C. Baral, T. C. Son, Extending ConGolog to allow partial ordering, Presented atATAL-99, pp. 188-204, 1999.

[18] J. A. Baier, C. Fritz, Beyond classical planning: procedural control knowledge and preferences in state-of-the-art planners, Presented at 23rd AAAI conference on artificial intelligence, Nectar Track, pp. 1509-1512, Chicago, Illinois, 2008.

[19] G. Lakemeyer, On sensing and off-line interpreting in golog, Logical foundations for cognitive agents, pp. 173-187, Springer, Berlin, 1999.

[20] H. Grosskreutz, G. Lakemeyer, Turning high-level plans into robot programs in uncertain domains, Presented at ECAI-2000, Berlin, Germany, 2000.

[21] F. Baader, I. Horrocks, and U. Sattler, Chapter 3. Description Logics, In F. van Harmelen, V. Lifschitz and B. Porter, editors, Handbook of knowledge representation, Elsevier, 2007.

[22] M. Tenorth, M. Beetz, A Unified Representation for Reasoning about Robot Actions, Processes, and their Effects on Objects. Presented at IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura, Algarve, Portugal, 2012.

[23] M. Tenorth, M. Beetz, KnowRob – A Knowledge Processing Infrastructure for Cognition-enabled Robots, International Journal of Robotics Research (IJRR), 32 (5):, pp. 566-590, 2013.

[24] G. H. Lim, I. H. Suh, H. Suh, Ontology-based unified robot knowledge for service robots in indoor environments, IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans, 41 (3): pp. 492-509, 2011.

[25] H. Nakawala, G. Ferrigno, E. De Momi, Development of an intelligent surgical training system for Thoracentesis, Artificial Intelligence in Medicine, 84: pp. 50-63, 2018.

[26] L. Kunze, T. Roehm, and M. Beetz, Towards semantic robot description language. Presented at in IEEE Int. Conf. Robotics and Automation (ICRA), 2011.

[27] Z. Ji, R. Qiu, A. Noyvirt, A. Soroka, M. Packianather, R. Setchi, D. Li and S. Xu, Towards automated task planning for service robots using semantic knowledge representation, Presented at Industrial informatics (INDIN), 10th IEEE International Conference On, Beijing, China, 2012.

[28] S. Aramaki, I. Nagasawa, S. Kurono, T. Nagai, T. Morita amd T. Suetsugu, The knowlegde representation and programming for robotic assembly task, Presented at Assembly and Task Planning (ISATP 99), IEEE International Symposium On, Porto, Portugal, 1999.

[29] R. Janssen, E. van Meijl, D. Di Marco, R. van de Molengraft, M. Steinbuch, Integrating planning and execution for ROS enabled service robots using hierarchical action representations, Presented at Advanced Robotics (ICAR), 16th International Conference on, Montevideo, Uruguay, 2014.

[30] A. Chella, S. Gaglio, R. Pirrone, Conceptual representations of actions for autonomous robots, Robotics and Autonomous Systems, 34 (4): pp. 251-263, 2001.

[31] J. Dzifcak, M. Scheutz, C. Baral, and P. Schermerhorn, What to do and how to do it: Translating natural language directives into temporal and dynamic logic representation for goal management and action execution, Presented at IEEE International Conference on Robotics and Automation, pp. 41634168, 2009.

[32] H. Kress-Gazit, G. Fainekos, and G. Pappas G., Temporal-Logic-Based Reactive Mission and Motion Planning, IEEE Transactions on Robotics, 25 (6), pp. 1370-1381, 2009.

[33] R. O. Fikes, N. J. Nilsson, STRIPS: A new approach to the application of theorm proving to problem solving. Presented at IJCAI, London, U.K., September 1-3, 1971.

[34] Pednault. Formulating multi-agent dynamic-world problems in the classical planning framework. In Michael Georgeff and Amy Lansky, editors, Reasoning about actions and plans pages 47-82. Morgan Kaufmann, San Mateo, CA, 1987.

[35] N. J. Nilsson, Shakey the Robot. SRI Tech. Note 323, Menlo Park, CA, 1984.

[36] M. Fox, D. Long, PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. Journal of Artificial Intelligence Research (JAIR), 2003: 20: 61124, 2003.

[37] K. Erol, J. Hendler, and D. Nau, Htn planning: Complexity and expressivity, Presented at the National Conference on Artificial Intelligence. John Wiley & Sons LTD, pp. 1123-1123, 1994.

[38] J. Bohren and S. Cousins, The SMACH High-Level Executive [ROS News], in IEEE Robotics Automation Magazine, vol. 17, no. 4, pp. 18-20, Dec. 2010.

[39] V. A. Ziparo, L. Iocchi, P Lima, D. Nardi, P. Palamara. Petri Net Plans - A framework for collaboration and coordination in multi-robot systems. Autonomous Agents and Multi-Agent Systems, vol. 23, no. 3, 2011.

[40] M. Haage, J. Malec, A. Nilsson, M. Stenmark, E. A. Topp, Semantic Modelling of Hybrid Controllers for Robotic Cells, Procedia Manufacturing, Volume 11, Pages 292-299, 2017.

[41] H. Costelha and P. Lima. Robot task plan representation by petri nets: modelling,identification, analysis and execution. Autonomous Robots, 2012

[42] M. Stenmark, J. Malec, A. Stolt, From High-Level Task Descriptions to Executable Robot Code. In: Filev D. et al. (eds) Intelligent Systems'2014. Advances in Intelligent Systems and Computing, vol 323. Springer, 2015.

[43] R. W. Brockett On the computer control of movement, Presented at IEEE Int. Conf. Robot. Autom., vol. 1, pp. 534540, 1998.

[44] E. Frazzoli, M. A. Dahleh, and E. Feron. Maneuver-based motion planning for nonlinear systems with symmetries. IEEE Trans. Robot. 21 (6): 10771091, 2005.

[45] S. Suzuki, S. Yuta, Analysis and description of sensor based behaviour program of autonomous robot using action mode representation and ROBOL/0 language. Presented at IEEE/RSJ International Workshop on Intelligent Robots and Systems (IROS1991), Osaka, Japan, Nov 3-5, 1991.

[46] L. S. H. De Mello, A. C. Sanderson, AND/OR graph representation of assembly plans, IEEE Transactions on robotics and automation, 6(2), 1990.

[47] H. liu, Y. Zhang, W. Si, X. Xie, Y. Zhu and S. C. Zhu, Interactive robot knowledge patching using augmented reality. Accepted to represent at International Conference on Robotics and Automation (ICRA), Australia, 2018.

[48] S. R. Ahmadzadeh, A. Paikan, F. Mastrogiovanni, L. Natale, P. Kormushev, D. G. Caldwell, Learning symbolic representations of actions from human demonstrations, Presented at Robotics and Automation (ICRA), IEEE International Conference on, Seattle, USA , 2015.

[49] M. Pardowitz, S. Knoop, R. Dillmann, R. D. Zllner, Incremental learning of tasks from user demonstrations, past experiences, and vocal comments, IEEE Transactions on systems, man, and cybernetix- part B: cybernetix, 37(2), pp. 322-332, 2007.

[50] J. Chen, A. Zelinsky, Programming by demonstration: Coping with suboptimal teaching actions, Int. J. Rob. Res., vol. 22, no. 5, pp. 299-319, May 2003.

[51] R. M. G. DeJong, Explanation-based learningAn alternative view, Mach. Learn., vol. 1, pp. 145-176, 1986.

[52] M. Magnusson, P. Doherty, Deductive planning with temporal constraints using TAL, Represented at PCAR06 2006 International Symposium on Practical cognitive agents and robots, pp. 141-152, Perth, Australia, Nov 27-28, 2006.

[53] G. H. Ogasawara, S. J. Russell, Planning using multiple execution architectures, Presented at IJCAI93, Chambery, France, 1993