Critical Section Macros – New Results (Extended Abstract)

Lukáš Chrpa[1], **Mauro Vallati**[2]

[1]Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague, Prague, Czechia
[2]School of Computing and Engineering, University of Huddersfield, Huddersfield, UK
chrpaluk@cvut.cz, m.vallati@hud.ac.uk

Introduction

Macro-operators, or *macros* for short, are a well-known technique for reformulating planning domain models that has a long history of use (Dawson and Siklóssy 1977; Korf 1985). In a nutshell, macros are encoded in the same way as ordinary actions while encapsulating sequences of actions. Thus, macros can be used in a planner-independent fashion. With macros, plans can be found in fewer steps, however, at the cost of increased branching factor that might be detrimental to the performance of planners. Hence, macros are subject to their utility value (Minton 1988) that determines how useful the macro can be (for a given planner).

A wide range of techniques tackles the problem of generating effective macros from different perspectives. Some techniques address weaknesses of specific planning techniques, such as delete-relaxation based heuristics (Botea et al. 2005; Coles and Smith 2007). Another technique leverages genetic programming to determine the usefulness of macros for a given planner (Newton et al. 2007). Addressing one of the main weaknesses of macros – a possibly large branching factor that they might introduce – is the key feature of the MUM technique (Chrpa, Vallati, and McCluskey 2014) that generates "instance-wise" macros that have a comparable number of instances to ordinary actions. Aiming at longer macros that might be frequently used in plans is another strategy for generating effective macros (Chrpa and Siddiqui 2015). *Critical Section Macros (CSMs)* are a recent technique that is inspired by Critical Sections in parallel computing in which some resources need to be locked (Chrpa and Vallati 2019, 2022). In the context of planning, such resources are, for example, robotic hands that can carry at most one object, trucks with limited capacity, etc. Noteworthy, there also exist macro generating techniques that deal with features beyond classical planning, such as ADL (Hofmann, Niemueller, and Lakemeyer 2020), durative actions (Bortoli et al. 2023), or black-box planning (Allen et al. 2021).

This extended abstract presents new results on the use of CSMs on the IPC-2023 learning track domains and four planners, including the winner of the agile track of the IPC-2023, and a lifted planner.

Critical Section Macros

The key element of *Critical Section Macros (CSMs)* is "lockable" resources that, in the planning context, are represented by pairs of grounded predicates, or atoms, p, q such that p represents a free resource while q represents a locked resource. Formally speaking, p and q have to be mutually exclusive and either (i) $args(p) \subset args(q)$, or (ii) $name(p) = name(q)$ and $|args(p) \cap args(q)| + 1 = |args(p)| = |args(q)|$. The meaning of (i) is that extra arguments of q represent the reason for locking the resource, for example, a robotic hand (when locked) holding a block. The meaning of (ii) is that both p and q have the same name, and they differ in exactly one argument that represents the reason for locking the resource, for example, the remaining capacity of the truck. Also, (ii) might represent multiphased locks (for example, trucks can accommodate multiple packages).

A simple CSM (having a single single-phased lock) starts with a p, q*-locker*, an action that deletes p and achieves q, and ends with a p, q*-releaser*, an action that deletes q and achieves back p. In between, a simple CSM can have q*-users*, actions having q in their preconditions, and *gluing* actions that connect other actions in the macro. An example of a gluing action is a *drive* action that moves a truck from one location to another that connects loading and unloading a package (*load-drive-unload* is an example of a simple CSM). Simple CSMs can be learned from training plans.

More complex, so-called *compound*, CSMs that contain multiple or multi-phased locks (or both) can be constructed by iterative application of (simple) CSM learning (learnt CSMs are considered as ordinary actions in the following iteration). For more details about CSMs, the interested reader is referred to (Chrpa and Vallati 2022).

Results

In this section we provide some new results achieved by exploiting CSMs on the IPC-2023 learning track benchmark domains. Training plans were generated by the LAMA planner (Richter and Westphal 2010) on training instances p20.pddl – p29.pddl. Other settings were the same as in (Chrpa and Vallati 2022).

Besides LAMA (in the lama-first setting), we considered Dual BFWS in Poly-seq settings (Lipovetzky et al. 2018), Decstar (Gnad, Torralba, and Shleyfman 2023), the winner

Planner	Coverage			PAR10			IPC quality score		
	O	S	C	O	S	C	O	S	C
Easy instances									
LAMA	161	**167**	**167**	950	**651**	**651**	**154.3**	146.3	144.0
BFWS	152	**164**	**164**	1400	**804**	**804**	**146.4**	141.8	140.2
Decstar	162	**167**	**167**	901	**651**	**651**	144.6	**150.3**	149.1
PL	129	**131**	**131**	2559	**2454**	**2454**	**121.2**	118.5	118.2
Medium Instances									
LAMA	127	**129**	126	2662	**2564**	2716	**122.9**	110.5	106.2
BFWS	**99**	98	98	**4058**	4107	4107	**97.6**	78.4	77.9
Decstar	**114**	109	110	**3305**	3565	3518	**105.5**	95.2	95.5
PL	61	**63**	62	5979	5888	**5935**	**59.8**	54.0	52.9
Hard Instances									
LAMA	43	47	**59**	6872	6683	**6083**	42.7	38.3	**50.2**
BFWS	**33**	32	**33**	**7369**	7422	7374	**33.0**	23.9	24.5
Decstar	41	42	**55**	6962	6915	**6261**	40.7	33.4	**46.4**
PL	**15**	6	6	**8273**	8704	8705	**15.0**	4.3	4.3

Table 1: Results comparing the (O)riginal models with CSM enhanced models, (S)imple CSMs, and (C)ompound CSMs. PL stands for Powerlifted.

of IPC-2023 agile track, and Powerlifted (Corrêa and Seipp 2022) with alt-bfws1 search, ff evaluator and yannakakis generator. The time limit for solving a testing instance was set to 900 seconds, and the memory limit to 4 GB. Experiments were run on AMD Ryzen 9 7950X 16-Core cluster.

Table 1 presents the results of the evaluation in three metrics: total coverage, average PAR10 (actual runtime for solved problems, or 10 times the time limit for unsolved problems) and total IPC quality score (the ratio of the minimum cost of the plan across the encodings and the actual cost of the plan, or 0 if the problem is unsolved). CSMs were generated in 6 domains, while in 4 domains CSMs were not generated (e.g., because of their infrequent use, or absence of lockable resources). In each domain, there were, in total, 90 problem instances, divided into three groups – easy, medium, hard – each consisting of 30 problem instances.[1]

Easy problem instances are solved in a fraction of a second, except in the Floortile domain, in which CSMs yield better coverage among all the planners. Powerlifted did not solve any instance in the Ferry domain because it contains negative preconditions. The results for medium problem instances are more mixed, mainly because of the blocksworld and sokoban domains, slightly favourable for CSMs only for Lama and Powerlifted. In hard problem instances, CSM led to considerably better results in the Ferry domain for Lama and Decstar. Quality-wise, CSMs usually led to worse results, which is not that surprising given the aim of macros to reduce plan generation time.

The results demonstrate that CSMs can still improve the performance of state-of-the-art planners such as Decstar. Powerlifted, however, did not benefit much from CSMs, especially in the Miconic domain, despite not having to do grounding.

[1]Detailed results are at https://github.com/lchrpa/CSMs.

Acknowledgements

This research is supported by the Czech Science Foundation (project no. 25-18003S) and by the European Union under the project ROBOPROX (reg. no. CZ.02.01.01/00/22_008/0004590).

References

Allen, C.; Katz, M.; Klinger, T.; Konidaris, G.; Riemer, M.; and Tesauro, G. 2021. Efficient Black-Box Planning Using Macro-Actions with Focused Effects. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 20211*, 4024–4031. ijcai.org.

Bortoli, M. D.; Chrpa, L.; Gebser, M.; and Steinbauer-Wagner, G. 2023. Enhancing Temporal Planning by Sequential Macro-Actions. In *Logics in Artificial Intelligence - 18th European Conference, JELIA 2023*, 595–604.

Botea, A.; Enzenberger, M.; Müller, M.; and Schaeffer, J. 2005. Macro-FF: Improving AI Planning with Automatically Learned Macro-Operators. *Journal of Artificial Intelligence Research (JAIR)*, 24: 581–621.

Chrpa, L.; and Siddiqui, F. H. 2015. Exploiting Block Deordering for Improving Planners Efficiency. In *IJCAI*, 1537–1543.

Chrpa, L.; and Vallati, M. 2019. Improving Domain-independent Planning via Critical Section Macro-Operators. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, 7546–7553.

Chrpa, L.; and Vallati, M. 2022. Planning with Critical Section Macros: Theory and Practice. *J. Artif. Intell. Res.*, 74: 691–732.

Chrpa, L.; Vallati, M.; and McCluskey, T. L. 2014. MUM: A Technique for Maximising the Utility of Macro-operators by Constrained Generation and Use. In *ICAPS*, 65–73.

Coles, A.; and Smith, A. 2007. Marvin: A Heuristic Search Planner with Online Macro-Action Learning. *J. Artif. Intell. Res.*, 28: 119–156.

Corrêa, A. B.; and Seipp, J. 2022. Best-First Width Search for Lifted Classical Planning. In *Proceedings of the Thirty-Second International Conference on Automated Planning and Scheduling, ICAPS 2022*, 11–15. AAAI Press.

Dawson, C.; and Siklóssy, L. 1977. The Role of Preprocessing in Problem Solving Systems. In *Proceedings of IJCAI*, 465–471.

Gnad, D.; Torralba, A.; and Shleyfman, A. 2023. Decstar. In *The Tenth IPC, the Classical Track*.

Hofmann, T.; Niemueller, T.; and Lakemeyer, G. 2020. Macro Operator Synthesis for ADL Domains. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*.

Korf, R. 1985. Macro-operators: A weak method for learning. *Artificial Intelligence*, 26(1): 35–77.

Lipovetzky, N.; Ramirez, M.; Frances, G.; and Geffner, H. 2018. Best-First Width Search in the IPC2018: Complete, Simulated, and Polynomial Variants. In *The Ninth IPC. the Deterministic Track*.

Minton, S. 1988. Quantitative Results Concerning the Utility of Explanation-Based Learning. In *Proceedings of AAAI*, 564–569.

Newton, M. A. H.; Levine, J.; Fox, M.; and Long, D. 2007. Learning Macro-Actions for Arbitrary Planners and Domains. In *Proceedings of ICAPS*, 256–263.

Richter, S.; and Westphal, M. 2010. The LAMA planner: guiding cost-based anytime planning with landmarks. *Journal Artificial Intelligence Research (JAIR)*, 39: 127–177.