Planning With Concurrent Transaction Logic*

Reza Basseda

Stony Brook University,
Stony Brook, NY, 11794, USA
rbasseda@cs.stonybrook.edu

Abstract. Automated planning has been the subject of intensive research and is at the core of several areas of AI, including intelligent agents and robotics. In this thesis proposal, we argue that Concurrent Transaction Logic ($\mathcal{CTR}$) is a natural specification language for planning algorithms, which enables one to see further afield and thus discover better and more general solutions than using one-of-a-kind formalisms. Specifically, we take the well-known *STRIPS* planning strategy and show that $\mathcal{CTR}$ lets one specify the *STRIPS* planning algorithm easily and concisely, and extend it in several respects. For instance, we show that extensions to allow indirect effects and to support action ramifications come almost for free. The original *STRIPS* planning strategy is also shown to be incomplete. Using concurrency operators in $\mathcal{CTR}$, we propose a non-linear *STRIPS* planning algorithm, which is proven to be complete. Moreover, this thesis proposal outlines several extensions of *STRIPS* planning strategy. All of the extensions show that the use of $\mathcal{CTR}$ accrues significant benefits in the area of planning.

1 Introduction

The classical problem of automated planning has been used in a wide range of applications such as robotics, multi-agent systems, and more. Due to this wide range of applications, automated planning has become one of the most important research areas in Artificial Intelligence (AI). The history of using logical deduction to solve classical planning problems in AI dates back to the late 1960s when situation calculus was applied in the planning domain [18]. There are several planners that encode planning problems into satisfiability problems [21], constraint satisfaction problems (CSP) [24][14][1], or answer set programming [22][17][16][23] and use logical deduction to solve the planning problems. Beside those planners, a number of deductive planning frameworks have been proposed over the years [5][6][7][19][20][13]

There are several reasons that make logical deduction suitable to be used by a classical planner: (1) Logic-based deduction used in planning can be cast as a formal framework that eases proving different planning properties such as completeness. (2) Logic-based systems naturally provide a declarative language that simplifies the specification of planning problems. (3) Logical deduction is usually an essential component of intelligent and knowledge representation systems. Therefore, applying logical deduction in classical planning makes the integration of planners with such systems simpler. Despite

* Extended Abstract as part of the program of the 1st Joint ADT/LPNMR 2015 Doctoral Consortium co-chaired by Esra Erdem and Nicholas Mattei.

these benefits of using logical deduction in planning, many of the above mentioned deductive planning techniques are not getting as much attention as algorithms specifically provided for planning problems. The most obvious reason for this shortcoming is as follows: These works generally show how they can represent and encode classical planning actions and rely on a theorem prover of some sort to find plans. Therefore, the planning techniques embedded in such planners are typically the simplest state space planning (e.g. forward state space search) that has a extremely large search space. Consequently, they cannot exploit planning heuristics and techniques.

In this thesis, we will show that *Concurrent Transaction Logic* (or $\mathcal{CTR}$) [12, 11, 10] addresses this issue and also provides multiple advantages for specifying, generalizing, and solving planning problems. To illustrate the point, we will take *STRIPS* planning technique and show that its associated planning algorithm easily and naturally lend itself to compact representation in $\mathcal{CTR}$.

The expressiveness of $\mathcal{CTR}$ let the *STRIPS* planning algorithm be naturally extended with intensional rules. Concurrency in $\mathcal{CTR}$ lets us introduce a non-linear *STRIPS* planning algorithm, which is also proven to be complete. After inspecting the logic rules that simulate the planning algorithm, we observe some heuristics that can be applied to reduce the search space. The elegant expression of *STRIPS* algorithm in $\mathcal{CTR}$ lets us first characterize the regression of literals through *STRIPS* actions, then enhance the proposed *STRIPS* algorithm with a regression analysis method. We also extend the *STRIPS* planning algorithm with a few extra rules to solve planning problems with negative derived literals.

This extended abstract paper is organized as follows. Section 2 briefly explains how we formally encode planning techniques in $\mathcal{CTR}$. Section 3 also provides the results of our simple experiments to illustrate the practical applications of this method. The last section concludes our paper.

2 Planning using $\mathcal{CTR}$

We assume denumerable pairwise disjoint sets of variables $\mathcal{V}$, constants $\mathcal{C}$, extensional predicate symbols $\mathcal{P}_{ext}$, and intensional predicate symbols $\mathcal{P}_{int}$. Table 2 shows the syntax of our language. Our work is based on our formal definitions of the basics of *STRIPS* planning that can be found in [4].[1] Here, we just briefly remind basics of planning to the reader. A *STRIPS* **action** is a triple of the form $\alpha = \langle p_\alpha(X_1, ..., X_n), Pre_\alpha, E_\alpha \rangle$, where $p_\alpha(X_1, ..., X_n)$ denotes the action and, Pre_α and E_α are sets of literals representing the precondition and effects of α. A **planning problem** $\langle \mathbb{R}, \mathbb{A}, G, \mathbf{S} \rangle$ consists of a set of rules $\mathbb{R}$, a set of *STRIPS* actions $\mathbb{A}$, a set of literals G, called the **goal** of the planning problem, and an **initial state** $\mathbf{S}$. A sequence of actions $\sigma = \alpha_1, \ldots, \alpha_n$ is a **planning solution** (or simply a **plan**) for the planning problem if there is a sequence of states $\mathbf{S}_0, \mathbf{S}_1, \ldots, \mathbf{S}_n$ such that

- $\mathbf{S} = \mathbf{S}_0$ and $G \subseteq \mathbf{S}_n$ (i.e., G is satisfied in the final state);

[1] To understand this report, the reader is expected to be familiar with $\mathcal{CTR}$. We provide a brief introduction to the relevant *subset* of $\mathcal{CTR}$ in [4] that is needed for the understanding of this paper. More explanation about $\mathcal{CTR}$ can be found in [9], [11], [12], [8], and [10].

Term	$t := V \mid c$	where $V \in \mathcal{V}, c \in \mathcal{C}$
Atom	$P_\tau := p(t_1, \ldots, t_k)$	where $p \in \mathcal{P}_\tau, \tau \in \{ext, int\}$
Literal	$L := P_{int} \mid P_{ext} \mid \neg P_{ext}$	
Rule	$R := P_{int} \leftarrow L_1 \wedge \cdots \wedge L_m$	where $m \geq 0$

Table 1. The syntax of the language for representing *STRIPS* planning problems.

– for each $0 < i \leq n$, α_i is executable in state $\mathbf{S}_{i-1}$ and the result of that execution (for some substitution) is the state $\mathbf{S}_i$. [2]

As mentioned in Section 1, we define a set of $\mathcal{TR}$ clauses that simulate the well-known *STRIPS* planning algorithm and extend this algorithm to handle intentional predicates and rules. In essence, these rules are a natural (and much more concise and general) verbalization of the classical *STRIPS* algorithm [15]. However, unlike the original *STRIPS*, these rules constitute a *complete* planner when evaluated with the $\mathcal{CTR}$ proof theory. Using this encoding of *STRIPS* planning algorithm, we have extended *STRIPS* algorithm via extensions of our encoding with different respects. Our encoding is defined as follows.

Definition 1 **($\mathcal{TR}$ planning rules).** *Let $\Pi = \langle \mathbb{R}, \mathbb{A}, G, \mathbf{S} \rangle$ be a STRIPS planning problem. We define a set of $\mathcal{TR}$ rules, $\mathbb{P}(\Pi)$, which provides a sound and complete solution to the STRIPS planning problem. $\mathbb{P}(\Pi)$ has three disjoint parts, $\mathbb{P}_\mathbb{R}$, $\mathbb{P}_\mathbb{A}$, and $\mathbb{P}_G$, that are briefly described below. The details of this definition can be found in [4].*

– *The $\mathbb{P}_\mathbb{R}$ part is an extension to the classical STRIPS planning algorithm and is intended to capture intentional predicates and ramification of actions.*
– *The part $\mathbb{P}_\mathbb{A} = \mathbb{P}_{actions} \cup \mathbb{P}_{atoms} \cup \mathbb{P}_{achieves}$ is constructed out of the actions in $\mathbb{A}$. $\mathbb{P}_{actions}$ are the $\mathcal{CTR}$ rules that maps actions in $\mathbb{A}$ to $\mathcal{CTR}$ transactions. $\mathbb{P}_{atoms} = \mathbb{P}_{achieved} \cup \mathbb{P}_{enforced}$ has two disjoint parts. $\mathbb{P}_{achieved}$ are $\mathcal{CTR}$ rules that say if an extensional literal is true in a state then that literal has already been achieved as a goal. $\mathcal{CTR}$ rules in $\mathbb{P}_{enforced}$ say that one way to achieve a goal that occurs in the effects of an action is to execute that action. $\mathbb{P}_{achieves}$ is a set of $\mathcal{CTR}$ rules saying that to execute an action, one must first achieve the precondition of the action and then perform the state changes prescribed by the action.*
– *The $\mathbb{P}_G$ part is showing how the goal will be achieved.*

□

Given a set $\mathbb{R}$ of rules, a set $\mathbb{A}$ of *STRIPS* actions, an initial state $\mathbf{S}$, and a goal G, Definition 1 gives a set of $\mathcal{TR}$ rules that specify a planning strategy for that problem. To find a solution for that planning problem, one simply needs to place the request

$$? - achieve_G\,. \tag{1}$$

at a desired initial state and use the $\mathcal{TR}$'s inference system to find a proof. As mentioned before, a solution plan for a *STRIPS* planning problem is a sequence of actions leading

[2] In this case we will also say that $\mathbf{S}_0, \mathbf{S}_1, \ldots, \mathbf{S}_n$ is an execution of σ.

to a state that satisfies the planning goal. Such a sequence can be extracted by picking out the atoms of the form p_α from a successful derivation branch generated by the $\mathcal{TR}$ inference system. We provide a technique to extract that sequence of actions, called ***pivoting sequence of actions***. Soundness of a planning strategy means that, for any *STRIPS* planning problem, if sequence of actions is extracted as a solution, then that sequence of actions is a *solution* to the planning problem. We also prove that the proposed technique is sound. That is, the pivoting sequence of actions is a solution to the planning problem. ***Completeness*** of a planning strategy means that, for any *STRIPS* planning problem, if there is a solution, the planner will find *at least one* plan. A stronger statement about completeness is called ***comprehensive completeness***: if there is a *non-redundant* solution for a *STRIPS* planning problem, the planner will find *exactly that* plan. We also prove that the proposed $\mathcal{TR}$-based planner is comprehensively complete.

We also introduce *fSTRIPS* — a modification of the previously introduced *STRIPS* transform, which represents to a new planning strategy, which we call *fast STRIPS*. We show that although the new strategy explores a smaller search space, it is still sound and complete. Our experiments show that *fSTRIPS* can be orders of magnitude faster than *STRIPS*. The details of *fSTRIPS* can be found in [4].

The third part of our work shows how the simplicity embedded in $\mathcal{TR}$ encoding of planning strategies let one apply heuristics in planning mechanism. As an indirect result of our research, in this stage, we have shown that sophisticated planning heuristics, such as regression analysis, can be naturally represented in $\mathcal{TR}$ and that such representation can be used to express complex planning strategies such as *RSTRIPS* [3]. The simplicity of $\mathcal{TR}$ representation of *RSTRIPS* let us prove its completeness for the first time. We have also extended our above mentioned non-linear $\mathcal{TR}$-based *STRIPS* planner with regression analysis. The $\mathcal{TR}$ representation of those planning strategies rely on the concept of regression of a *STRIPS* action that is explained in [2]. The idea behind planning with regression is that the already achieved goals should be protected so that subsequent actions of the planner would not *unachieve* those goals. The details of these techniques can be found in [3].

In the last stage of our dissertation research, we have extended out above mentioned $\mathcal{TR}$-based *STRIPS* planner with respect to negative derived atoms. Our original domain specification language syntax reflected in Table 2 did not include negative derived atoms as none of the existing planners were able to solve planning problems with negative derived atoms. We also extend both the planning domain specification language and our above mentioned planner to solve planning problems with negative derived atoms. The idea behind this planner is that, in order to make a derived atom false, the planner disables every derivation can lead to make that atom true. The details of this extension along with the proofs of its soundness and completeness will be published in our under-preparation journal paper.

3 Experiments

In this section we first briefly report on our experiment. The first set of experiment compares *STRIPS* planning algorithm and forward state space search [2] that show that *STRIPS* can be faster than forward state space search up to three orders of magnitude.

Our results in [4] also compare *STRIPS* and *fSTRIPS* and show that *fSTRIPS* can be two orders of magnitude faster than *STRIPS*. We compared *RSTRIPS* and *fSTRIPS* to demonstrate our proposed regression analysis mechanism can improve the performance of planning up to three orders of magnitude [3]. Due to space limitations we skip explanations of the test environment, test cases, and the complete result tables and the interested readers are referred to our previous papers [2], [4], and [3]. We have also explained our PDDL2TR translator that maps planning problems to $\mathcal{TR}$ rules in [2].

4 Conclusion

This dissertation will demonstrate that the use of $\mathcal{CTR}$ and $\mathcal{TR}$ accrues significant benefits in the area of planning. As an illustration, we have shown that sophisticated planning strategies, such as *STRIPS*, not only can be naturally represented in $\mathcal{TR}$, but also such representation can be used to design new planning strategies such as non-linear *STRIPS*, *fSTRIPS*, non-linear *STRIPS* with regression analysis, *RSTRIPS*, and extensions with negative derived atoms. There are several promising directions to continue this work. One is to investigate other planning strategies and, hopefully, accrue similar benefits. We are also working to represent $\mathcal{G}raphPlan$ planning algorithm in $\mathcal{TR}$ to get same benefits.

Acknowledgments. This work was supported, in part, by the NSF grant 0964196. I also thank Prof. Michael Kifer for his great advises on my PhD research.

References

1. Barták, R., Toropila, D.: Solving sequential planning problems via constraint satisfaction. Fundam. Inform. 99(2), 125–145 (2010), http://dx.doi.org/10.3233/FI-2010-242
2. Basseda, R., Kifer, M.: State space planning using transaction logic. In: Pontelli, E., Tran, S.C. (eds.) Practical Aspects of Declarative Languages - 17th International Symposium, PADL 2015, Portland, OR, USA, June 18-19, 2015. Proceedings. Lecture Notes in Computer Science, Springer (2014)
3. Basseda, R., Kifer, M.: Planning with regression analysis in transaction logic. In: Cate, B., Mileo, A. (eds.) Web Reasoning and Rule Systems - 8th International Conference, RR 2015, Berlin, Germany, August 4-5, 2015. Proceedings. Lecture Notes in Computer Science, Springer (2015)
4. Basseda, R., Kifer, M., Bonner, A.J.: Planning with transaction logic. In: Kontchakov, R., Mugnier, M. (eds.) Web Reasoning and Rule Systems - 8th International Conference, RR 2014, Athens, Greece, September 15-17, 2014. Proceedings. Lecture Notes in Computer Science, vol. 8741, pp. 29–44. Springer (2014)
5. Bibel, W.: A deductive solution for plan generation. New Generation Computing 4(2), 115–132 (1986)
6. Bibel, W.: A deductive solution for plan generation. In: Schmidt, J.W., Thanos, C. (eds.) Foundations of Knowledge Base Management, pp. 453–473. Topics in Information Systems, Springer Berlin Heidelberg (1989)
7. Bibel, W., del Cerro, L.F., Fronhfer, B., Herzig, A.: Plan generation by linear proofs: On semantics. In: Metzing, D. (ed.) GWAI-89 13th German Workshop on Artificial Intelligence, Informatik-Fachberichte, vol. 216, pp. 49–62. Springer Berlin Heidelberg (1989)

8. Bonner, A., Kifer, M.: Transaction logic programming. In: Int'l Conference on Logic Programming. pp. 257–282. MIT Press, Budapest, Hungary (June 1993)
9. Bonner, A., Kifer, M.: Transaction logic programming (or a logic of declarative and procedural knowledge). Tech. Rep. CSRI-323, University of Toronto (November 1995), http://www.cs.toronto.edu/~bonner/transaction-logic.html
10. Bonner, A., Kifer, M.: Concurrency and communication in transaction logic. In: Joint Int'l Conference and Symposium on Logic Programming. pp. 142–156. MIT Press, Bonn, Germany (September 1996)
11. Bonner, A., Kifer, M.: A logic for programming database transactions. In: Chomicki, J., Saake, G. (eds.) Logics for Databases and Information Systems, chap. 5, pp. 117–166. Kluwer Academic Publishers (March 1998)
12. Bonner, A.J., Kifer, M.: An overview of transaction logic. Theoretical Computer Science 133 (1994)
13. Cresswell, S., Smaill, A., Richardson, J.: Deductive synthesis of recursive plans in linear logic. In: Biundo, S., Fox, M. (eds.) Recent Advances in AI Planning, Lecture Notes in Computer Science, vol. 1809, pp. 252–264. Springer Berlin Heidelberg (2000)
14. Erol, K., Hendler, J.A., Nau, D.S.: UMCP: A sound and complete procedure for hierarchical task-network planning. In: Hammond, K.J. (ed.) Proceedings of the Second International Conference on Artificial Intelligence Planning Systems, University of Chicago, Chicago, Illinois, USA, June 13-15, 1994. pp. 249–254. AAAI (1994), http://www.aaai.org/Library/AIPS/1994/aips94-042.php
15. Fikes, R.E., Nilsson, N.J.: STRIPS: A new approach to the application of theorem proving to problem solving. Artificial Intelligence 2(34), 189 – 208 (1971)
16. Gebser, M., Kaufmann, R., Schaub, T.: Correct reasoning. chap. Gearing Up for Effective ASP Planning, pp. 296–310. Springer-Verlag, Berlin, Heidelberg (2012), http://dl.acm.org/citation.cfm?id=2363344.2363364
17. Gelfond, M., Lifschitz, V.: Action languages. Electron. Trans. Artif. Intell. 2, 193–210 (1998), http://www.ep.liu.se/ej/etai/1998/007/
18. Green, C.: Application of theorem proving to problem solving. In: Proceedings of the 1st International Joint Conference on Artificial Intelligence. pp. 219–239. IJCAI'69, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1969), http://dl.acm.org/citation.cfm?id=1624562.1624585
19. Hölldobler, S., Schneeberger, J.: A new deductive approach to planning. New Generation Computing 8(3), 225–244 (1990)
20. Kahramanogullari, O.: On linear logic planning and concurrency. Information and Computation 207(11), 1229 – 1258 (2009), special Issue: 2nd International Conference on Language and Automata Theory and Applications (LATA 2008)
21. Kautz, H., Selman, B.: Planning as satisfiability. In: Proceedings of the 10th European Conference on Artificial Intelligence. pp. 359–363. ECAI '92, John Wiley & Sons, Inc., New York, NY, USA (1992), http://dl.acm.org/citation.cfm?id=145448.146725
22. Lifschitz, V.: Answer set programming and plan generation. Artificial Intelligence 138(12), 39 – 54 (2002), http://www.sciencedirect.com/science/article/pii/S0004370202001868, knowledge Representation and Logic Programming
23. Son, T.C., Baral, C., Tran, N., Mcilraith, S.: Domain-dependent knowledge in answer set planning. ACM Trans. Comput. Logic 7(4), 613–657 (Oct 2006), http://doi.acm.org/10.1145/1183278.1183279
24. Stefik, M.J.: Planning with Constraints. Ph.D. thesis, Stanford, CA, USA (1980), aAI8016868