Formalizing Goal Serializability for Evaluation of Planning Features

Reza Basseda(✉) and Michael Kifer

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

Abstract. Evaluation of the properties of various planning techniques such as completeness and termination plays an important role in choosing an appropriate planning technique for a particular planning problem. In this paper, we use the already existing formal specification of two well-known and classic state space planning techniques, forward state space planning and goal stack state space planning techniques, in Transaction Logic($\mathcal{TR}$) to study their completeness. Our study shows that using $\mathcal{TR}$, we can formally specify the serializability of planning problems and prove the completeness of *STRIPS* planning problems for planning problems with serializable goals.

Keywords: Deductive planning · STRIPS planning · Transaction Logic

1 Introduction

Evaluation of different properties of planning techniques such as termination and completeness becomes more essential when, in many cases, different search strategies of a planning technique may affect the performance of planning process while different properties such as completeness and termination is required. For example, forward state space planning and goal stack space planning techniques may have different execution time when they are used by a robot in the famous block world example while Sussman Anomaly [1] shows that goal stack space planning is not guaranteed to succeed. Such considerations need appropriate formal frameworks to represent the planning techniques properly.

Using logical deduction to solve a planning problem has been a popular approach for more than three decades [2–6]. However, none of the existing logical frameworks of deductive planners are used to study the different properties of planning techniques because these planners are just relying on their corresponding inference systems to search for a plan and they are not expressive enough to formally represent more complicated planning techniques such as goal state space planning. Therefore, the requirement of a formal, neat, and expressive logical framework for such studies seems to be inevitable.

This work was supported, in part, by the NSF grant 0964196.

L. Michael and A. Kakas (Eds.): JELIA 2016, LNAI 10021, pp. 515–521, 2016.
DOI: 10.1007/978-3-319-48758-8_33

In this paper, we are using the encoding of forward state space planning and goal stack state space planning techniques in Transaction Logic($\mathcal{TR}$) for the evaluation of their completeness. Our paper shows that $\mathcal{TR}$ is an appropriate logical framework for such evaluation. Unlike above mentioned logical frameworks, the well-defined model theory of $\mathcal{TR}$ together with its sound and complete proof theory let us easily prove different properties of planning techniques such as completeness. It also lets us formally redefine the concept of goal serializablity in planning that is used to examine the completeness of goal stack state space planning technique for planning problems. Our simple and straightforward proofs for two classic planning techniques, forward state space planning (called naïve[1]) and goal stack state space (called *STRIPS*[2]) are evidences for this claim.

The next section briefly characterizes a planning problem. The third section explains how we formally encode planning techniques in $\mathcal{TR}$ and the last section concludes our paper.

2 Characterization of a Planning Problem

In a *STRIPS* planning problem, actions update the state of a system. We assume denumerable sets of variables $\mathcal{X}$, constants $\mathcal{C}$, and disjoint sets of predicate symbols, extensional ($\mathcal{P}_{ext}$) and intensional ($\mathcal{P}_{int}$) ones. A *term* is a variable or constant. Extensional (resp. intensional) ***Atoms*** have the form $p(t_1, ..., t_n)$, where t_i is a term and $p \in \mathcal{P}_{ext}$ (resp. $p \in \mathcal{P}_{int}$). A ***ground*** atom is a variable free atom. A ***literal*** is either an atom or a negated extensional atom, $\neg p(t_1, ..., t_n)$. Note that negative intensional atoms cannot form literals. A substitution θ is a set of expressions of the form $X \longleftarrow c$, where $X \in \mathcal{X}$ and $c \in \mathcal{C}$. Given a substitution θ, an atom $a\theta$ is obtained from atom a by replacing its variables with constants according to θ.

Intensional predicate symbols are defined by *rules*. A rule r, shown as $head(r) \leftarrow b_1 \wedge \cdots \wedge b_n$, consists of an intensional atom $head(r)$ in the head and a conditional body, a (possibly empty) conjunction of literals $b_1, \ldots, b_n$, where $b_i \in body(r)$. A ***ground instance*** of a rule, $r\theta$, is any rule obtained from r by a substitution of $head(r)$ and $body(r)$ with ground atoms $head(r)\theta$ and $body(r)\theta$ respectively. Given a set of literals $\mathbf{S}$ and a ground rule $r\theta$, the rule is *true* in $\mathbf{S}$ if either $head(r)\theta \in \mathbf{S}$ or $body(r)\theta \not\subseteq \mathbf{S}$. A (possibly non-ground) rule is *true* in $\mathbf{S}$ if all of its ground instances are true in $\mathbf{S}$. A set $\mathbf{S}$ of literals is ***consistent*** if there is no atom, a, such that $\{a, \neg a\} \subseteq \mathbf{S}$.

Definition 1 (State). *Given a set of rules $\mathbb{R}$, a consistent set $\mathbf{S}$ of literals is called a* **state** *if and only if*

1. *For each ground extensional atom a, either, $a \in \mathbf{S}$, or $\neg a \in \mathbf{S}$.*
2. *Every rule of $\mathbb{R}$ is true in $\mathbf{S}$.*

[1] Due to its simple nature.
[2] As it was originally proposed by [7] in *STRIPS*.

Definition 2 (*STRIPS* action). *A STRIPS* **action** $\alpha = \langle p_\alpha(X_1, ..., X_n), Pre(\alpha), E(\alpha)\rangle$ *consists of an intensional atom* $p_\alpha(X_1, ..., X_n)$ *in which* $p_\alpha \in \mathcal{P}_{int}$ *is a predicate that is reserved to represent the action* α *and can be used for no other purpose, a set of literals* $Pre(\alpha)$*, called the* ***precondition*** *of* α*, and a consistent set of extensional literals* $E(\alpha)$*, called the* ***effect*** *of* α*. The variables in* $Pre(\alpha)$ *and* $E(\alpha)$ *must occur in* $\{X_1, ..., X_n\}$*.*

Note that the literals in $Pre(\alpha)$ can be both extensional and intensional, while the literals in $E(\alpha)$ can be extensional only.

Definition 3 (Execution of a *STRIPS* action). *A STRIPS action* α *is* ***executable*** *in a state* $\mathbf{S}$ *if there is a substitution* θ *such that* $\theta(Pre(\alpha)) \subseteq \mathbf{S}$*. A* ***result of the execution*** *(with respect to* θ*) is the state* $\mathbf{S}'$ *such that* $\mathbf{S}' = (\mathbf{S} \setminus \neg\theta(E(\alpha))) \cup \theta(E(\alpha))$*, where* $\neg E = \{\neg\ell | \ell \in E\}$*.*

Note that $\mathbf{S}$ is well-defined since $E(\alpha)$ is consistent. Observe also that, if α has variables, the result of an execution, $\mathbf{S}$, may depend on the chosen substitution θ.

Definition 4 (Planning problem). *Given a set of rules* $\mathbb{R}$*, a set of STRIPS actions* $\mathbb{A}$*, a set of literals* G*, called the* ***goal****, and an* ***initial state*** $\mathbf{S}$*, a* **planning solution** *(or simply a* ***plan****) for the planning* $\Pi = \langle \mathbb{R}, \mathbb{A}, G, \mathbf{S}\rangle$ *is a sequence of ground actions* $\sigma = \alpha_1, \ldots, \alpha_n$ *such that for each* $1 \leq i \leq n$*;*

- *there is a substitution* θ_i *and a STRIPS action* $\alpha'_i \in \mathbb{A}$ *such that* $\alpha'_i\theta = \alpha_i$*; and*
- *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);*
  - α_i *is executable in state* $\mathbf{S}_{i-1}$ *and the result of that execution is the state* $\mathbf{S}_i$*.*

The following definition of ***goal serializable planning*** problems constitutes a measure that recognizing planning problems, for which the *STRIPS* planning technique is proven to be complete.

Definition 5 (Goal serializable planning problem). *Given a planning problem* $\Pi = \langle \mathbb{R}, \mathbb{A}, \mathbf{S}, G\rangle$*, let* σ *be the shortest solution plan for* Π *and* $G' \subset G$ *be any arbitrary set of literals such that* $G \neq G'$*. We call* Π *a* **goal serializable planning problem** *if and only if, for every* σ'*, that is a planning solution for* $\Pi' = \langle \mathbb{R}, \mathbb{A}, G', \mathbf{S}\rangle$ *and the result of its execution is* $\mathbf{S}'$ *where* $|\sigma'| \leq |\sigma|$*, there is a planning solution* σ'' *for* $\Pi'' = \langle \mathbb{R}, \mathbb{A}, G, \mathbf{S}'\rangle$ *such that* $|\sigma''| < |\sigma|$*.*

A brief introduction to the *subset* of $\mathcal{TR}$ [8–11] has been appeared in [12–14]. Although such introduction is required to make our paper self-contained, we omit that introduction to save space and refer the reader to [14].

3 The $\mathcal{TR}$ Planners

The informal idea of using $\mathcal{TR}$ as a planning formalism and an encoding of *STRIPS* and naive planning as a set of $\mathcal{TR}$ rules first appeared in an unpublished

report [8]. We extend and slightly modify the original methods to prove different properties of each planning technique.

Given a set of extensional literals G, we define $Enf(G)$ to be the set of elementary updates that makes G true. Next we introduce a natural correspondence between *STRIPS* actions and $\mathcal{TR}$ rules.

Definition 6 (Actions as $\mathcal{TR}$ rules). *Let $\alpha = \langle p_\alpha(\overline{X}), Pre(\alpha), E(\alpha)\rangle$ be a STRIPS action. We define its* ***corresponding*** *TR* ***rule****, $tr(\alpha)$, to be a rule of the form*

$$p_\alpha(\overline{X}) \leftarrow (\wedge_{\ell \in Pre(\alpha)} \ell) \ \otimes \ (\otimes_{u \in Enf(E(\alpha))} u). \tag{1}$$

Note that in (1) the actual order of action execution in the last component, $\otimes_{u \in Enf(E(\alpha))} u$, is immaterial, since all such executions happen to lead to the same state.

We now define a set of $\mathcal{TR}$ clauses that simulate naive and *STRIPS* [7] planning techniques. Moreover, for convenience, we use $a \mathbin{\widehat{\otimes}} b$ as a shorthand for $a \otimes b \vee b \otimes a$. This connective is called the *shuffle* operator in [8]. We define it to be commutative and associative and thus extend it to arbitrary number of operands.

Definition 7 (Naïve planning rules). *Given a STRIPS planning problem $\Pi = \langle \mathbb{R}, \mathbb{A}, G, \mathbf{S}\rangle$ (see Definition 4), we define a set of $\mathcal{TR}$ rules, $\mathbb{P}(\Pi)$, which simulate naive planning technique to provide a planning solution to the planning problem. $\mathbb{P}(\Pi)$ has two parts, $\mathbb{P}_{general}$, $\mathbb{P}_{\mathbb{A}}$, described below.*

- *The $\mathbb{P}_{general}$ part: contains a couple of rules as follows;*

$$\begin{array}{l} plan \leftarrow . \\ plan \leftarrow execute\_action \otimes plan. \end{array} \tag{2}$$

  These rules construct a sequence of actions and bind them to the plan.
- *The $\mathbb{P}_{actions}$ part: for each $\alpha \in \mathbb{A}$, $\mathbb{P}_{actions}$ has a couple of rules as follows;*

$$\begin{array}{l} p_\alpha(\overline{X}) \leftarrow (\wedge_{\ell \in Pre(\alpha)} \ell) \ \otimes \ (\otimes_{u \in Enf(E(\alpha))} u). \\ execute\_action \leftarrow p_\alpha(\overline{X}). \end{array} \tag{3}$$

  This is the $\mathcal{TR}$ rule that corresponds to the action α, introduced in Definition 6 and generally links an action to a plan.

Definition 8 (*STRIPS* planning rules). *Let $\Pi = \langle \mathbb{R}, \mathbb{A}, G, \mathbf{S}\rangle$ be a STRIPS planning problem (see Definition 4). We define a set of $\mathcal{TR}$ rules, $\mathbb{P}(\Pi)$, which simulate STRIPS planning technique to provide a planning solution to the planning problem. $\mathbb{P}(\Pi)$ has three disjoint parts, $\mathbb{P}_{\mathbb{R}}$, $\mathbb{P}_{\mathbb{A}}$, and $\mathbb{P}_G$, described below.*

- *The $\mathbb{P}_{\mathbb{R}}$ part: for each rule $p(\overline{X}) \leftarrow p_1(\overline{X}_1) \wedge \cdots \wedge p_k(\overline{X}_n)$ in $\mathbb{R}$, $\mathbb{P}_{\mathbb{R}}$ has a rule of the form*

$$achieve\_p(\overline{X}) \leftarrow \widehat{\otimes}_{i=1}^{n} achieve\_p_i(\overline{X}_i). \tag{4}$$

  Rule (4) is an extension to the classical STRIPS planning algorithm. It captures intentional predicates and ramification of actions, and it is the only major aspect of our $\mathcal{TR}$-based rendering of STRIPS that was not present in the original in one way or another.

– *The part* $\mathbb{P}_{\mathbb{A}} = \mathbb{P}_{actions} \cup \mathbb{P}_{atoms} \cup \mathbb{P}_{achieves}$ *is constructed out of the actions in* $\mathbb{A}$ *as follows:*
  - $\mathbb{P}_{actions}$*: similar to Definition 7.*
  - $\mathbb{P}_{atoms} = \mathbb{P}_{achieved} \cup \mathbb{P}_{enforced}$ *has two disjoint parts as follows:*
    – $\mathbb{P}_{achieved}$*: for each extensional predicate* $p \in \mathcal{P}_{ext}$*,* $\mathbb{P}_{achieved}$ *has the rules*

$$\begin{array}{l} achieve\_p(\overline{X}) \leftarrow p(\overline{X}). \\ achieve\_not\_p(\overline{X}) \leftarrow \neg p(\overline{X}). \end{array} \tag{5}$$

    These rules say that if an extensional literal is true in a state then that literal has already been achieved as a goal.
    – $\mathbb{P}_{enforced}$*: for each action* $\alpha = \langle p_\alpha(\overline{X}), Pre(\alpha), E(\alpha)\rangle$ *in* $\mathbb{A}$ *and each* $e(\overline{Y}) \in E(\alpha)$*,* $\mathbb{P}_{enforced}$ *has the following rule:*

$$achieve\_e(\overline{Y}) \leftarrow \neg e(\overline{Y}) \otimes execute\_p_\alpha(\overline{X}). \tag{6}$$

    This rule says that one way to achieve a goal that occurs in the effects of an action is to execute that action.
  - $\mathbb{P}_{achieves}$*: for each action* $\alpha = \langle p_\alpha(\overline{X}), Pre(\alpha), E(\alpha)\rangle$ *in* $\mathbb{A}$*,* $\mathbb{P}_{achieves}$ *has the following rule:*

$$execute\_p_\alpha(\overline{X}) \leftarrow (\widehat{\otimes}_{\ell \in Pre(\alpha)} achieve\_\ell) \otimes p_\alpha(\overline{X}). \tag{7}$$

    This means that to execute an action, one must first achieve the precondition of the action and then perform the state changes prescribed by the action.
– $\mathbb{P}_G$*: Let* $G = \{g_1, ..., g_k\}$*. Then* $\mathbb{P}_G$ *has a rule of the form:*

$$achieve_G \leftarrow (\widehat{\otimes}_{g_i=1}^{k} achieve\_g_i) \otimes (\wedge_{i=1}^{k} g_i). \tag{8}$$

Given a *STRIPS* planning problem $\mathit{\Pi} = \langle \mathbb{R}, \mathbb{A}, G, \mathbf{S}\rangle$, each of Definitions 7 and 8 gives a set of $\mathcal{TR}$ rules that specifies the corresponding planning strategy for that problem. To find a solution for that planning problem, one simply needs to place the request (9) (resp. (10)) in the initial state and use the set of rules from Definition 7 (resp. Definition 8) and the $\mathcal{TR}$'s inference system to find a proof.

$$? - plan \otimes (\wedge_{g_i \in G} g_i). \tag{9}$$

$$? - achieve_G\,. \tag{10}$$

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.

Theorem 1 (Completeness of naive planning). *If there is a plan that achieves the goal* G *from the initial state* $\mathbf{D}_0$ *then the* $\mathcal{TR}$*-based naive planner will find a plan.*

Proof (Sketch). The proof is a direct consequence of $\mathcal{TR}$ inference system completeness.

Theorem 2 (Completeness of *STRIPS* planning). *Given a goal serializable planning problem $\Pi = \langle \mathbb{R}, \mathbb{A}, G, \mathbf{D}_0 \rangle$, if there is a plan that achieves the goal G from the initial state $\mathbf{D}_0$ then the $\mathcal{TR}$-based STRIPS planner will find a plan.*

Proof (Sketch). By induction on the length of the plan. The full proof can be found in the full report.[3]

4 Conclusion

This paper has demonstrated that the use of Transaction Logic opens up new possibilities for generalizations and considerations of the properties of existing planning techniques. For instance, we have shown that once the *STRIPS* algorithm is cast as a set of rules in $\mathcal{TR}$, the different properties of the framework can be studied, almost for free, to recognize and define such advanced concepts as goal serializability of planning. The concept of serializability, not only classifies planning problem regarding to the completeness of *STRIPS* planning technique, but also establishes further explorations in different areas such as algorithms and graph theory.

References

1. Sacerdoti, E.D.: The nonlinear nature of plans. In: Proceedings of the 4th International Joint Conference on Artificial Intelligence, IJCAI 1975, vol. 1, pp. 206–214. Morgan Kaufmann Publishers Inc., San Francisco (1975)
2. Bibel, W.: A deductive solution for plan generation. In: Schmidt, J.W., Thanos, C. (eds.) Foundations of Knowledge Base Management. Topics in Information Systems, pp. 453–473. Springer, Heidelberg (1989)
3. Kahramanoğulları, O.: On linear logic planning and concurrency. In: Martín-Vide, C., Otto, F., Fernau, H. (eds.) LATA 2008. LNCS, vol. 5196, pp. 250–262. Springer, Heidelberg (2008). doi:10.1007/978-3-540-88282-4_24
4. Cresswell, S., Smaill, A., Richardson, J.: Deductive synthesis of recursive plans in linear logic. In: Biundo, S., Fox, M. (eds.) ECP 1999. LNCS (LNAI), vol. 1809, pp. 252–264. Springer, Heidelberg (2000). doi:10.1007/10720246_20
5. Guglielmi, A.: Concurrency and plan generation in a logic programming language with a sequential operator. In: Hentenryck, P.V. (ed.) ICLP, pp. 240–254. MIT Press (1994)
6. Kahramanogullari, O.: Towards planning as concurrency. In: Hamza, M.H. (ed.) Artificial Intelligence and Applications, pp. 387–393. IASTED/ACTA Press (2005)
7. Fikes, R.E., Nilsson, N.J.: STRIPS: a new approach to the application of theorem proving to problem solving. Artif. Intell. **2**, 189–208 (1971)
8. Bonner, A., Kifer, M.: Transaction logic programming (or a logic of declarative and procedural knowledge). Technical report CSRI-323, University of Toronto, November 1995. http://www.cs.toronto.edu/~bonner/transaction-logic.html

[3] http://ewl.cewit.stonybrook.edu/planning/Goal-Serializability.pdf.

9. Bonner, A., Kifer, M.: A logic for programming database transactions. In: Chomicki, J., Saake, G. (eds.) Logics for Databases and Information Systems, pp. 117–166. Kluwer Academic Publishers, March 1998
10. Bonner, A.J., Kifer, M.: An overview of transaction logic. Theoret. Comput. Sci. **133**(32), 205–265 (1994)
11. Bonner, A., Kifer, M.: Transaction logic programming. In: International Conference on Logic Programming, Budapest, Hungary, pp. 257–282. MIT Press, June 1993
12. Basseda, R., Kifer, M., Bonner, A.J.: Planning with transaction logic. In: Kontchakov, R., Mugnier, M.-L. (eds.) RR 2014. LNCS, vol. 8741, pp. 29–44. Springer, Heidelberg (2014). doi:10.1007/978-3-319-11113-1_3
13. Basseda, R., Kifer, M.: Planning with regression analysis in transaction logic. In: Cate, B., Mileo, A. (eds.) RR 2015. LNCS, vol. 9209, pp. 45–60. Springer, Heidelberg (2015). doi:10.1007/978-3-319-22002-4_5
14. Basseda, R., Kifer, M.: State space planning using transaction logic. In: Pontelli, E., Son, T.C. (eds.) PADL 2015. LNCS, vol. 9131, pp. 17–33. Springer, Heidelberg (2015). doi:10.1007/978-3-319-19686-2_2