Software Impacts 16 (2023) 100505

Contents lists available at ScienceDirect

Software Impacts

journal homepage: www.journals.elsevier.com/software-impacts

Original software publication

A tool for declarative Trace Alignment via automated planning

Giuseppe De Giacomo [a,b], Francesco Fuggitti [b,c,*], Fabrizio Maria Maggi [d], Andrea Marrella [b], Fabio Patrizi [b]

[a] University of Oxford, Oxford, UK
[b] Sapienza University, Rome, Italy
[c] York University, Toronto, Canada
[d] Free University of Bozen-Bolzano, Bolzano, Italy

ARTICLE INFO

Keywords:
Declarative Trace Alignment
Business process management
Linear-time temporal logics on finite traces
Automated planning

ABSTRACT

We present a tool, called TraceAligner, for solving Trace Alignment by first compiling into Planning and then solving it with any available cost-optimal planner. TraceAligner can produce different variants of the output Planning instance, each offering different degrees of readability and solution efficiency. The Planning instance is expressed in PDDL, the Planning Domain Definition Language. The tool can be easily extended and coupled with any planner taking PDDL as input language. A thorough experimental analysis has shown that the approach dramatically outperforms existing ad-hoc tools, thus making TraceAligner the best-performing tool for Trace Alignment with declarative specifications.

Code metadata

Current code version	v1.0
Permanent link to code/repository used for this code version	https://github.com/SoftwareImpacts/SIMPAC-2023-27
Permanent link to Reproducible Capsule	https://codeocean.com/capsule/3375070/tree/v1
Legal Code License	MIT
Code versioning system used	git
Software code languages, tools, and services used	Java, Python
Compilation requirements, operating environments & dependencies	Compilation: Java JDK 1.8, Gradle; Dependencies: Open XES, Lydia
If available Link to developer documentation/manual	–
Support email for questions	fuggitti@diag.uniroma1.it

1. Context and motivation

Business Process Management (BPM) is the research area concerned with discovering, modeling, analyzing, and managing business processes (BPs) to measure their productivity and improve their performance [1]. Usually, BPs are high-level processes involving automated and human-based activities such that, when executed, generate finite sequences of activities (or events) called *traces*, typically collected in a *log* (i.e., a set of traces). When activities require manual intervention, it is not uncommon for log traces to be inconsistent with the expected process behavior. For instance, an insurance claim process where a human operator is responsible for collecting all the documents related to the claim, checking the information they contain, and, if correct, starting the claim process is highly error-prone. Therefore, identifying and analyzing such traces to prevent errors is of paramount importance, and this is the main objective of what in BPM is known as *Trace Alignment* [2,3]. Existing works from Process Mining have witnessed that trace alignment is a highly-relevant problem with practical relevance to uncover common and frequent deviation patterns in several Computer Science domains.

An instance of trace alignment includes a log trace, a BP model, or *specification*, and a cost for each modification (insertion or deletion of activities) applicable to the input trace. A BP model defines the (possibly partial) execution order of the activities of interest and can be

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author at: Sapienza University, Rome, Italy.
E-mail addresses: degiacomo@diag.uniroma1.it (G. De Giacomo), fuggitti@diag.uniroma1.it (F. Fuggitti), maggi@inf.unibz.it (F.M. Maggi), marrella@diag.uniroma1.it (A. Marrella), patrizi@diag.uniroma1.it (F. Patrizi).

https://doi.org/10.1016/j.simpa.2023.100505
Received 23 January 2023; Received in revised form 14 March 2023; Accepted 18 April 2023

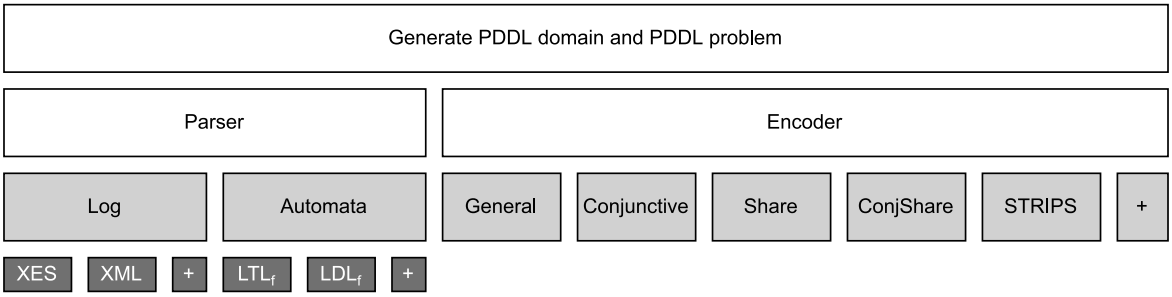

Fig. 1. The TraceAligner architecture.

specified either *procedurally* or *declaratively*. Here, we focus on declarative specifications expressed using formal languages such as Linear-time Temporal and Dynamic Logics on finite traces, i.e., LTL$_f$ or LDL$_f$ [4]. In such a setting, trace alignment is the problem of checking whether an actual trace related to a BP execution conforms to the expected process behavior and, if not, finding a *minimal* set of changes that *aligns* the trace to the process. Changes mainly consist in adding or deleting activities at some points of the trace, when necessary. To solve the trace alignment problem, existing approaches, e.g., [5,6], are based on ad-hoc implementations of the well-known A* search algorithm, which compromise scalability as the input complexity increases, namely with large specifications and long traces.

The theoretical solution, first presented in [7], solves the trace alignment problem by reducing it to deterministic cost-optimal planning [8]. A cost-optimal planning problem combines a domain with an initial state and a goal (usually compactly represented with the de-facto standard Planning Domain Definition Language – PDDL for short) and consists in looking for a cost-optimal sequence of actions that transforms the initial state into a desired goal state. For this problem, many automated planning techniques have been devised over the years, and many implemented solvers (often referred to as *planners*) are available today. Although planning is theoretically intractable in the worst-case, current approaches can typically solve large problem instances fast, regardless of their worst-case guarantees. We can thus exploit the efficiency, versatility, and customization of state-of-the-art automated planners to solve trace alignment problems effectively.

Based on this, we have devised a tool, called TraceAligner, which implements the theoretical solution and solves the target problem using any off-the-shelf cost-optimal planner available, such as Fast-Downward [9] or SymbA*-2 [10]. The tool is validated in an experimental analysis, whose results show that our approach dramatically outperforms the existing ad-hoc techniques included in the competitor toolkit ProM (https://promtools.org).

2. Software description

TraceAligner is a Java tool that takes as input both a BP log collected using either the XML format or the XES format and a set of LTL$_f$/LDL$_f$ declarative models, and produces a set of cost-optimal planning instances expressed in PDDL. Every planning problem instance corresponds to a specific log trace, while a cost-optimal solution to such instance corresponds to an optimal alignment of the input log trace, wrt to the input set of specifications. Once TraceAligner has generated the problem instances corresponding to the log traces, one can use any off-the-shelf cost-optimal planner, such as Fast-Downward [9] or SymbA*-2 [10] to solve the task and retrieve a proper alignment solution. TraceAligner comes as a Java library but also exposes a convenient command-line interface for direct interaction.

2.1. Architecture

The main feature of TraceAligner is a method that generates PDDL domains and PDDL problems. This method relies on two components, the *Parser* and the *Encoder*, as illustrated in Fig. 1.

Parser. The parsing component, as the name suggests, handles the parsing of input log traces (XES or XML format) and input model specifications represented as Deterministic Finite-state Automata (DFA). Input model specifications are given in LTL$_f$/LDL$_f$ formulas and are internally translated to DFAs through a call to the Lydia tool [11], which implements the state-of-the-art compositional approach to translate LTL$_f$/LDL$_f$ formulas into minimal DFAs. Although we already cover a good amount of commonly used input formats for both log traces and formal languages[1], the implementation is designed to ease the addition of new input formats, such as the Pure-Past LTL [13,14], which has recently been advocated for planning [14–16].

Encoder. The encoding component is perhaps the most important component of the package. In general, given the theoretical reduction of trace alignment to cost-optimal planning, there exist several different encoding schemata in PDDL. In fact, for a given planning problem, many possible semantically-equivalent formulations are possible, with each one featuring a different, possibly dramatic, impact on the solution performance. TraceAligner implements two classes of encoding variants, along with some possible optimizations. In the first class, there is a high-level general encoding and three variants. The Conjunctive variant has conjunctive formulas to encode planning goal states; in the Share variant, the number of fluents modeling automata states is reduced; and the ConjShare combines the two. On the other hand, the second class includes a low-level encoding that can be considered as the instantiated (or *grounded*) version of the General encoding. While the encodings of the first class have the advantage of increased readability and understandability at the expense of a slight performance loss, those in the second class gain performance and scalability at the expense of readability. Finally, TraceAligner tool has been developed with a special focus on extensibility. As shown in Fig. 1, TraceAligner can be augmented on any and all of its components, maintaining exactly the same main API to generate PDDL domain and PDDL problems.

3. Impact

The STRIPS encoding part of TraceAligner has been largely employed in [7], which reports on results of experiments conducted with several planners fed with combinations of real-life and synthetic event logs and processes. A clever PDDL encodings generation combined with the latest in automated planning techniques (i.e., planners and heuristics) makes the TraceAligner system the best-performing alignment tool available in the BPM research area when dealing with declarative specifications. In particular, results in [7] show that, when process models and event-log traces are of considerable size, the approach

[1] Note that, in the BPM area, BP declarative models are usually expressed using DECLARE patterns [12]. Here, instead, we allow LTL$_f$ and LDL$_f$, which are strictly more expressive formalisms.

implemented by TraceAligner outperforms the existing approach based on ad-hoc implementations of the A* algorithm, presented in [6], by several orders of magnitude.

From its inception in [7], the TraceAligner tool has contributed to opening up new research questions and bringing new perspectives for related research areas. In fact, there have been increasing synergies between AI and BPM witnessed by the continuous publication of several related works, e.g., [17–19]. Moreover, the theoretical elegance and the impressive performance have raised new research questions that are worth exploring. From a theoretical perspective on planning, developing *process-aware* planning heuristics could bring about several insights into the problem and significantly improve the planning search. On the other hand, the trace alignment problem can also be employed to identify and fix potential deviations in an AI agent's behavior. Indeed, traces can model agents' executions, and specifications can model properties that agent executions are expected to satisfy.

In general, the development of TraceAligner has raised awareness about viewing trace alignment as an interesting application of planning, thus demonstrating the power and generality of planning once again. It is worth noticing that parts of the reduction and encodings implemented in the work are applicable to any problem that includes temporal constraints expressible as finite-state automata. Additionally, the experimental study carried out has provided useful guidelines for efficient representations of such constraints as part of a planning domain. For instance, novel techniques that may benefit from these guidelines include applications to workflow construction in the context of Robotic Process Automation [20]. Finally, a number of initiatives are starting to be undertaken by both the AI and the BPM communities. Examples include tutorials, such as [21] at the BPM conference, the "AI4BPM" bridge workshop at AAAI 2023, and the "PMAI" workshop at IJCAI 2023.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been partially supported by the EU ERC Advanced Grant WhiteMech (No. 834228), the EU ICT-48 2020 project TAILOR (No. 952215), the PRIN project RIPER, Italy (No. 20203FFYLK), and the PNRR MUR project FAIR, Italy (No. PE0000013).

References

[1] M. Dumas, M. La Rosa, J. Mendling, H.A. Reijers, et al., Fundamentals of Business Process Management, vol. 2, Springer, 2018.
[2] A. Adriansyah, N. Sidorova, B.F. van Dongen, Cost-Based Fitness in Conformance Checking, in: ACSD 2011, IEEE, 2011.
[3] J. Carmona, B.F. van Dongen, A. Solti, M. Weidlich, Conformance Checking - Relating Processes and Models, Springer, 2018.
[4] G. De Giacomo, M.Y. Vardi, Linear Temporal Logic and Linear Dynamic Logic on Finite Traces, in: 23th Int. Conf. on AI (IJCAI'13), 2013.
[5] M. de Leoni, F.M. Maggi, W.M.P. van der Aalst, Aligning Event Logs and Declarative Process Models for Conformance Checking, in: 10th Int. Conf. on Business Process Management (BPM 2012), 2012.
[6] M. de Leoni, F.M. Maggi, W. van der Aalst, An alignment-based framework to check the conformance of declarative process models and to preprocess event-log data, Inf. Syst. 47 (2015) 258–277.
[7] G. De Giacomo, F. Maggi, A. Marrella, F. Patrizi, On the Disruptive Effectiveness of Automated Planning for LTL$_f$-Based Trace Alignment, in: AAAI, AAAI Press, 2017, pp. 3555–3561.
[8] H. Geffner, B. Bonet, A Concise Introduction to Models and Methods for Automated Planning, Synth.Lect. on AI and ML 8 (1) (2013).
[9] M. Helmert, The Fast Downward Planning System, J. Artif. Intell. Res.(JAIR) 26 (2006) 191–246.
[10] A. Torralba, V. Alcazar, D. Borrajo, P. Kissmann, S. Edelkamp, Symba: A symbolic bidirectional a planner, in: International Planning Competition, 2014, pp. 105–108.
[11] G. De Giacomo, M. Favorito, Compositional Approach to Translate LTLf/LDLf into Deterministic Finite Automata, in: ICAPS, AAAI Press, 2021, pp. 122–130.
[12] W. van der Aalst, M. Pesic, H. Schonenberg, Declarative Workflows: Balancing Between Flexibility and Support, Computer Science - R&D 23 (2) (2009) 99–113.
[13] O. Lichtenstein, A. Pnueli, L.D. Zuck, The Glory of the Past, in: Logic of Programs, in: LNCS, 193, Springer, 1985, pp. 196–218.
[14] G. De Giacomo, A. Di Stasio, F. Fuggitti, S. Rubin, Pure-Past Linear Temporal and Dynamic Logic on Finite Traces, in: IJCAI, ijcai.org, 2020, pp. 4959–4965.
[15] G. De Giacomo, M. Favorito, F. Fuggitti, Planning for Temporally Extended Goals in Pure-Past Linear Temporal Logic: A Polynomial Reduction to Standard Planning, CoRR abs/2204.09960 (2022).
[16] L. Bonassi, G. De Giacomo, M. Favorito, F. Fuggitti, A. Gerevini, E. Scala, Planning for Temporally Extended Goals in Pure-Past Linear Temporal Logic, in: ICAPS, 2023.
[17] A. Polyvyanyy, Z. Su, N. Lipovetzky, S. Sardiña, Goal Recognition Using Off-The-Shelf Process Mining Techniques, in: AAMAS, International Foundation for Autonomous Agents and Multiagent Systems, 2020, pp. 1072–1080.
[18] G. De Giacomo, A. Murano, F. Patrizi, G. Perelli, Timed Trace Alignment with Metric Temporal Logic over Finite Traces, in: KR, 2021, pp. 227–236.
[19] F. Chiariello, F. Maggi, F. Patrizi, ASP-Based Declarative Process Mining, in: AAAI, AAAI Press, 2022, pp. 5539–5547.
[20] T. Chakraborti, Y. Rizk, V. Isahagian, B. Aksar, F. Fuggitti, From Natural Language to Workflows: Towards Emergent Intelligence in Robotic Process Automation, in: BPM (Blockchain and RPA Forum), in: Lecture Notes in Business Information Processing, 459, Springer, 2022, pp. 123–137.
[21] A. Marrella, T. Chakraborti, Applications of Automated Planning for Business Process Management, in: BPM, in: Lecture Notes in Computer Science, 12875, Springer, 2021, pp. 30–36.