# Declarative Trace Alignment via Automated Planning (Extended Abstract)

**Giacomo Acitelli**[1] · **Giuseppe De Giacomo**[1,2] · **Francesco Fuggitti**[1,3]
**Fabrizio Maria Maggi**[4] · **Andrea Marrella**[1] · **Fabio Patrizi**[1]

[1]Sapienza University, Rome (Italy), [2]University of Oxford, Oxford (UK)
[3]York University, Toronto (Canada), [3]Free University of Bozen-Bolzano, Bolzano (Italy)
Contact: fuggitti@diag.uniroma1.it

**Abstract**

Analyzing traces of events produced by a process under execution is critical to many tasks in Business Process Management (BPM). However, modeling, implementation, or human-execution errors often make a trace non-compliant with respect to a model. *Trace alignment* is the problem of aligning process executions to a process model by "repairing" execution traces with a minimal number of modifications. We consider *declarative* trace alignment, where the process model is a formal specification expressed in the linear-time temporal or dynamic logics on finite traces. We solve the problem by reduction to cost-optimal planning and resort to state-of-the-art automated planners. The resulting approach impressively outperforms existing ad-hoc solutions.

The present work is a short version of (De Giacomo et al. 2017), appeared in the Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI-17), and an extension of it is currently under review for an AI journal.

## 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 (Dumas et al. 2018). Usually, BPs are high-level processes involving automated and human-based activities such that, when executed, generate 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* (Adriansyah, Sidorova, and van Dongen 2011; Carmona et al. 2018). 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. Thus, 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. Such changes mainly consist in adding or deleting activities at some instances of the trace when necessary. To solve the trace alignment problem, existing approaches, e.g., (de Leoni, Maggi, and van der Aalst 2012, 2015), are based on ad-hoc implementations of the A$^*$ algorithm, which compromise scalability as the input complexity increases, namely when there are large specifications and long traces. In our work, we reduce the trace alignment problem to deterministic cost-optimal planning (Geffner and Bonet 2013) to exploit the efficiency, versatility, and customization of state-of-the-art planners. In (De Giacomo et al. 2017), the approach is validated through experimental analysis, and results show that it outperforms by far ad-hoc techniques included in the PROM toolkit (promtools.org). Such good results are confirmed and further improved in our recent extension, which not only extends the expressiveness of the specification formalism (more on this in Section 2) but also analyzes several semantically equivalent reductions and evaluates their effectiveness.

Trace alignment is interesting for the AI community in two respects. First, the problem can be applied to executing agents: traces can model agent executions, whereas specifications can model properties that agent executions are expected to satisfy. In this way, solving the problem can be regarded as an approach to identify and possibly fix potential deviations of an agent's behavior from its nominal one. Second, the problem is an interesting application of planning, which turns out to be orders of magnitude more efficient than state-of-the-art ad-hoc tools, thus witnessing once more the power and generality of planning. Finally, it is worth noticing that parts of the reduction and encodings devised in the work are applicable in general to any problem that includes temporal constraints expressible as finite-state automata and that the experimental study carried out here provides useful guidelines for an efficient representation of such constraints as part of a planning domain. Thus, while devised and presented on the specific case of trace alignment, the obtained results are of general applicability.

## 2 Model Specification Languages

A BP model defines the (possibly partial) execution order of the activities of interest and can be specified either *procedurally* or *declaratively*. A procedural specification consists of an executable model like a Petri net, whereas a declarative specification consists of a set of formal constraints over the traces (BP executions). While a procedural specification prescribes the execution steps of the BP (i.e., when and under which conditions the various activities have to be executed), a declarative specification uses formal languages to prescribe the set of requirements BP executions have to comply with. Generally, the former approach is well suited for actual execution, while the latter provides a process description amenable to various forms of analysis.

Previous approaches employing declarative models, such as (de Leoni, Maggi, and van der Aalst 2012, 2015; De Giacomo et al. 2016, 2017), are all based on a restricted set of predefined template modeling formulas called DECLARE (van der Aalst, Pesic, and Schonenberg 2009). Although DECLARE identifies useful patterns, it is limited in the expressiveness. For this reason, in our recent work, we consider model specifications provided as formulas in full-fledged linear-time temporal logic and linear-time dynamic logic on finite traces, respectively $\text{LTL}_f$ and $\text{LDL}_f$ (De Giacomo and Vardi 2013), which are strictly more expressive than DECLARE. These temporal logics express properties that involve trace positions, together with their interpretation – in our case, the events occurring at that position. $\text{LTL}_f$ and $\text{LDL}_f$ give great flexibility in constraints modeling. While $\text{LTL}_f$ is usually sufficient to capture properties of interest without compromising readability, $\text{LDL}_f$ augments $\text{LTL}_f$ with regular expression keeping the same complexity properties. The key aspect of using $\text{LTL}_f$ and $\text{LDL}_f$ formulas as specification models is exploiting their translation to equivalent finite-state automaton (De Giacomo and Vardi 2013) (more on this later in Section 3).

## 3 Trace Alignment as Cost-Optimal Planning

We reduce trace alignment to cost-optimal planning (Geffner and Bonet 2013), which is the variant of deterministic planning where actions are associated with costs, and the task is to find a minimal-cost plan that reaches a goal state, with plan cost defined as the sum of the costs of the component actions. The reduction is based on two main observations: (1) trace modifications are actions that modify the input trace by adding or removing events; (2) every $\text{LTL}_f$/$\text{LDL}_f$ formula can be fully characterized by a finite-state automaton accepting all and only the traces that satisfy the formula (De Giacomo and Vardi 2013).

Reducing trace alignment to cost-optimal planning amounts to the following steps. First, we provide an automata-theoretic formalization of trace alignment that characterizes the notion of solution in terms of languages. Second, we define a planning problem encoding such formalization. Last, we use any state-of-the-art optimal planner, such as FAST-DOWNWARD (Helmert 2006) or SYMBA*-2 (Torralba et al. 2014), to efficiently solve the task.

The idea underlying the automata-theoretic formalization is to model the execution trace as a deterministic automaton and the $\text{LTL}_f$/$\text{LDL}_f$ specification formula as a nondeterministic automaton. After that, these automata are extended in a particular way to allow the execution of special events that identify traces modifications. We call such events "repair events" and define them as follows: *skip* an event, i.e., leave the event unchanged; *delete* an event from a position; *add* a new event at a certain position. The solution to the trace alignment is given in terms of languages on these customized automata. Every cost-optimal trace accepted by both automata (i.e., their intersection) represents a solution to the trace alignment problem. More specifically, the resulting trace tells us what are the minimal number of modifications to apply to the original trace so to have a trace compliant with the given model.

The second step encodes the automata-theoretic solution into a planning problem. As usual, a planning problem consists of a domain, an initial state, and a goal formula. In our case, the domain simulates the execution of the modified automata representing the process execution and the model specification, with actions modeling the reading of an event from the input trace, the addition of an event in the current position, or the deletion of the event from the current position. In the initial state of the planning problem, the automata are in their initial states and no event from the input trace has been read. The goal requires that all the input trace has been read (including the possibly added events) and that both automata ends up in a final state. As to action costs, although the reduction can accommodate any cost model, we consider, for simplicity, a unitary cost for every modification and a zero cost for reading an activity in the trace.

Ultimately, solving the planning problem corresponds to searching for the shortest plan exploiting off-the-shelf optimal planners (third step).

## 4 Results

The proposed approach has been implemented in an open-source tool[1] that, given a trace alignment instance, generates the corresponding planning problem and then employs any classical planner to solve it. We carried out an extensive experimental evaluation to evaluate the planning-based solution to declarative trace alignment, considering both synthetic and real logs from existing benchmarks.

The goal of the analysis is twofold: understanding the impact of different encodings on performance, as well as of the planner used to find the solution, and comparing our approach against other existing ones. As to the encodings, we have tested several semantically equivalent variants, spanning from a very high-level and easily-readable one up to grounded STRIPS-like versions. The encodings differ in many features, such as the presence of conjunctive goals or the way states are encoded. As to the comparison with other tools, we have considered a state-of-the-art technique included in the PROM toolkit for BP analysis (promtools.org), based on an $A^*$ implementation specifically tailored for trace alignment (de Leoni, Maggi, and van der Aalst 2012, 2015). Our system impressively outperforms all such tools.

[1] https://github.com/whitemech/trace-alignment

## Acknowledgments

This work has been partially supported by the ERC Advanced Grant WhiteMech (No. 834228), by the EU ICT-48 2020 project TAILOR (No. 952215), by the PRIN project RIPER (No. 20203FFYLK), by the JPMorgan AI Faculty Research Award "Resilience-based Generalized Planning and Strategic Reasoning", by the Italian projects RoMA and NEPTIS, by the Sapienza Project "Drape", by the Italian cluster SM&ST and by the Sapienza project "Immersive Cognitive Environments".

## References

Adriansyah, A.; Sidorova, N.; and van Dongen, B. F. 2011. Cost-Based Fitness in Conformance Checking. In *ACSD 2011*. IEEE.

Carmona, J.; van Dongen, B. F.; Solti, A.; and Weidlich, M. 2018. *Conformance Checking - Relating Processes and Models*.

De Giacomo, G.; Maggi, F.; Marrella, A.; and Patrizi, F. 2017. On the disruptive effectiveness of automated planning for LTLf-based trace alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

De Giacomo, G.; Maggi, F. M.; Marrella, A.; and Sardiña, S. 2016. Computing Trace Alignment against Declarative Process Models through Planning. In *26th Int. Conf. on Automated Planning and Scheduling (ICAPS 2016)*.

De Giacomo, G.; and Vardi, M. Y. 2013. Linear Temporal Logic and Linear Dynamic Logic on Finite Traces. In *23th Int. Conf. on AI (IJCAI'13)*.

de Leoni, M.; Maggi, F. M.; and van der Aalst, W. 2015. An alignment-based framework to check the conformance of declarative process models and to preprocess event-log data. *Inf. Syst.*, 47: 258–277.

de Leoni, M.; Maggi, F. M.; and van der Aalst, W. M. P. 2012. Aligning Event Logs and Declarative Process Models for Conformance Checking. In *10th Int. Conf. on Business Process Management (BPM 2012)*.

Dumas, M.; La Rosa, M.; Mendling, J.; Reijers, H. A.; et al. 2018. *Fundamentals of Business Process Management*, volume 2. Springer.

Geffner, H.; and Bonet, B. 2013. A Concise Introduction to Models and Methods for Automated Planning. *Synth.Lect. on AI and ML*, 8(1).

Helmert, M. 2006. The Fast Downward Planning System. *J. Artif. Intell. Res.(JAIR)*, 26: 191–246.

Torralba, A.; Alcazar, V.; Borrajo, D.; Kissmann, P.; and Edelkamp, S. 2014. Symba: A symbolic bidirectional a planner. In *International Planning Competition*, 105–108.

van der Aalst, W.; Pesic, M.; and Schonenberg, H. 2009. Declarative Workflows: Balancing Between Flexibility and Support. *Computer Science - R&D*, 23(2): 99–113.