Presentation Speech

Francesco Fuggitti

Good morning, my thesis focuses on the application of LTL and Past LTL on finite traces in planning and declarative process mining. I will introduce the topic, set the objectives of the thesis and explain major contributions giving some examples and then, discuss conclusions.

Linear Temporal Logic is a well known formal language for expressing temporal specification and it has been applied both in artificial intelligence and in business process management. We studied its variants evaluated on finite traces LTLf, which looks at the future, and on PLTL, which looks into the past. In AI, these temporal logics have been used for expressing extended goals in planning; while in BPM, especially in declarative process mining, they have been used to express constraint formulas.

The objectives of the thesis are to provide a new technique to transform LTLf/ PLTL formulas into Deterministic Finite-state Automaton, provide an approach to planning for extended goals (reducing the problem to standard planning problem) and provide an extension of an new approach to declarative process mining. Finally, to implement all these topics.

So, PLTL and LTLf are two variants of LTL, they have the same syntax of LTL, but they are evaluated over finite traces. They talk about properties over time. LTLf looks at the future with four main (temporal) operators: next, until, eventually, always. The next operator means that at the next instant of time a property will be true; the until operator means that the second proposition will be true at a certain instant in the future and until that time the first is true. Eventually means that sooner or later the proposition will be true and, finally, always means that a certain property is true at each instant. On the other hand, PLTL looks at the past, hence its operators are the dual of the LTLf operators. Reasoning with these two formalisms is done by transforming the formula into the corresponding Deterministic Automaton knowing that a trace makes the formula true if and only if it is recognized by the language associated to the automaton. We will exploit this feature for our purposes.

The new technique for translating a formula to its corresponding automaton is based on a translation procedure. We start from a formula, translate it to FOL on finite sequences and evaluate it on the right instant: in $x = 0$ for LTLf formulas since they look at the future, while in $x = last$ where $last$ is the last instant of the trace for PLTL formulas since they look at the past. Then, we apply the MONA tool that transforms a logical specification expressed in FOL to a minimum DFA.

This procedure has been implemented in a python package called LTLf2DFA that supports parsing of LTLf/PLTL formulas, translation to FOL and DFA and the option for DECLARE assumption. Moreover, it is available as an online tool at the link. Here, we report two output examples. On the left, φ_1 means that is always true that if a happens then b will happen at the next instant of time. While φ_2 requires that g happened at least once in the past.
This tool has been used into both implementations of planning and declarative process mining.

Next, we studied FOND planning for extended temporal goals. In particular, a fully observable non deterministic domain with initial state is a tuple where F is the set of fluents, A the set of actions, there is the set of states, actions preconditions and actions effects. It can be specified in standard PDDL as a domain D and a problem P. The non determinism is captured by the fact that the agent chooses the action to execute while the environment chooses the successor state. Then, a goal will be an LTLf/PLTL formula and planning will be a game between the two players. Finally, a plan will be a strategy for the agent to win such a game.

We proposed an approach, called FOND4LTLfPLTL, in which the idea is to reduce the problem to standard FOND planning expressed in PDDL. It is made of the following steps. First, we transform the goal formula into the corresponding automaton A_φ, through LTLf2DFA. Secondly, since the formula has grounded symbols, we capture the general representation of the automaton on the domain applying a transformation from objects of interest to variables, modifying its alphabet Σ and the set of states Q.

Then, we introduce a new predicate called *turnDomain* that allows to alternate steps between the domain and the automaton. After that, we encode the transition function of the parametric automaton as a new operator called `trans`, where we have variables as parameters, the negation of *turnDomain* as precondition and all transitions are encoded as conditional effects. Finally, we produce the new initial state and the goal specification in the problem. For the initial state, we add *turnDomain* and the initial state of the automaton (now, with instantiated symbols) to the initial state of the domain; while, for the goal we want that the *turnDomain* is true and that the automaton is in a final state.

Once we have modified both the PDDL domain and problem, we feed them to a FOND planner and get the plan. We implemented our approach in python using the FOND-SAT planner. In the example, we show an execution of a triangle tireworld domain in which the objective is to drive from a location to another. A tire may be going flat and if there is a spare tire in the location of the car, then the car can use it to fix the flat tire. We impose a PLTL goal that says: "reach location 22 passing through location 31". Our approach works, indeed we can observe the resulting transition system output from FOND-SAT, where no matter how the environment behaves, the agent can reach the goal.

Regarding BPM, declarative process mining is a technique that looks if a trace is interesting with respect to a given constraint φ, expressed as a temporal specification. This technique has a problem, called "ex falso quod libet" meaning that a constraint can be satisfied even though never activated. Recently, Cecconi and others proposed in the Janus approach a new definition of the constraint that has an activation condition and a formula with mixed past and future operators. Moreover, they provide the definition of a function to compute the interestingness degree which is the ratio between the number of times the constraint is fulfilled on the total number of activations. For example, given the trace t the interestingness degree is 2 over 3 since a appears three times, but the formula is satisfied only twice. However, this approach has two main drawbacks: namely α can be only a single task and the implementation is limited to declare constraints.

Hence, we extended the approach by giving a new representation of the constraint formula. Our constraint is a disjunction triples made of separated formulas with a past formula on the left, a propositional formula that triggers potential interest on the trace on the center and a future formula on the right. With this new definition, we give another definition of the function for computing the interestingness degree which we proved to be more general than the one defined in Cecconi and others.

Also in this case, we provided an implementation that takes advantage of our generalization. So, we can have any type of constraint since automata are generated by LTLf2DFA. Moreover, we have evaluated our tool against the real world event log Sepsis, which reports trajectories of patients showing symptoms of sepsis. In particular, we can see that the triggering condition now is the propositional formula "(Leucocytes and LacticAcid)" and the interestingness degree for the trace is 0.5.

Finally, we can discuss results achieved by this thesis and possible future works. We provided the LTLf2DFA tool implementing the translation procedure from LTLf/PLTL to DFA. Then, we proposed and implemented an approach to FOND planning for LTLf/PLTL goals and lastly, we extended the janus approach theoretically and practically.

There are several possible future works to be done. Here, we reported some of them. First, the LTLpf logic can be investigated. Secondly, we can extend our FOND4LTLfPLTL approach to partially observable domains and, finally, optimizing and enriching all developed tools. Thanks for the attention!