Excellence Path

Francesca Cardia

a.a. 2020/2021

Abstract

I have realized my Excellence Path under the supervision of Prof. Giuseppe De Giacomo. During this path I have analyzed the multi-tier domains, focusing the attention on the temporal goals.

1 Brief introduction

In this work I have integrated temporal goals in a multi-tier framework described in the paper [Ciolek et al.(2020)Ciolek, D'Ippolito, Pozanco, and Sardiña]. I have used the $FOND4LTL_f$ tool that allows to compile Fully Observable Non-Deterministic (FOND) planning problems (see the paper [Geffner and Geffner(2018)]) with temporally extended goals, specified either in LTL_f or in $PLTL_f$ (see the papers [Fuggitti(2020)], [Pereira et al.(2021)Pereira, Fuggitti, and Giacomo]). Unlike Classical Planning (see the paper [Geffner and Bonet(2013)]), in which actions are deterministic, in FOND planning, some actions have an uncertain outcome, namely they are non-deterministic. The $FOND4LTL_f$ tool is available at github page `https://github.com/whitemech/FOND4LTLf` or at online version `https://fond4ltlf.herokuapp.com/`.

2 Temporal Goals

Before of introducing the work description, it is important to spend a few words about Temporal Goals, that constitute the core of this work. They are represented through Temporal logic formalism.

Temporal Logic a convenient formalism for specifying and verifying properties of reactive systems. In particular these logics are used when propositions have their truth value dependent on time. More specifically, Temporal logic extends classical propositional logic with a set of temporal operators that navigate between worlds using this accessibility relation. In the following subsections I focus the attention on LTL, LTL_f, *and* $PLTL_f$.

2.1 Linear Temporal Logic (LTL)

Linear Temporal Logic (LTL) is a modal temporal logic with modalities referring to time. Given a propositional symbols $\mathcal{P}$ a valid temporal formula ϕ can be defined as follows:

$$\phi \quad ::= \quad a \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \bigcirc\phi \mid \phi_1 \mathcal{U} \phi_2$$

where $a \in \mathcal{P}$.

In particular the $\bigcirc$(*next-time*) and $\mathcal{U}$ (*until*) are temporal operators, while $\top$ and $\bot$ symbols need to denote *true* and $false$ respectively. Moreover, all classical logic operators $\vee, \Rightarrow, \Leftrightarrow, true$ and $false$ can be used. Then, $\bigcirc\phi$ needs to specify that ϕ is true at the *next* instant, $\phi_1 \mathcal{U} \phi_2$ denotes that at some future instant, φ_2 will hold and *until* that point ϕ_1 holds.

Moreover it is also possible to define common abbreviations for some specific temporal formulas: *eventually* as $\Diamond\phi \doteq true \mathcal{U} \phi$, *always* as $\Box\phi \doteq \neg\Diamond\neg\varphi$, *weak-next* as $\bullet\varphi \doteq \neg\bigcirc\neg\varphi$ and *release* as $\varphi_1 \mathcal{R} \varphi_2 \doteq \neg(\neg\varphi_1 \mathcal{U} \neg\varphi_2)$.

2.2 Linear Temporal Logic over Finite Traces (LTL_f)

Linear Temporal Logic on Finite Traces (LTL_f) is the variant of LTL interpreted over finite traces (see the paper [De Giacomo and Vardi(2015)]) that refers to future events (as LTL) . In some cases the interpretation of a formula over finite traces completely changes its meaning with respect to the one over infinite traces. The syntax is equivalent to LTL and similarly it is possible to define common abbreviations for temporal operators:

$$\Diamond\phi \doteq true \mathcal{U} \phi \tag{1}$$

$$\Box\phi \doteq \neg\Diamond\neg\phi \tag{2}$$

$$\bullet\phi \doteq \neg\bigcirc\neg\phi \tag{3}$$

$$\phi_1 \mathcal{R} \varphi_2 \doteq \neg(\neg\phi_1 \mathcal{U} \neg\phi_2) \tag{4}$$

$$Last \doteq \bullet false \tag{5}$$

$$End \doteq \Box false \tag{6}$$

In particular the difference is in the abbreviations 5 and 6 . The first denotes the last instance of the trace, while the second indicates that the trace is ended.

A trace $\tau = \tau_0\tau_1 \ldots \tau_n$ is a sequence of propositional interpretations, where $\tau_m \in 2^P$ $(m \geq 0)$ is the $m-th$ interpretation of τ , and τ represents the length of τ . A finite trace is defined as $\tau_m \in 2^{P^*}$ and given τ and an LTL_f formula ϕ it is possible to define when ϕ holds in τ at position i (where $0 \leq i < |\tau|$), written $\tau, i \models \phi$:

$$\tau, i \models a, \iff a \in \tau_i \tag{7}$$

$$\tau, i \models \neg\phi \iff \tau, i \not\models \phi \tag{8}$$

$$\tau, i \models \phi_1 \wedge \phi_2 \iff \tau, i \models \phi_1 \wedge \tau, i \models \phi_2$$

$$\tau, i \models \bigcirc\phi \text{ iff } i+1 < |\tau| \wedge \tau, i+1 \models \phi \tag{9}$$

$$\tau, i \models \phi_1 \mathcal{U} \phi_2 \iff \exists j. i \leq j < |\tau| \wedge \tau, j \models \phi_2, \wedge\forall k, i \leq k < j, \tau, k \models \phi_1. \tag{10}$$

So An LTL_f formula ϕ is true in τ , denoted by $\tau \models \phi$, if and only if $\tau, 0 \models \phi$

2.3 Past Linear temporal Logic over Finite Traces ($PLTL_f$)

Now I describe the Past Linear Temporal Logic for Finite Traces b ($PLTL_f$, see paper [De Giacomo et al.(2020)De Giacomo, Stasio, Fuggitti, and Rubin]) based on $PLTL$ which uses temporal modalities for referring to past events, instead of future ones (as in LTL, LTL_f cases).

$PLTL_f$ formulas are built on top from a set P of propositional symbols and are closed under the boolean connectives, the unary temporal operator $\ominus$ (*previous-time*) and the binary operator $\mathcal{S}$ (*since*)

The formulas are the following:

$$\varphi \quad ::= \quad a \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \ominus\varphi \mid \varphi_1 \mathcal{S} \varphi_2$$

where $a \in \mathcal{P}$.

I can illustrate also the common abbreviations for temporal operator :

$$\diamondsuit\phi \doteq true \mathcal{S} \phi \tag{11}$$

$$\boxminus\phi \doteq \neg\diamondsuit\neg\phi \tag{12}$$

In particular $\diamondsuit\phi$ in 11 is called *once* while $\boxminus\phi$ in 12 is known as *historically.*

Also in this case given a finite trace τ and an $PLTL_f$ formula it is possible to define when ϕ holds in τ at position i (where $0 \leq i < |\tau|$).

$$\tau, i \models \ominus\phi \iff i-1 \geq 0 \wedge \tau, i-1 \models \phi \tag{13}$$

$$\tau, i \models \phi_1 \mathcal{S} \phi_2 \iff \exists k. 0 \leq k \leq i \wedge \tau, k \models \phi_2, \wedge\forall j, k < j \leq i, \tau, j \models \phi_1 \tag{14}$$

So A $PLTL_f$ formula ϕ is true in τ, denoted by $\tau \models \phi \iff \tau, |\tau| - 1 \models \phi$.

3 Available Tool

In order to obtain a multi-tier structure able to integrate also temporal goals, I have used *pypddl-translator*. In fact this represents a crucial starting point in order to develop this work related to the temporal goals in a multi-tier framework.

3.1 Pypddl-translator

This is a PDDL's domain, problem parser and translator written in Python3 that uses the library *ply* `http://www.dabeaz.com/ply/`.
This tool can read PDDL planning domains and problems either as one single file, or in two separated files (domain file and problem file), it can print them to console or files and it can handle typed and untyped (if there is only one) objects and parameters .

Moreover it contains an interface that allows to modify domains and problems.

The interesting part for this thesis with regard to this tool refers to multi-tier context; in fact it can produce MTP domain and problem files from a labelled PDDL planning and problem description. The Planning domains supports the :requirements :strips, :typing, :equality, :probabilistic- effects. It allows to use non deterministic effects making use of *oneof* keyword and conditional effects using when keyword.

It is possible to download this tool from github:
`https://github.com/ssardina-planning/pypddl-translator`.
From the compilation of a `labelled domain` with pypddl-traslator we can get:

- a different action for each level.
- a set of new predicates to signal the current level, the effects, the passage from acting phase, non acting phase, the execution of unfair action etc..
- for each action defined in labelled domain a new set of actions to simulate conditional effects allowing the alignment phase.
- a new set of actions to check the reaching of goal for each different level.
- a new set of unfair actions to allow the degradation phase.
- a set of actions in order to perform the degradation to lower levels.
- a set of actions to remain on the current level.

More specifically these are the predicates added after the compilation phase:

- *e_dx* : for each level *dx* it signals that model *dx* is a/the highest model explaining the effect of the last executed action.
- *l_dx* : for each level *dx* it tracks the most "ambitious" compatible model so far.
- *act*: it denotes that the system is in the acting phase.
- *not(act)* : it denotes that the system is in the alignment phase.
- u_a: for each action a, this predicate allows the execution of an unfair action.

- eff_edx_action: it enables the alignment phase.
- *end*: it denotes the goal achievement for the current domain of execution.

3.2 Multi-tier compilation

The new actions are added from the parser during the compilation:

- for each action x defined in `labelled domain`, is defined an associated action x for each layer; for instance considering two layers: x_{d1} and x_{d2}.
- for each action x defined in `labelled domain`, are inserted new actions to simulate conditional effects allowing alignment phase: *x_en1_explained_by_dn1*, where x is the name of the action and $n1, n2$ are the numbers that represent the planning domains.
- the *degrade* actions to downgrade to lower levels.
- new actions to *continue* in the current layer after the alignment phase, without downgrading.
- new actions to check goal related to the different layers: *check_goal_dn*, where n is the number that represents the layer.

4 Work Description

4.1 Work Overview

The work can be divided in different phases:

1. in the first phase I have designed a new multi-tier domain.
2. in the second phase I have split this multi-tier domain into different domain components and I have used $FOND4LTL_f$ to integrate the temporal goals.
3. in the third phase I have modified *pypddl-translator* 3.1;so it can take as input multiple domains and multiple problems characterized by temporal goals, and it can return an unique multi-tier domain and multi-tier problem with temporal goals integration.

4.2 Steps Description

The paper [Ciolek et al.(2020)Ciolek, D'Ippolito, Pozanco, and Sardiña] represents the starting point to encode the multi-tier model. In fact in this paper the multi-tier framework is explained through theorems, definitions, and encoding procedures; moreover an implementation example is presented. A multi-tier domain is characterized by many hierarchical planning domains where the agent can operate. Each planning domain is characterized by different assumptions and different goals. The objective of

the agent consists of reaching all the goals that characterize the multi-tier model. In particular in this paper the agent starts from the highest planning domains and it can downgrade to less refined domains when something "goes wrong", so that when some less refined effects are seen. The implementation related to the paper [Ciolek et al.(2020)Ciolek, D'Ippolito, Pozanco, and Sardiña] uses *pypddl-translator* (see github page `https://github.com/ssardina-planning/pypddl-translator`) to convert a labelled domain and a labelled problem (that are characterized by labels which indicate the planning domains, d_3 for planning domain 3, d_2 for planning domain 2 etc..) into a multi-tier files; then also in this case I have considered this tool. So now I describe accurately the steps of the work.

In the first phase I have created a new multi-tier model. I have chosen the **gripper domain**. In this model the agent is a robot with a gripper; there are two rooms($room_1$ and $room_2$) and in each room there is a finite number of balls. The agent can execute three different actions: *PICK*, *DROP* and *MOVE*. The first one needs to take the balls, the second to drop them and the third action needs to move from one room to another. I have codified the domains as follows: practically I have decided to design the model into two planning domains, d_2 the most ambitious and d_1 the least ambitious.

The highest planning domain is characterized by less executions but more assumptions (in fact, it represents the most ambitious layer), instead, the lowest planning domain by more executions and less assumptions.

The two domains contain the same actions with the same preconditions but different effects. In fact the planning domain d_2 is more refined and here "all work", while the lowest domain is less refined and here "something may not work". The agent starts to operate at the highest domain d_2, according to its initial state it executes some actions(in the current level and downgrading to the lowest level) to reach the two goals related to the two planning domains.

The code related to this new multi-tier model can be available at `https://github.com/francycar/Excellence_path.git`. In particular the folder **labelled_version** contains the labelled domain and the labelled problem, while the folder **multi-tier_version** includes the related multi-tier versions generated by *pypddl-translator*.

Now I analyze the second phase of the work. The aim of this work consists of applying temporal goals into this multi-tier framework. Then in order to realize this, I have used $FOND4LTL_f$ tool. This tool accepts as input only standard domains and standard problems, so it cannot treat labelled files. Then I have split the labelled domain and the labelled problem into their components. In fact each labelled domain is a "concatenation" of multiple standard domains (in the **gripper** example, two domains), while each labelled problem is a "concatenation" of multiple standard problem files. In this way, it is possible to treat separately the single couple of domain and problem related to each planning layer. So from the *labelled domain* and *labelled problem* files related to **gripper** domain, I obtain four different files:

- *gripper_domain_d*1.*pddl*
- *gripper_problem_d*1.*pddl*

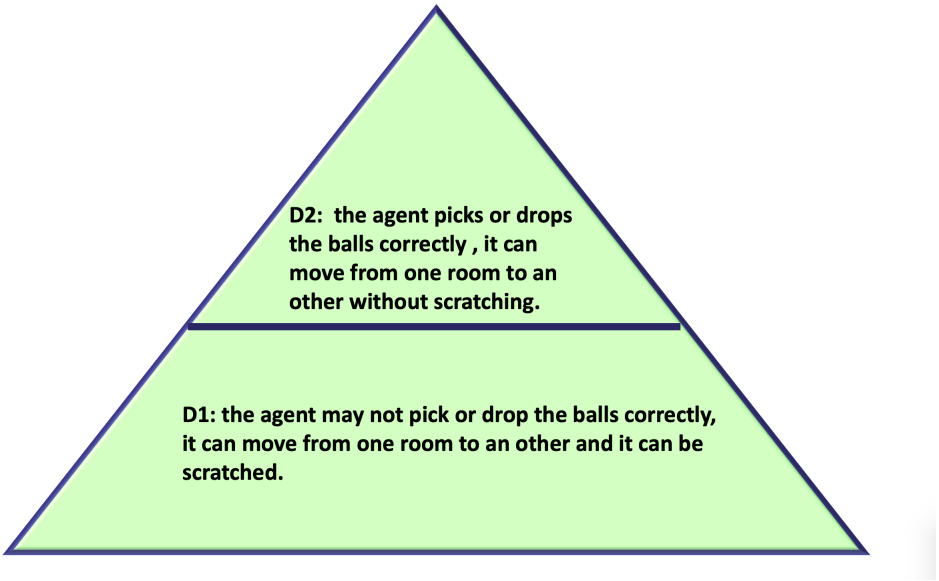

Figure 1: Planning domains overview

- $gripper\_domain\_d2.pddl$
- $gripper\_problem\_d2.pddl$

So now I can treat the two couples $\langle domain, problem \rangle$ separately. I define two LTL_f temporal goals, one for the highest model d_2 and an other for the lowest model d_1.

- temporal goal for D_2:

  $F(carry\_b1\_g)\&G(atrob\_r1 -> F(atrob\_r2\&at\_b1\_r2))$

  this means that soon or later the robot carries the ball b_1 and every time that the robot is in $room_1$, then soon or later it will be in $room_2$ and also the $ball_1$ will be in $room_2$.

- temporal goal for D_1:

  $F(scratch)\&G(atrob\_r1 -> F(free\_g\&atrob\_r2))$

  this means that soon or later the robot will be scratched and every time that the robot is in $room_1$, then soon or later it will be ain $room_2$ and its gripper will be free.

From the compilation with $FOND4LTL_f$ I obtain two new couples of files with temporal goals integration:

- $gripper\_temp\_goal\_d1.pddl$
- $gripper\_temp\_goal\_p1.pddl$
- $gripper\_temp\_goal\_d2.pddl$

- *gripper_temp_goal_p2.pddl*

Also these files are in the github page `https://github.com/francycar/Excellence_path.git`; in the folder *d2_temporal_goals* there are the domain and problem files of domain 2 with the temporal goal integration, and consequently in *d1_temporal_goals* there are the domain and problem files of domain 1 with temporal goal.

So now I can introduce the third phase of the work. This phase is characterized by the translation into multi-tier model. So I have modified *pypddl-translator*'s implementation; in fact since $FOND4LTL_f$ cannot treat labelled models (but I would obtain a multi-tier model), I write the path of the single standard domains and problems (with temporal goals integration) on two separated csv files. My new *pypddl-translator* takes as input the two csv files and returns the compiled multi-tier domain and problem with the temporal goals integration. It is always possible to find the new *pypddl-translator* version on my github page. So in this way the designer has not to write a labelled domain and problem, but only standard couples $\langle domain, problem \rangle$.

So the new compilation returns a multi-tier version of domain and problem with the temporal goals:

- *multigripd.pddl*
- *multigripp.pddl*

These files are included in the folder **Multi_tier_temp_goals**. Moreover during the compilation also two intermediate files are generated: the labelled domain and the labelled problem with the temporal goals (they are included in the folder **intermediate_labelled** inside **Multi_tier_temp_goals** directory).
The pipeline of the work is the following:

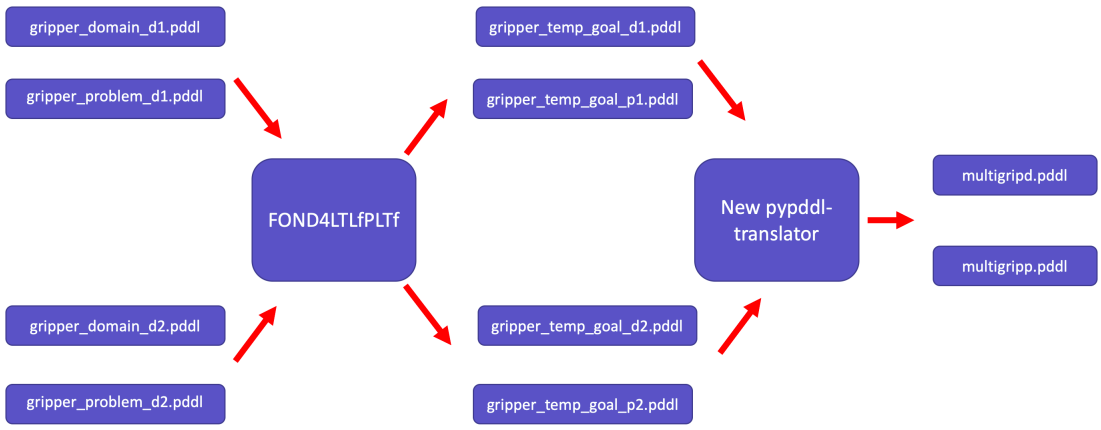

Figure 2: Steps overview

References

[Ciolek et al.(2020)Ciolek, D'Ippolito, Pozanco, and Sardiña] Daniel Ciolek, Nicolás D'Ippolito, Alberto Pozanco, and Sebastian Sardiña. Multi-Tier automated planning for adaptive behavior. In *Proceedings of the 30th International Conference on Automated Planning and Scheduling (ICAPS)*, pages 66–74. AAAI Press, 2020.

[De Giacomo and Vardi(2015)] Giuseppe De Giacomo and Moshe Y. Vardi. Synthesis for ltl and ldl on finite traces. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 1558–1564. AAAI Press, 2015. ISBN 9781577357384.

[De Giacomo et al.(2020)De Giacomo, Stasio, Fuggitti, and Rubin] Giuseppe De Giacomo, Antonio Di Stasio, Francesco Fuggitti, and Sasha Rubin. Pure-past linear temporal and dynamic logic on finite traces. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 4959–4965. International Joint Conferences on Artificial Intelligence Organization, 7 2020. doi: 10.24963/ijcai.2020/690. URL `https://doi.org/10.24963/ijcai.2020/690`. Survey track.

[Fuggitti(2020)] Francesco Fuggitti. Fond planning for ltlf and pltlf goals, 2020.

[Geffner and Bonet(2013)] Hector Geffner and Blai Bonet. *A Concise Introduction to Models and Methods for Automated Planning*. Morgan & Claypool Publishers, 2013.

[Geffner and Geffner(2018)] Tomas Geffner and Hector Geffner. Compact policies for fully observable non-deterministic planning as SAT. In *Twenty-Eighth International Conference on Automated Planning and Scheduling (ICAPS)*, 2018.

[Pereira et al.(2021)Pereira, Fuggitti, and Giacomo] Ramon Fraga Pereira, Francesco Fuggitti, and Giuseppe De Giacomo. Recognizing ltlf/pltlf goals in fully observable non-deterministic domain models, 2021.