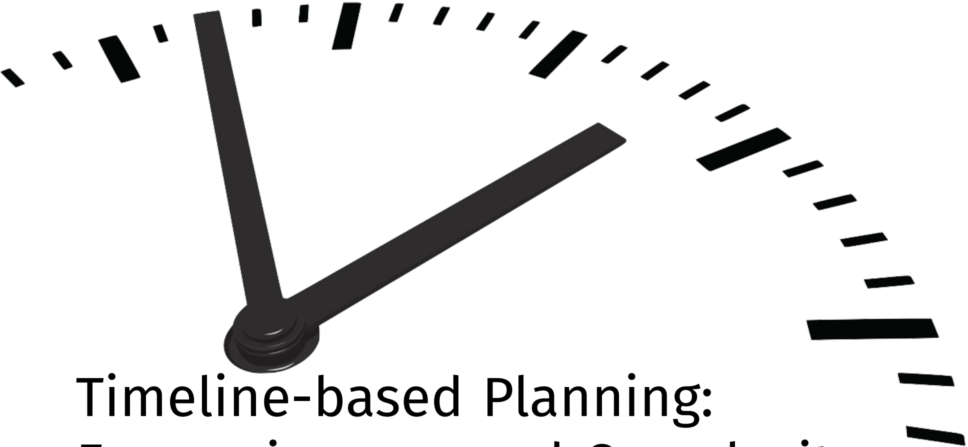


Timeline-based Planning: Expressiveness and Complexity

Nicola Gigante

FMAI 2019 - *Second Workshop on Formal Methods and AI*
Rennes, France - May 1–4, 2019

Automated planning

Automated planning is a field of Artificial Intelligence that studies how to design systems that, given a description of the world, can autonomously obtain a given goal.

Automated planning

Automated planning is a field of Artificial Intelligence that studies how to design systems that, given a description of the world, can autonomously obtain a given goal.

Automated Planning is one of the oldest fields of AI:

1. Tracing back to the ’60s
2. Efficient systems developed over the last decades
3. Successful employment in a variety of application domains
4. Most common planning languages (i.e. PDDL) are action-based

Timeline-based planning

Timeline-based planning is an approach to AI planning originally introduced in the context of planning and scheduling of space operations.

Muscettola (1994)

N. Muscettola (1994). “HSTS: Integrating Planning and Scheduling.” In: *Intelligent Scheduling*. Ed. by Monte Zweben and Mark S. Fox. Morgan Kaufmann. Chap. 6, pp. 169–212

Timeline-based planning

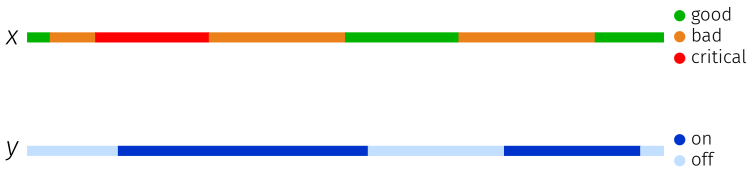
Timeline-based planning is mostly focused on temporal reasoning:

- no clear separation between actions, states, and goals;
- planning problems are modeled as systems made of a number of independent, but interacting, components;
- components are described by state variables;
- the timelines describe their evolution over time;
- the evolution of the system is governed by a set of temporal constraints called synchronization rules.

Timeline-based planning shines when modeling systems made of many components, rather than the behavior of a single agent.

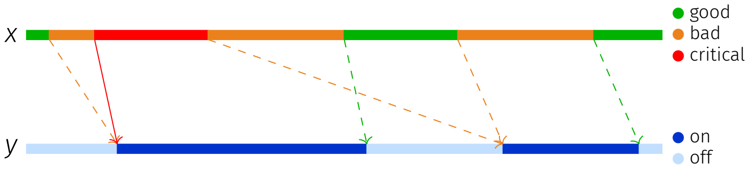
Example

A switch (component y) reacts to an alert signal (component x).



Example

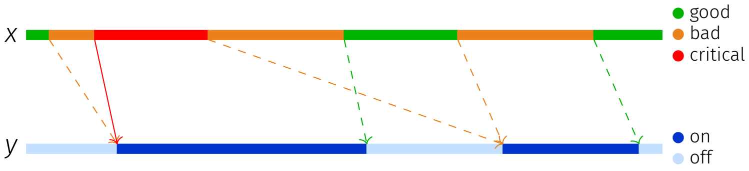
A switch (component *y*) reacts to an alert signal (component *x*).



1. The switch must eventually be turned *on* ($y =$ ●) some time after the signal becomes *bad* ($x =$ ●),
2. The switch must turn *on* no later than ten time steps after the signal becomes *critical* ($x =$ ●)
3. The switch cannot turn *off* ($y =$ ●) before at least ten time steps after the signal became *good* again ($x =$ ●).

Example

A switch (component *y*) reacts to an alert signal (component *x*).

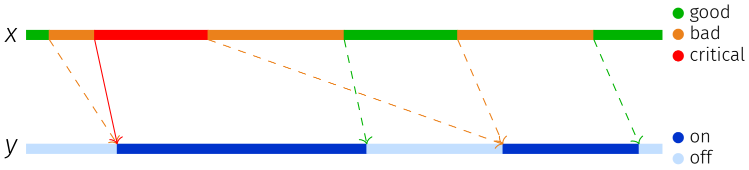


We can express these constraints with **synchronization rules**:

- if a **token** *a* where $x =$ ● starts, a token *b* with $y =$ ● starts as well;
- if a token *a* where $x =$ ● starts, a token *b* with $y =$ ● starts as well **at most 10 steps** after the start of *a*;
- any token *a* where $y =$ ● starts **at least 10 steps** after a the start of a token *b* with $x =$ ●;

Example

A switch (component *y*) reacts to an alert signal (component *x*).



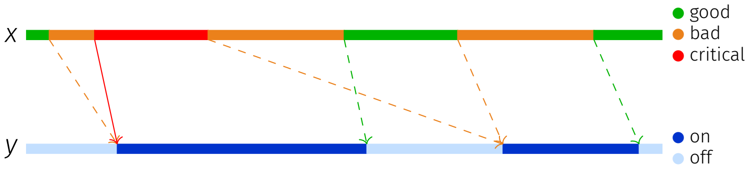
We can express these constraints with synchronization rules:

- if a token *a* where $x = $ ● starts, a token *b* with $y = $ ● starts as well at most 10 steps after the start of *a*;

$$a[x = \bullet] \longrightarrow \exists b[y = \bullet]\ .\ \mathsf{start}(a) \leqslant_{[0,10]} \mathsf{start}(b)$$

Example

A switch (component *y*) reacts to an alert signal (component *x*).



Trigger-less rules for goals, invariants, etc...

$$\top \longrightarrow \exists a[x = ●]b[x = ●]c[x = ●] \; . \\ \mathrm{end}(a) \leqslant \mathrm{start}(b) \wedge \mathrm{end}(b) \leqslant \mathrm{start}(c)$$

Timeline-based planning problems

Timeline-based planning problems

Given a problem $P = (SV, S)$, where SV is a set of state variables, and S is a set of rules over SV, find a set of timelines over SV that satisfy the rules in S.

Timelines and space exploration

Timeline-based planning has been (and still is) used in actual mission planning and scheduling systems, both on-board and on-ground.

HSTS (Muscettola 1994)
EUROPA (Bedrax-Weiss et al. 2005)
ASPEN (Chien et al. 2000)

European Space Agency

APSI-TRF (Fratini and Donati 2011)
GOAC (Fratini, Cesta, et al. 2011)

Timelines and space exploration

Timeline-based planning has been (and still is) used in actual mission planning and scheduling systems, both on-board and on-ground.

Recent notable examples:

Rosetta (ASPEN)

Steve A. Chien, Gregg Rabideau, Daniel Tran, Martina Troesch, Joshua Doubleday, Federico Nespoli, Miguel Perez Ayucar, Marc Costa Sitja, Claire Vallat, Bernhard Geiger, Nico Altobelli, Manuel Fernandez, Fran Vallejo, Rafael Andres, and Michael Kueppers (2015). "Activity-Based Scheduling of Science Campaigns for the Rosetta Orbiter." In: *Proceedings of the 24th International Joint Conference on Artificial Intelligence.* Ed. by Qiang Yang and Michael Wooldridge. AAAI Press, pp. 4416–4422. URL: `http://ijcai.org/Abstract/15/655`

Mars Express (APSI)

Amedeo Cesta, Gabriella Cortellessa, Michel Denis, Alessandro Donati, Simone Fratini, Angelo Oddi, Nicola Policella, Erhard Rabenau, and Jonathan Schulster (2007). "Mexar2: AI Solves Mission Planner Problems." In: *IEEE Intelligent Systems* 22.4, pp. 12–19. DOI: `10.1109/MIS.2007.75`

What was missing?

Despite the great practical success of timeline-based systems, very little was known about the paradigm from a formal perspective.

In contrast, theory about action-based planning is well understood.

Our contribution

Our goal

To provide a comprehensive theoretical understanding of the timeline-based approach to planning.

In our recent works, we provided several results towards this goal:

- Expressiveness comparison with action-based languages
- Computational complexity of the planning problems
- Expressiveness of the language in logical terms

Expressiveness

We started our study from the expressiveness side.

- How do these problems relate to the action-based counterparts (e.g. PDDL)?
- Which problems can be expressed with timelines vs. actions and vice versa?

Critical syntactic elements

Timeline-based planning is a rich formalism.
Many elements non-trivially affect expressiveness and complexity:

- the bound on the solution **horizon**, given as input;
- **unbounded** temporal relations, such as

$$start(a) \leqslant_{[0,+\infty]} start(b)$$

- tokens of **unbounded** length, i.e. variables $x = (V_x, T_x, D_x, \gamma_x)$ where $D(v) = (0, +\infty)$ for some $v \in V$;
- constrained syntactic structure of the synchronization rules:

$$a_0[x_0 = v_0] \longrightarrow \exists a_1[x_1 = v_1] \ldots a_n[x_n = v_n] \,.\, \mathcal{E}_1 \vee \cdots \vee \mathcal{E}_m$$

  In particular:
  - fixed ∀∃ quantification scheme
  - only top-level disjunctions
  - no negation

Timelines vs. Actions

It turns out that timeline-based planning problems can fully capture temporal PDDL with:

- only tokens of bounded length;
- only bounded relations:

$$\cancel{start(a) \leqslant_{[0,+\infty]} start(b)}$$

Note: these restrictions are quite artificial, but they give us the smallest (yet) expressive enough instance of the problem.

Theorem

Given a temporal PDDL problem T, a timeline-based planning problem P can be built in polynomial time, such that:

- T admits a solution plan if P does as well
- a solution plan for T can be extracted in polynomial time from a solution plan for P, and vice versa

Timelines vs. Actions

It turns out that timeline-based planning problems can fully capture temporal PDDL with:

- only tokens of bounded length;
- only bounded relations:

$$\cancel{\mathit{start}(a) \leqslant_{[0,+\infty]} \mathit{start}(b)}$$

Note: these restrictions are quite artificial, but they give us the smallest (yet) expressive enough instance of the problem.

TIME 2016

Nicola Gigante, Angelo Montanari, Marta Cialdea Mayer, and Andrea Orlandini (2016). "Timelines Are Expressive Enough to Capture Action-Based Temporal Planning." In: *Proceedings of the 23rd International Symposium on Temporal Representation and Reasoning*. Ed. by Curtis E. Dyreson, Michael R. Hansen, and Luke Hunsberger. IEEE Computer Society, pp. 100–109. DOI: `10.1109/TIME.2016.18`

Complexity of timeline-based planning

Computational complexity was the most important theoretical property that we wanted to study.

- Finding a solution plan for a classical PDDL problem is PSPACE-complete (Bylander 1994)
- How about finding solution plans for a timeline-based planning problem $P = (SV, S)$?

Complexity results

results	complexity
unbounded horizon	EXPSPACE-complete
bounded horizon	NEXPTIME-complete
infinite horizon	EXPSPACE-complete

ICAPS 17

Nicola Gigante, Angelo Montanari, Marta Cialdea Mayer, and Andrea Orlandini (2017). “Complexity of Timeline-Based Planning.” In: *Proceedings of the 27th International Conference on Automated Planning and Scheduling*. Ed. by Laura Barbulescu, Jeremy Frank, Mausam, and Stephen F. Smith. AAAI Press, pp. 116–124. URL: `https://aaai.org/ocs/index.php/ICAPS/ICAPS17/paper/view/15758`

Complexity results

results	complexity
unbounded horizon	EXPSPACE-complete
bounded horizon	NEXPTIME-complete
infinite horizon	EXPSPACE-complete

KR 2018

Dario Della Monica, Nicola Gigante, Angelo Montanari, and Pietro Sala (2018). “A Novel Automata-Theoretic Approach to Timeline-Based Planning.” In: *Proceedings of the 16th International Conference on Principles of Knowledge Representation and Reasoning*. Ed. by Michael Thielscher, Francesca Toni, and Frank Wolter. AAAI Press, pp. 541–550. URL: `https://aaai.org/ocs/index.php/KR/KR18/paper/view/18024`

Complexity results

results	complexity
unbounded horizon	EXPSPACE-complete
bounded horizon	NEXPTIME-complete
infinite horizon	EXPSPACE-complete

There are also interesting **undecidability** results on **dense-time** by Bozzelli et al. (2018).

Tools from Formal Methods

We'll give a brief look at two results that involved tools from logic and formal methods.

- a logical characterization of timeline-based planning problems
- a game-theoretic approach to timeline-based planning with uncertainty

Timelines and Temporal Logic

Logical characterization of planning problems

Let’s go back to expressiveness issues.

We wanted to capture timeline-based planning problems with a well-behaved logical language.

Why?

- Logical characterizations are available:
  - for STRIPS-like planning (Cialdea Mayer et al. 2007) and
  - for temporal planning (Cimatti et al. 2017)
- Easier to compare the expressiveness of different languages if they are reduced to commonly known logics.
- Easier to think about the synthesis of controllers if the specification language is a well-defined logical formalism.

The result

We introduced a new logic,

Bounded TPTL with Past ($\mathsf{TPTL_b + P}$)

showing that:

- its satisfiability problem is EXPSPACE-complete, and
- it can capture a notable subset of the general formalism:
  - given a problem P, there is a $\mathsf{TPTL_b + P}$ formula ϕ_P such that ϕ_P is satisfiable iff there is a solution plan for P.

IJCAI 2017

D. Della Monica, N. Gigante, A. Montanari, G. Sciavicco, and P. Sala (2017). "Bounded Timed Propositional Temporal Logic with Past Captures Timeline-based Planning with Bounded Constraints." In: *Proc. of the 26th International Joint Conference on Artificial Intelligence*, pp. 1008–1014

Timed Propositional Temporal Logic

The TPTL logic was originally introduced for the verification of **real-time systems** (Alur and Henzinger 1994).

$$\phi := p \mid \neg\phi_1 \mid \phi_1 \vee \phi_2 \mid \overbrace{x.\phi_1}^{\substack{\text{freeze quantifier}\\ \downarrow}} \mid \overbrace{x \leqslant y + c \mid x \leqslant c}^{\substack{\text{timed constraints}\\ c\in\mathbb{Z}\\ \downarrow}}$$

$$\underbrace{\mid X\,\phi_1 \mid \phi_1\, \mathcal{U}\, \phi_2}_{\substack{\uparrow\\ \text{Linear Temporal Logic}\\ \text{temporal operators}}}$$

Formulae are interpreted over **timed words** (σ, τ), *i.e.*, each state σ_i is associated with a **timestamp** τ_i.

Example

p must hold infinitely often:

$$LTL: \quad GFp$$

Timed Propositional Temporal Logic

The TPTL logic was originally introduced for the verification of **real-time systems** (Alur and Henzinger 1994).

$$\phi := p \mid \neg\phi_1 \mid \phi_1 \vee \phi_2 \mid \overbrace{x.\phi_1}^{\substack{\text{freeze quantifier}\\ \downarrow}} \mid \overbrace{x \leqslant y + c \mid x \leqslant c}^{\substack{\text{timed constraints}\\ c\in\mathbb{Z}\\ \downarrow}}$$

$$\underbrace{\mid \mathsf{X}\,\phi_1 \mid \phi_1\, \mathcal{U}\, \phi_2}_{\substack{\uparrow\\ \text{Linear Temporal Logic}\\ \text{temporal operators}}}$$

Formulae are interpreted over **timed words** (σ, τ), *i.e.*, each state σ_i is associated with a **timestamp** τ_i.

Example

p must hold infinitely often and at least every five timesteps:

$$TPTL: \quad Gx.Fy.(p \wedge y \leqslant x + 5)$$

Timed Propositional Temporal Logic

The TPTL logic was originally introduced for the verification of real-time systems (Alur and Henzinger 1994).

$$\phi := p \mid \neg\phi_1 \mid \phi_1 \vee \phi_2 \mid \overbrace{x.\phi_1}^{\substack{\text{freeze quantifier}\\ \downarrow}} \mid \overbrace{x \leqslant y + c \mid x \leqslant c}^{\substack{\text{timed constraints}\\ c\in\mathbb{Z}\\ \downarrow}}$$

$$\underbrace{\mid \mathsf{X}\,\phi_1 \mid \phi_1 \,\mathcal{U}\, \phi_2}_{\substack{\uparrow\\ \text{Linear Temporal Logic}\\ \text{temporal operators}}}$$

The freeze quantifier binds the timestamp of the current state to a variable, used in the evaluation of the timed constraints.

Timed Propositional Temporal Logic

The TPTL logic was originally introduced for the verification of real-time systems (Alur and Henzinger 1994).

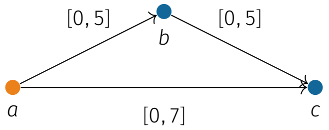
$$\phi := p \mid \neg\phi_1 \mid \phi_1 \vee \phi_2 \mid \overbrace{x.\phi_1}^{\substack{\text{freeze quantifier}\\ \downarrow}} \mid \overbrace{x \leqslant y + c \mid x \leqslant c}^{\substack{\text{timed constraints}\\ c\in\mathbb{Z}\\ \downarrow}}$$

$$\underbrace{\mid \mathrm{X}\,\phi_1 \mid \phi_1\,\mathcal{U}\,\phi_2}_{\substack{\uparrow\\ \text{Linear Temporal Logic}\\ \text{temporal operators}}}$$

The satisfiability problem for TPTL is EXPSPACE-complete.

Why TPTL?

The freeze quantifier and timed constraints allow one to compactly express constraints of this kind:

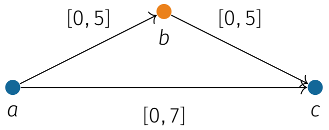


$$a[\ldots] \longrightarrow \exists b[\ldots]c[\ldots] \,.\, start(a) \leqslant_{[0,5]} start(b) \wedge start(b) \leqslant_{[0,5]} start(c) \wedge start(a) \leqslant_{[0,7]} start(c)$$

$$\mathsf{G}\, t_a.(p_a \longrightarrow \mathsf{F}\, t_b.(p_b \wedge t_b \leqslant t_a + 5 \wedge \mathsf{F}\, t_c.(p_c \wedge t_c <= t_b + 5 \wedge t_c <= t_a + 7)))$$

Why TPTL?

The freeze quantifier and timed constraints allow one to compactly express constraints of this kind:



$$b[\ldots] \longrightarrow \exists a[\ldots]c[\ldots]\ .\ \mathit{start}(a) \leqslant_{[0,5]} \mathit{start}(b) \land \mathit{start}(b) \leqslant_{[0,5]} \mathit{start}(c) \land \mathit{start}(a) \leqslant_{[0,7]} \mathit{start}(c)$$

But what if the trigger was token b?

TPTL with Past

We need past operators to encode synchronization rules, which can talk about future and past interchangeably.

$$\phi := p \mid \neg\phi_1 \mid \phi_1 \vee \phi_2 \mid \overbrace{x.\phi_1}^{\substack{\text{freeze quantifier}\\ \downarrow}} \mid \overbrace{x \leqslant y + c \mid x \leqslant c}^{\substack{\text{timed constraints}\\ c\in\mathbb{Z}\\ \downarrow}}$$

$$\underbrace{\mid \mathsf{X}\,\phi_1 \mid \phi_1\,\mathcal{U}\,\phi_2 \mid \mathsf{Y}\,\phi_1 \mid \phi_1\,\mathcal{S}\,\phi_2}_{\substack{\uparrow\\ \text{future } (\mathsf{X},\,\mathcal{U}) \text{ and past } (\mathsf{Y},\,\mathcal{S})\\ \text{temporal operators}}}$$

Unfortunately, however, past operators make the satisfiability problem become non elementary.

TPTL with Past

We need past operators to encode synchronization rules, which can talk about future and past interchangeably.

$$\phi := p \mid \neg\phi_1 \mid \phi_1 \vee \phi_2 \mid \overbrace{x.\phi_1}^{\substack{\text{freeze quantifier}\\ \downarrow}} \mid \overbrace{x \leqslant y + c \mid x \leqslant c}^{\substack{\text{timed constraints}\\ c\in\mathbb{Z}\\ \downarrow}}$$

$$\mid \underbrace{\mathsf{X}\,\phi_1 \mid \phi_1\,\mathcal{U}\,\phi_2 \mid \mathsf{Y}\,\phi_1 \mid \phi_1\,\mathcal{S}\,\phi_2}_{\substack{\uparrow\\ \text{future } (\mathsf{X},\,\mathcal{U}) \text{ and past } (\mathsf{Y},\,\mathcal{S})\\ \text{temporal operators}}}$$

Why? Together with the freeze quantifier, we can simulate first-order existential quantification.

$$\exists x \phi(x) \equiv y.\,\mathsf{F}\,x.\,\mathsf{P}\,z.(y = z \wedge \phi(x))$$

Bounded TPTL with Past

To recover an acceptable complexity while being still able to use past operators, we restricted the temporal operators with a bound w.

$$\phi := p \mid \neg\phi_1 \mid \phi_1 \vee \phi_2 \mid \overbrace{x.\phi_1}^{\substack{\text{freeze quantifier}\\ \downarrow}} \mid \overbrace{x \leqslant y + c \mid x \leqslant c}^{\substack{\text{timed constraints}\\ c\in\mathbb{Z}\\ \downarrow}}$$

$$\mid \underbrace{\mathsf{X}_w\, \phi_1 \mid \phi_1\, \mathcal{U}_w\, \phi_2 \mid \mathsf{Y}_w\, \phi_1 \mid \phi_1\, \mathcal{S}_w\, \phi_2}_{\substack{\uparrow\\ \text{MTL-like temporal operators}\\ w\in\mathbb{N}\cup\{\infty\}}}$$

The bound limits the timestamp of the states,
e.g., $X_5\phi$ holds at state σ_i iff ϕ holds at σ_{i+1} and $\tau_{i+1} - \tau_i \leqslant 5$.

Bounded TPTL with Past

To recover an acceptable complexity while being still able to use past operators, we **restricted** the temporal operators with a bound w.

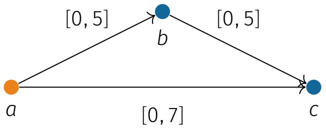
$$\phi := p \mid \neg\phi_1 \mid \phi_1 \vee \phi_2 \mid \overbrace{x.\phi_1}^{\substack{\text{freeze quantifier}\\ \downarrow}} \mid \overbrace{x \leqslant y + c \mid x \leqslant c}^{\substack{\text{timed constraints}\\ c\in\mathbb{Z}\\ \downarrow}}$$

$$\underbrace{\mid \mathsf{X}_w\,\phi_1 \mid \phi_1\,\mathcal{U}_w\,\phi_2 \mid \mathsf{Y}_w\,\phi_1 \mid \phi_1\,\mathcal{S}_w\,\phi_2}_{\substack{\uparrow\\ \text{MTL-like temporal operators}\\ w\in\mathbb{N}\cup\{\infty\}}}$$

The bound can be **omitted** (*i.e.*, $w = \infty$) only if the underlying formulae are **closed**, *i.e.*, they do not refer to variables quantified outside.

$\mathsf{TPTL_b + P}$ Example

Now the previous example can be encoded in both cases:

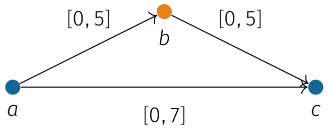


$$a[\ldots] \longrightarrow \exists b[\ldots]c[\ldots]\,.\,\mathit{start}(a) \leqslant_{[0,5]} \mathit{start}(b) \wedge$$
$$\mathit{start}(b) \leqslant_{[0,5]} \mathit{start}(c) \wedge \mathit{start}(a) \leqslant_{[0,7]} \mathit{start}(c)$$

$$\mathsf{G}\, t_a.(p_a \longrightarrow \mathsf{F}_{10}\, \mathsf{P}_{10}\, t_b.(p_b \wedge \mathsf{F}_{10}\, \mathsf{P}_{10}\, t_c.(p_c \wedge$$
$$t_b \leqslant t_a + 5 \wedge t_c \leqslant t_b + 5 \wedge t_c \leqslant t_a + 7)))$$

$\mathsf{TPTL_b + P}$ Example

Now the previous example can be encoded in both cases:



$$b[\ldots] \longrightarrow \exists a[\ldots]c[\ldots] \,.\, start(a) \leqslant_{[0,5]} start(b) \wedge \\ start(b) \leqslant_{[0,5]} start(c) \wedge start(a) \leqslant_{[0,7]} start(c)$$

$$\mathsf{G}\, t_b.(p_b \longrightarrow \mathsf{F}_{10}\, \mathsf{P}_{10}\, t_a.(p_a \wedge \mathsf{F}_{10}\, \mathsf{P}_{10}\, t_c.(p_c \wedge \\ t_b \leqslant t_a + 5 \wedge t_c \leqslant t_b + 5 \wedge t_c \leqslant t_a + 7)))$$

The encoding

Thus, the encoding generally works this way. Given a rule:

$$a[x = v] \longrightarrow \exists b[y = v']c[z = v''] \,.\, \mathcal{C}$$

- We decompose the corresponding rule graph into its **bounded components**
- If the unbounded edges between the components form a tree, we can encode the rule in $\mathsf{TPTL_b} + \mathsf{P}$
- Each component is found by a combination of **bounded** operators
- The different components are related by means of **unbounded** operators

Complexity of $\mathsf{TPTL_b + P}$

The complexity of $\mathsf{TPTL_b + P}$ is proved by providing two **tableau methods**.

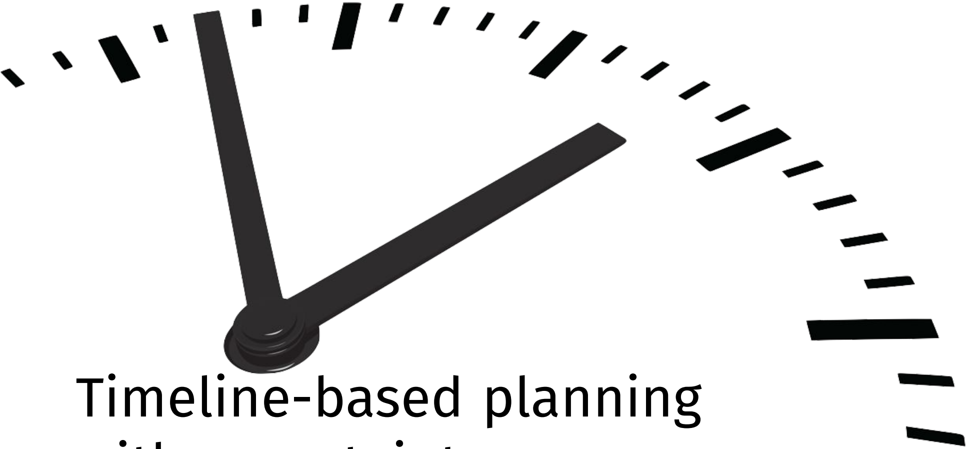
- An adaptation of the graph-shaped tableau for TPTL by Alur and Henzinger (1994)
- An adaptation of the **one-pass tree-shaped** tableau for LTL by Reynolds (2016)

LPAR-21

Nicola Gigante, Angelo Montanari, and Mark Reynolds (2017). "A One-Pass Tree-Shaped Tableau for LTL+Past." In: *Proceedings of the 21st International Conference on Logic for Programming, Artificial Intelligence and Reasoning*. Ed. by Thomas Eiter and David Sands. Vol. 46. EPiC Series in Computing. EasyChair, pp. 456–473. URL: `http://www.easychair.org/publications/paper/340363`

GandALF 2018

Luca Geatti, Nicola Gigante, Angelo Montanari, and Mark Reynolds (2018). "One-Pass and Tree-Shaped Tableau Systems for TPTL and TPTLb+Past." In: *Proceedings of the 9th International Symposium on Games, Automata, Logics, and Formal Verification*. Ed. by Andrea Orlandini and Martin Zimmermann. Vol. 277. EPTCS, pp. 176–190. DOI: `10.4204/EPTCS.277.13`



Timeline-based planning with uncertainty

Timeline-based planning with uncertainty

Real-world scenarios need to account for the inherent uncertainty and nondeterminism coming from the interaction with the environment.

Timeline-based systems handle temporal uncertainty by means of flexible plans:

- envelopes of plans that differ in the precise timing of events
- dynamic controllability checking ensures execution
- re-planning occurs when the planner’s predictions fail

Game-theoretic approach

We propose to approach timeline-based planning with uncertainty in game-theoretic terms.

- We define the timeline-based planning game as a two-player game;
- the controller tries to satisfy the given set of synchronization rules;
- the environment plays arbitrarily.

Timeline-based games

A timeline-based game is a tuple $G = (SV_C, SV_E, S, D)$.

- Two players, **Charlie** (the controller) and **Eve** (the environment);
- players play by starting and ending tokens, building a plan;
- Charlie can start tokens for variables in SV_C, Eve those for variable in SV_E;
- Charlie decides when to stop **controllable** tokens, while Eve decides when to stop **uncontrollable** ones;
- Charlie tries to satisfy the set S of **system rules**, whatever the behavior of Eve;
- both players are assumed to play as to satisfy the set D of **domain rules**.

Strategies

We want to guarantee the existence of a winning strategy for Charlie.

- a strategy is a function σ that given a partial plan gives the next move of the player (i.e. which token to start/end, if any).
- a strategy σ is **admissible** if any play played according to σ will eventually satisfy $\mathcal{D}$.
- a strategy σ_C for Charlie is **winning** if, for any admissible strategy σ_E for Eve, any play played according to σ_C and σ_E is going to satisfy $S \cup D$.

Winning strategies

Charlie has a winning strategy if he can play to satisfy the rules no matter what Eve does, supposing rules in D are satisfied.

- a general form of nondeterminism is handled in this way, not only temporal uncertainty;
- no need for re-planning, as the winning strategy can already handle any behavior of Eve;
- greater modeling flexibility: domain rules allow to describe complex interactions between the agent and the environment;
- provably subsumes the approach based on dynamically controllable flexible plans;

Finding a winning strategy

How hard is to find a winning strategy?
2EXPTIME-complete

Finding a winning strategy

The decision procedure is based on ATL* model-checking over concurrent game structures (Alur, Henzinger, and Kupferman 2002):

- concurrent game structures (CGS) are a general formalism to represent multi-agent concurrent systems.
- Alternating-time Temporal Logic (ATL) and its generalization ATL*, are interpreted over CGSs;
- ATL and ATL* are similar to CTL and CTL*, but branching modalities quantify over paths played according to specific strategies of a specific set of players;

Finding a winning strategy

A doubly exponential size (turn-based synchronous) CGS can be built to represent the game;

- nodes are partial plans, edges labeled by players moves;
- particular attention to guarantee a finite state space;
- states where D and S are satisfied are labelled, respectively, by proposition letters d and w;
- The winning condition is then encoded in ATL* as follows:

$$\phi \equiv \langle\langle 1 \rangle\rangle \big(\mathsf{F} d \to \mathsf{F} w \big)$$

- Model-checking a fixed-size ATL* formula over a CGS can be done in polynomial time, hence the 2EXPTIME complexity.
- Hardness proved with a reduction from a particular domino-tiling game.

Making the state space finite

Abstractly, the state space of the game is infinite:

- each synchronization rule can in principle be affected by anything happening arbitrarily far in the past or in the future;
- but we do not really need to keep track of all the history;
    - bounded contraints can be checked locally;
    - for unbounded constraints we can just keep in mind what happened in the past or not, instead of when;

Making the state space finite

Abstractly, the state space of the game is infinite:

- each synchronization rule can in principle be affected by anything happening arbitrarily far in the past or in the future;
- but we do not really need to keep track of all the history;
- a suitable compact representation $[\pi]$ of a partial plan π can be built such that:
  1. we can decide if π satisfies the rules looking only at $[\pi]$;
  2. given a round ρ of the game we can build $[\rho(\pi)]$ from $[\pi]$;
- then, states of the game are all the possible $[\pi]$;
  - a single state encapsulates all important data about the history of the game: the game is **positional**;
- size of $[\pi]$ is exponential, hence the state space is doubly exponential.

Conclusions

Timeline-based planning is a rich formalism, with interesting formal properties.

- we provided the first thorough theoretical investigation of timeline-based planning problems and languages
- expressiveness and computational complexity
- logical characterization
- game-theoretic approach to timeline-based planning with uncertainty

Future work

There are many open questions and future paths:

- logical characterization for the full problem
- synthesis of controllers for timeline-based games
- comparison of timeline-based games with FOND planning and similar
- extension of timeline-based games to more complex scenarios (multi- player, distributed, probabilistic, ...)

Thank you

Questions?

Bibliography

Alur, R. and T. A. Henzinger (1994). “A Really Temporal Logic.” In: *Journal of the ACM* 41.1, pp. 181–204. DOI: 10.1145/174644.174651. URL: http://doi.acm.org/10.1145/174644.174651.

Alur, R., T. A. Henzinger, and O. Kupferman (2002). “Alternating-time Temporal Logic.” In: *Journal of the ACM* 49.5, pp. 672–713.

Bedrax-Weiss, Tania, Conor McGann, Andrew Bachmann, Will Edgington, and Michael Iatauro (2005). *EUROPA2: User and contributor guide*. Tech. rep. NASA Ames Research Center.

Bozzelli, Laura, Alberto Molinari, Angelo Montanari, and Adriano Peron (2018). “Decidability and Complexity of Timeline-Based Planning over Dense Temporal Domains.” In: *Proceedings of the 16th International Conference on Principles of Knowledge Representation and Reasoning*. Ed. by Michael Thielscher, Francesca Toni, and Frank Wolter. AAAI Press, pp. 627–628. URL: https://aaai.org/ocs/index.php/KR/KR18/paper/view/17995.

Bylander, T. (1994). “The Computational Complexity of Propositional STRIPS Planning.” In: *Artificial Intelligence* 69.1-2, pp. 165–204.

Cesta, Amedeo, Gabriella Cortellessa, Michel Denis, Alessandro Donati, Simone Fratini, Angelo Oddi, Nicola Policella, Erhard Rabenau, and Jonathan Schulster (2007). “Mexar2: AI Solves Mission Planner Problems.” In: *IEEE Intelligent Systems* 22.4, pp. 12–19. DOI: 10.1109/MIS.2007.75.

Chien, Steve A., Gregg Rabideau, Daniel Tran, Martina Troesch, Joshua Doubleday, Federico Nespoli, Miguel Perez Ayucar, Marc Costa Sitja, Claire Vallat, Bernhard Geiger, Nico Altobelli, Manuel Fernandez, Fran Vallejo, Rafael Andres, and Michael Kueppers (2015). “Activity-Based Scheduling of Science Campaigns for the Rosetta Orbiter.” In: *Proceedings of the 24th International Joint Conference on Artificial Intelligence*. Ed. by Qiang Yang and Michael Wooldridge. AAAI Press, pp. 4416–4422. URL: http://ijcai.org/Abstract/15/655.

Cialdea Mayer, Marta, Carla Limongelli, Andrea Orlandini, and Valentina Poggioni (2007). “Linear temporal logic as an executable semantics for planning languages.” In: *Journal of Logic, Language and Information* 16.1, pp. 63–89. DOI: 10.1007/s10849-006-9022-1.

Bibliography

Cimatti, A., A. Micheli, and M. Roveri (2017). “Validating Domains and Plans for Temporal Planning via Encoding into Infinite-State Linear Temporal Logic.” In: *Proc. of the 31st AAAI Conference on Artificial Intelligence*, pp. 3547–3554.

Della Monica, D., N. Gigante, A. Montanari, G. Sciavicco, and P. Sala (2017). “Bounded Timed Propositional Temporal Logic with Past Captures Timeline-based Planning with Bounded Constraints.” In: *Proc. of the 26th International Joint Conference on Artificial Intelligence*, pp. 1008–1014.

Della Monica, Dario, Nicola Gigante, Angelo Montanari, and Pietro Sala (2018). “A Novel Automata-Theoretic Approach to Timeline-Based Planning.” In: *Proceedings of the 16th International Conference on Principles of Knowledge Representation and Reasoning*. Ed. by Michael Thielscher, Francesca Toni, and Frank Wolter. AAAI Press, pp. 541–550. URL: `https://aaai.org/ocs/index.php/KR/KR18/paper/view/18024`.

Fratini, Simone, Amedeo Cesta, Andrea Orlandini, Riccardo Rasconi, and Riccardo De Benedictis (2011). “APSI-based Deliberation in Goal Oriented Autonomous Controllers.” In: *ASTRA 2011*. Vol. 11. ESA.

Fratini, Simone and L. Donati (2011). *APSI Timeline Representation Framework v. 3.0*. Tech. rep. European Space Agency - ESOC.

Geatti, Luca, Nicola Gigante, Angelo Montanari, and Mark Reynolds (2018). “One-Pass and Tree-Shaped Tableau Systems for TPTL and TPTLb+Past.” In: *Proceedings of the 9th International Symposium on Games, Automata, Logics, and Formal Verification*. Ed. by Andrea Orlandini and Martin Zimmermann. Vol. 277. EPTCS, pp. 176–190. DOI: `10.4204/EPTCS.277.13`.

Gigante, Nicola, Angelo Montanari, Marta Cialdea Mayer, and Andrea Orlandini (2016). “Timelines Are Expressive Enough to Capture Action-Based Temporal Planning.” In: *Proceedings of the 23rd International Symposium on Temporal Representation and Reasoning*. Ed. by Curtis E. Dyreson, Michael R. Hansen, and Luke Hunsberger. IEEE Computer Society, pp. 100–109. DOI: `10.1109/TIME.2016.18`.

Bibliography

Gigante, Nicola, Angelo Montanari, Marta Cialdea Mayer, and Andrea Orlandini (2017). "Complexity of Timeline-Based Planning." In: *Proceedings of the 27th International Conference on Automated Planning and Scheduling*. Ed. by Laura Barbulescu, Jeremy Frank, Mausam, and Stephen F. Smith. AAAI Press, pp. 116–124. URL: https://aaai.org/ocs/index.php/ICAPS/ICAPS17/paper/view/15758.

Gigante, Nicola, Angelo Montanari, and Mark Reynolds (2017). "A One-Pass Tree-Shaped Tableau for LTL+Past." In: *Proceedings of the 21st International Conference on Logic for Programming, Artificial Intelligence and Reasoning*. Ed. by Thomas Eiter and David Sands. Vol. 46. EPiC Series in Computing. EasyChair, pp. 456–473. URL: http://www.easychair.org/publications/paper/340363.

Muscettola, N. (1994). "HSTS: Integrating Planning and Scheduling." In: *Intelligent Scheduling*. Ed. by Monte Zweben and Mark S. Fox. Morgan Kaufmann. Chap. 6, pp. 169–212.

Reynolds, Mark (2016). "A New Rule for LTL Tableaux." In: *Proceedings of the 7th International Symposium on Games, Automata, Logics and Formal Verification*. Vol. 226. EPTCS, pp. 287–301. DOI: 10.4204/EPTCS.226.20.