Time Flies When Looking out of the Window: Timed Games with Window Parity Objectives

James C. A. Main [1] Mickael Randour [1,2] Jeremy Sproston [3]

[1]UMONS – Université de Mons, Belgium

[2]F.R.S.-FNRS, Belgium

[3]Università degli Studi di Torino, Italy

CONCUR 2021

Overview

- In this talk, we cover window parity objectives in a timed setting.

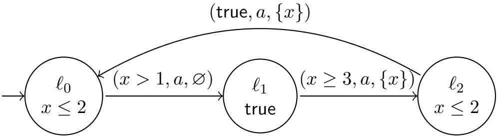
Window parity objectives

For given bound λ on the size of windows, the direct timed window parity objective requires that at all times along a play, there is a window of size less than λ in which the smallest priority is even. The timed window parity objective requires the direct objective to hold from some point on.

- We discuss verification of timed automata and realizability in timed games for timed window parity objectives.
- Each problem can be solved by a reduction to the same problem for safety (direct case) or co-Büchi objectives (non-direct case).

Timed automata

- Timed automata [AD94] are used to model real-time systems.
- The elapse of time is measured by a finite number of clock variables, or clocks, that progress at the same rate.
- Clock constraints are conjunctions of conditions of the form $x \leq c$, $x < c$, $x \geq c$ and $x > c$ where x is a clock and c a natural number.



- Timed automata consist of:
  - a finite set of locations constrained by invariants with a distinguished initial location ℓ_{init} and
  - a finite set of edges labeled by guards, actions and clocks to reset.

Timed automata

We always assume there is a clock γ that cannot be reset.

Semantics of timed automata

A timed automaton gives rise to an uncountable transition system.

- States of this transition system are pairs of locations and clock valuations (mappings assigning a non-negative real number to each clock of the automaton). The initial state is $(\ell_{\mathsf{init}}, \mathbf{0}^C)$.
- Moves are pairs (d, a) where d is a delay (non-negative real number) and a is an action of the timed automaton or a special standby action $\perp$.
- Transitions are constrained by the invariants and guards of the timed automaton.
- A path of a timed automaton is an infinite sequence of states and moves following transitions.

Verification of timed automata

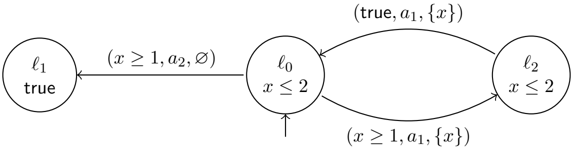
- Given a specification as an objective, i.e., a set of correct sequences of states, we wish to check that all sequence of states derived from paths of the timed automaton comply with the objective.
- However, not all paths of a timed automaton are meaningful.
- A path of a timed automaton is time-convergent if the valuation of γ is bounded along the path. Otherwise, the path is time-divergent.

Verification problem

Given an objective, check whether all time-divergent initial paths of a timed automaton comply with the objective.

Timed games

- We consider two-player games played on timed automata.
- A timed game is given by a timed automaton and a partition of the actions of the timed automaton in $\mathcal{P}_1$ actions and $\mathcal{P}_2$ actions.
- These games are concurrent: at each round, both players present a move and the play proceeds following a fastest move – a transition is chosen non-deterministically if both players present moves with the same delay.



- Example 1: $(\ell_0, 0)\,((1, a_1), (1, a_2))\,(\ell_1, 1)$
- Example 2: $(\ell_0, 0)\,((1, a_1), (1, a_2))\,(\ell_2, 0)$

Timed games

- Plays are non-terminating: a play is a sequence of alternating states of the transition system underlying the timed automaton and pairs consisting of $\mathcal{P}_1$ and $\mathcal{P}_2$ moves.
- A strategy for $\mathcal{P}_i$ is a function mapping finite prefixes of plays ending in states to moves of $\mathcal{P}_i$.

Winning in timed games

- Due to the phenomenon of time-convergence, we distinguish objectives and winning conditions, following [dAFH+03].
- Given an objective, we say a play belongs to its associated winning condition if one of the two following conditions is fulfilled:
  - the play is time-divergent and satisfies the objective;
  - the play is time-convergent and from some point on, transitions in the play cannot be achieved by $\mathcal{P}_1$'s moves.
- We say a strategy is winning from some initial state if all plays starting in this state consistent with the strategy satisfy the winning condition.

Realizability problem

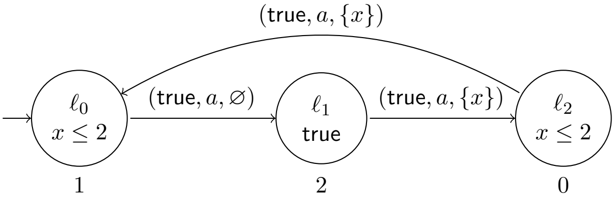
Given an objective, check whether $\mathcal{P}_1$ has a winning strategy from the initial state.

Objectives of interest

- Safety objective: for a set of locations F, the safety objective over F, denoted by Safe(F), consists of sequences of states along which no location in F ever appears.
- Co-Büchi objective: for a set of locations F, the co-Büchi objective over F, denoted by coBüchi(F), consists of sequences of states along which no location in F appears infinitely often.
- Parity objective: given a priority function p mapping a non-negative integer to locations, the parity objective Parity(p) consists of sequences of states along which the smallest priority appearing infinitely often is even.

Windows

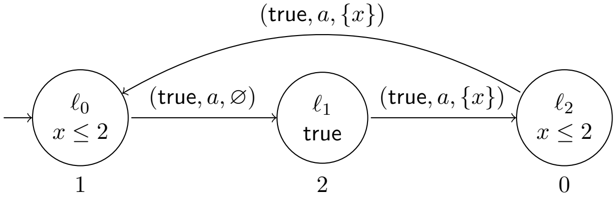
- For the classical parity objective, there are no timing constraints between odd priorities and smaller even priorities.



- Through the window mechanism, one can specify such timing constraints.

Good windows

- The window objectives are based on the notion of good windows.
- Fix a bound λ on the length of windows. A good window for the parity objective is a window in which:
  - strictly less than λ time units elapse and
  - the smallest priority appearing in the window is even.



Examples for $\lambda = 2$ (global clock γ omitted from states):

- $((\ell_0, 0)(1, a)(\ell_1, 1)(0, a)(\ell_2, 0)(0, a))^\omega \rightsquigarrow$ good window at the start
- $((\ell_0, 0)(1, a)(\ell_1, 1)(1.2, a)(\ell_2, 0)(0, a))^\omega \rightsquigarrow$ bad window at the start

Good windows

- Timed good window parity objective: the window at the start of the path or play is good. Formally, let $\mathsf{TGW}(\lambda)$ be

$$\{(\ell_0, \nu_0)(\ell_1, \nu_1)\ldots \mid \exists\, n,\ (\min_{0\leq k\leq n} p(\ell_k)) \bmod 2 = 0 \land (\nu_n - \nu_0)(\gamma) < \lambda\}.$$

- We say that the window opened at some step j closes at step n if n satisfies

$$(\min_{j\leq k\leq n} p(\ell_k)) \bmod 2 = 0 \land \forall\, j \leq n' < n,\ (\min_{j\leq k\leq n'} p(\ell_k)) \bmod 2 = 1.$$

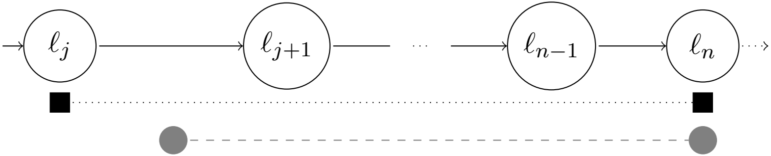
- If a window does not close in strictly less than λ time units, we say that this window is bad.

Timed window parity objectives

- Direct timed window parity objective: there is a good window at all steps. Let $\mathsf{DTW}(\lambda)$ be

$$\{(\ell_0, \nu_0)(\ell_1, \nu_1)\ldots \mid \forall\, n,\ (\ell_n, \nu_n)(\ell_{n+1}, \nu_{n+1})\ldots \in \mathsf{TGW}(\lambda)\}.$$

This objective is equivalent to requiring good windows even in intermediate states occurring during delays.



- Timed window parity objective: the direct window parity holds from some point on. Let $\mathsf{TW}(\lambda)$ be

$$\{(\ell_0, \nu_0)(\ell_1, \nu_1)\ldots \mid \exists\, n,\ (\ell_n, \nu_n)(\ell_{n+1}, \nu_{n+1})\ldots \in \mathsf{DTW}(\lambda)\}.$$

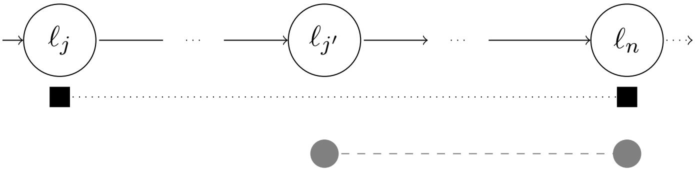
Inductive property of windows

The key to our reduction is the inductive property of windows.

Inductive property of windows

Along all paths of a timed automaton or plays of a timed game, for all j, if the window opened at step j closes at step n in strictly less than λ time units, then for all $j \leq j' \leq n$, the window opened at step j' is good.

$p(\ell_n) = \min_{j \leq k \leq n} p(\ell_k)$



Reduction

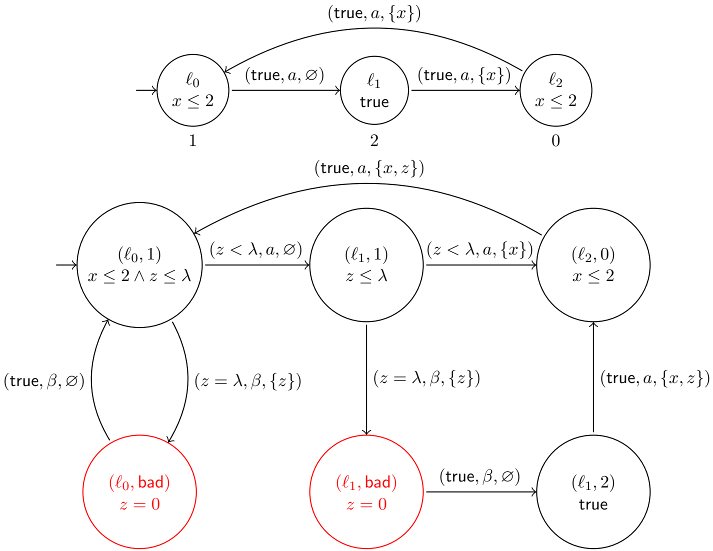
$\rightsquigarrow$ The inductive property implies that it suffices to keep track of one window at a time.

- One can reduce the verification and realizability problems for the direct timed window parity objective to the verification and realizability problems for the safety objective respectively.
- One can reduce the verification and realizability problems for the timed window parity objective to the verification and realizability problems for the co-Büchi objective respectively.

Reduction

- We expand timed automata to include information on the current window to detect bad windows.
- We expand locations to encode the current lowest priority of the window or a special value bad to be avoided.
- We introduce a new clock z to measure how long the current window has been open.
- We change guards and invariants so that bad locations are visited whenever a bad window is witnessed.
- For each player, we add a new action to enter and exit bad locations, written β_1 and β_2.

Example of the reduction



Correctness of the reduction

Correctness can be proven using two mappings: an expansion mapping Ex and a projection mapping Pr:

- Ex maps paths (resp. plays) of a timed automaton (resp. game) to paths (resp. plays) of its expansion;
- Pr maps paths (resp. plays) of an expanded timed automaton (resp. game) to paths (resp. plays) of the original one.

Correctness of the reduction

- For all time-divergent paths or plays π, π satisfies $\mathsf{DTW}(\lambda)$ (resp. $\mathsf{TW}(\lambda)$) if and only if $\mathsf{Ex}(\pi)$ satisfies $\mathsf{Safe}(\mathsf{Bad})$ (resp. $\mathsf{coBüchi}(\mathsf{Bad})$).
- For all time-divergent initial paths or plays π of an expanded timed automaton or game, π satisfies $\mathsf{Safe}(\mathsf{Bad})$ (resp. $\mathsf{coBüchi}(\mathsf{Bad})$) if and only if $\mathsf{Pr}(\pi)$ satisfies $\mathsf{DTW}(\lambda)$ (resp. $\mathsf{TW}(\lambda)$).

Theorem (Correctness for verification)

- *All time-divergent initial paths of a timed automaton satisfy a direct timed window parity objective if and only if all time-divergent initial paths of its expansion satisfy a safety objective over bad locations.*
- *All time-divergent initial paths of a timed automaton satisfy a timed window parity objective if and only if all time-divergent initial paths of its expansion satisfy a co-Büchi objective over bad locations.*

Correctness of the reduction

The mappings Ex and Pr can be used to translate winning strategies between a timed game and its expansion.

- The expansion mapping can be used to translate strategies of an expanded timed game to strategies of the original timed game.

  Roughly: $\bar{\sigma}$ translated to $\bar{\sigma} \circ \mathsf{Ex}$

- The projection mapping can be used to translate strategies of a timed game to strategies of its expansion.

  Roughly: σ translated to $\sigma \circ \mathsf{Pr}$

Correctness of the reduction

For a timed game $\mathcal{G}$, let $\mathcal{G}(\lambda)$ denote its expansion.

Theorem

Let s_{init} be the initial state of $\mathcal{G}$ and $\bar{s}_{\mathsf{init}}$ be the initial state of $\mathcal{G}(\lambda)$.

- *There is a winning strategy σ for $\mathcal{P}_1$ for the objective* $\mathsf{DTW}(\lambda)$ *from s_{init} in $\mathcal{G}$ if and only if there is a winning strategy $\bar{\sigma}$ for $\mathcal{P}_1$ for the objective* $\mathsf{Safe}(\mathsf{Bad})$ *from $\bar{s}_{\mathsf{init}}$ in $\mathcal{G}(\lambda)$.*
- *There is a winning strategy σ for $\mathcal{P}_1$ for the objective* $\mathsf{TW}(\lambda)$ *from s_{init} in $\mathcal{G}$ if and only if there is a winning strategy $\bar{\sigma}$ for $\mathcal{P}_1$ for the objective* $\mathsf{coBüchi}(\mathsf{Bad})$ *from $\bar{s}_{\mathsf{init}}$ in $\mathcal{G}(\lambda)$.*

Multi-dimensional objectives

- The reduction can be adapted for conjunctions of direct timed window parity objectives and conjunctions of timed window parity objectives.
- By the inductive property, we need only keep track of one window per dimension.
- The construction is similar: locations are expanded with vectors of priorities and one new clock per objective is introduced.

Complexity results

- The reduction yields a PSPACE algorithm for the verification problem and an EXPTIME algorithm for the realizability problem, even for multiple dimensions.
- Hardness can be established by reducing the verification and realizability problems for safety objectives to the verification and realizability problem for direct or non-direct timed window parity objectives.

Complexity summary

	Single dimension	Multiple dimensions
Timed automata	PSPACE-complete	PSPACE-complete
Timed games	EXPTIME-complete	EXPTIME-complete
Games (untimed) [BHR16]	P-complete	EXPTIME-complete

References I

Rajeev Alur and David L. Dill.
A theory of timed automata.
Theor. Comput. Sci., 126(2):183–235, 1994.

Thomas Brihaye, Florent Delgrange, Youssouf Oualhadj, and Mickael Randour.
Life is random, time is not: Markov decision processes with window objectives.
Log. Methods Comput. Sci., 16(4), 2020.

Véronique Bruyère, Quentin Hautem, and Mickael Randour.
Window parity games: an alternative approach toward parity games with time bounds.
In Domenico Cantone and Giorgio Delzanno, editors, *Proceedings of the Seventh International Symposium on Games, Automata, Logics*

References II

and Formal Verification, GandALF 2016, Catania, Italy, 14-16 September 2016, volume 226 of *EPTCS*, pages 135–148, 2016.

Krishnendu Chatterjee, Laurent Doyen, Mickael Randour, and Jean-François Raskin.
Looking at mean-payoff and total-payoff through windows.
Inf. Comput., 242:25–52, 2015.

Luca de Alfaro, Marco Faella, Thomas A. Henzinger, Rupak Majumdar, and Mariëlle Stoelinga.
The element of surprise in timed games.
In Roberto M. Amadio and Denis Lugiez, editors, *CONCUR 2003 - Concurrency Theory, 14th International Conference, Marseille, France, September 3-5, 2003, Proceedings*, volume 2761 of *Lecture Notes in Computer Science*, pages 142–156. Springer, 2003.

Teaser (1/2)

Window objectives have been studied in discrete-time settings:

- in turn-based games with mean-payoff and total-payoff objectives [CDRR15];
- in turn-based games with parity objectives [BHR16];
- in Markov decision processes for parity and mean-payoff objectives [BDOR20].

We extend window objectives to a continuous-time setting, for timed automata and timed games.

Teaser (2/2)

- In a nutshell, the direct timed window parity objective requires, for a fixed bound λ on the size of windows, that at all times along a play, there is a window of size at most λ in which the smallest priority is even.
- We also consider a prefix-independent variant, requiring the direct objective to hold from some point forward.
- For these objectives, verification of timed automata is PSPACE-complete and realizability in timed games is EXPTIME-complete.